

INTERNATIONAL STANDARD

IEC
61158-4

Third edition
2003-05

**Digital data communications
for measurement and control –
Fieldbus for use in industrial
control systems –**

**Part 4:
Data link protocol specification**



Reference number
IEC 61158-4:2003(E)

Publication numbering

As from 1 January 1997 all IEC publications are issued with a designation in the 60000 series. For example, IEC 34-1 is now referred to as IEC 60034-1.

Consolidated editions

The IEC is now publishing consolidated versions of its publications. For example, edition numbers 1.0, 1.1 and 1.2 refer, respectively, to the base publication, the base publication incorporating amendment 1 and the base publication incorporating amendments 1 and 2.

Further information on IEC publications

The technical content of IEC publications is kept under constant review by the IEC, thus ensuring that the content reflects current technology. Information relating to this publication, including its validity, is available in the IEC Catalogue of publications (see below) in addition to new editions, amendments and corrigenda. Information on the subjects under consideration and work in progress undertaken by the technical committee which has prepared this publication, as well as the list of publications issued, is also available from the following:

- **IEC Web Site** (www.iec.ch)

- **Catalogue of IEC publications**

The on-line catalogue on the IEC web site (http://www.iec.ch/searchpub/cur_fut.htm) enables you to search by a variety of criteria including text searches, technical committees and date of publication. On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

This summary of recently issued publications (http://www.iec.ch/online_news/justpub/jp_entry.htm) is also available by email. Please contact the Customer Service Centre (see below) for further information.

- **Customer Service Centre**

If you have any questions regarding this publication or need further assistance, please contact the Customer Service Centre:

Email: custserv@iec.ch
Tel: +41 22 919 02 11
Fax: +41 22 919 03 00

INTERNATIONAL STANDARD

IEC 61158-4

Third edition
2003-05

Digital data communications for measurement and control – Fieldbus for use in industrial control systems –

Part 4: Data link protocol specification

© IEC 2003 — Copyright - all rights reserved

No part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from the publisher.

International Electrotechnical Commission, 3, rue de Varembé, PO Box 131, CH-1211 Geneva 20, Switzerland
Telephone: +41 22 919 02 11 Telefax: +41 22 919 03 00 E-mail: inmail@iec.ch Web: www.iec.ch



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE **XW**

For price, see current catalogue

CONTENTS

FOREWORD.....	24
0 Introduction.....	27
0.1 General.....	27
0.2 Nomenclature for references within this standard	27
1 Scope and object	29
1.1 Overview	29
1.2 Specifications.....	30
1.3 Procedures.....	32
1.4 Applicability.....	32
1.5 Conformance.....	32
2 Normative references.....	32
3 Terms and definitions.....	33
3.1 Reference model terms and definitions.....	33
3.2 Service convention terms and definitions.....	35
3.3 Common terms and definitions	36
3.4 Type 1: Additional terms and definitions	39
3.5 Type 2: Additional terms and definitions	45
3.6 Type 3: Additional terms and definitions	52
3.7 Type 4: Additional terms and definitions	54
3.8 Type 5: Additional terms and definitions	55
3.9 Type 6: Additional terms and definitions	55
3.10 Type 7: Additional terms and definitions	65
3.11 Type 8: Additional terms and definitions	67
4 Symbols and abbreviations	69
4.1 Common symbols and abbreviations	69
4.2 Type 1: Additional symbols and abbreviations	69
4.3 Type 2: Additional symbols and abbreviations	73
4.4 Type 3: Additional symbols and abbreviations	74
4.5 Type 4: Additional symbols and abbreviations	78
4.6 Type 5: Additional symbols and abbreviations	79
4.7 Type 6: Additional symbols and abbreviations	79
4.8 Type 7: Additional symbols and abbreviations	80
4.9 Type 8: Additional symbols and abbreviations	81
5 DL-protocol elements common to multiple DL-protocol Types	84
5.1 Frame check sequence.....	84
6 Type 1: Overview of the DL-protocol	88
6.1 Three-level model of the DLL	88
6.2 Service provided by the DLL.....	91
6.3 Structure and definition of DL-addresses	97
6.4 Service assumed from the PhL	109
6.5 Functions of the DLL	112
6.6 Functional classes.....	114
6.7 Local parameters, variables, counters, timers and queues.....	116

7	Type 1: General structure and encoding of PhIDUs and DLPDUs, and related elements of procedure	130
7.1	PhIDU structure and encoding	130
7.2	Common DLPDU structure, encoding and elements of procedure	130
8	Type 1: DLPDU-specific structure, encoding and elements of procedure.....	139
8.1	Establish Connection (EC) DLPDU	141
8.2	Disconnect Connection (DC) DLPDU	143
8.3	Reset Connection (RC) DLPDU	146
8.4	Compel Acknowledgement (CA) DLPDU	147
8.5	Compel Data (CD) DLPDU	154
8.6	Exchange Data (ED) DLPDU	162
8.7	Data (DT) DLPDU.....	170
8.8	Status Response (SR) DLPDU	178
8.9	Compel Time (CT) DLPDU	181
8.10	Time Distribution (TD) DLPDU.....	182
8.11	Round-Trip-Delay Query (RQ) DLPDU	184
8.12	Round-Trip-Delay Reply (RR) DLPDU	186
8.13	Probe Node DL-address (PN) DLPDU.....	188
8.14	Probe Response (PR) DLPDU	191
8.15	Pass Token (PT) DLPDU	192
8.16	Execute Sequence (ES) DLPDU	201
8.17	Return Token (RT) DLPDU	208
8.18	Request Interval (RI) DLPDU.....	209
8.19	Claim LAS (CL) DLPDU	210
8.20	Transfer LAS (TL) DLPDU	212
8.21	Wakeup (WK) DLPDU.....	216
8.22	Idle (IDLE) DLPDU	217
8.23	Spare DLPDUs	218
8.24	Reserved (not to be used) DLPDUs.....	220
9	Type 1: DLPDU-parameter structure and encoding	221
9.1	Structure and encoding of EC-PARAMETERS	221
9.2	Structure and encoding of DC-PARAMETERS	226
9.3	Structure and encoding of RC-PARAMETERS	226
9.4	Structure and encoding of SD-Parameters.....	228
9.5	Structure and encoding of SR-parameters	235
9.6	Structure and encoding of TD-parameters	237
9.7	Structure and encoding of RQ-parameters.....	239
9.8	Structure and encoding of RR-parameters	240
9.9	Structure and encoding of PN-parameters	240
9.10	Structure and encoding of DD-parameters	242
10	Type 1: DL-service elements of procedure	243
10.1	Operation of the DL(SAP)-address, buffer and queue management services	243
10.2	Operation of the connection-mode services	247
10.3	Operation of the connectionless-mode services.....	288
10.4	Operation of the scheduling guidance services	300

11	Type 1: DL-support subprotocol	310
11.1	Scope	310
11.2	Overview of LAS operation	310
11.3	DL-support subprotocol definition	311
11.4	Elements of Procedures for receiving SPDUs	345
12	Type 1: Other DLE elements of procedure.....	347
12.1	DLE initialization	347
12.2	LAS behavior and operation	351
12.3	DL-support operation.....	358
12.4	DL-bridge elements of procedure and bridge sub-protocol.....	363
12.5	DL-management-information	393
12.6	Implementation profiles	398
13	Type 1: PICS proforma.....	404
13.1	Introduction	404
13.2	Scope.....	404
13.3	Normative references	404
13.4	Definitions	404
13.5	Abbreviations	404
13.6	Conformance	405
13.7	Instructions	405
13.8	Identification.....	406
13.9	Implementation profile	407
13.10	Major low-level capabilities.....	411
13.11	Major high-level capabilities	425
14	Type 2: Overview of the DL-protocol	434
14.1	General	434
14.2	Services provided by the DL	436
14.3	Structure and definition of DL-addresses	437
14.4	Services assumed from the PhL	439
14.5	Functional classes.....	441
15	Type 2: General structure and encoding of PhIDUs and DLPDUs and related elements of procedure	442
15.1	Overview	442
15.2	Media access procedure.....	442
15.3	DLPDU structure and encoding	445
15.4	Lpacket components.....	447
15.5	DLPDU procedures.....	449
15.6	Summary of DLL support services and objects	450
16	Type 2: Specific DLPDU structure, encoding and procedures.....	453
16.1	Modeling language	453
16.2	DLS user services	455
16.3	Generic Tag Lpacket	461
16.4	Moderator Lpacket	461
16.5	Time distribution Lpacket.....	463
16.6	UCMM Lpacket.....	465
16.7	Keeper UCMM Lpacket.....	465
16.8	TUI Lpacket.....	466

16.9	Link parameters Lpacket and tMinus Lpacket	467
16.10	I'm alive Lpacket	469
16.11	Ping Lpackets.....	470
16.12	WAMI Lpacket	472
16.13	Debug Lpacket	472
17	Type 2: Objects for station management	474
17.1	General	474
17.2	ControlNet object	475
17.3	Keeper object.....	485
17.4	Scheduling object.....	508
17.5	TCP/IP interface object	517
17.6	Ethernet link object.....	524
18	Type 2: Other DLE elements of procedure	529
18.1	Network Attachment Monitor (NAM).....	529
18.2	Calculating link parameters	536
19	Type 2: Detailed specification of DL components	544
19.1	General	544
19.2	Access Control Machine (ACM)	544
19.3	TxLLC.....	561
19.4	RxLLC.....	565
19.5	Transmit Machine (TxM).....	568
19.6	Receive Machine (RxM).....	571
19.7	Serializer	577
19.8	Deserializer	578
19.9	DLL management	579
20	Type 3: Overview of the DL-protocol	582
20.1	General	582
20.2	Overview of the medium access control and transmission protocol	582
20.3	Transmission modes and DL-entity.....	583
20.4	Service assumed from the PhL	588
20.5	Operational elements	590
20.6	Cycle and system reaction times	605
21	Type 3: General structure and encoding of DLPDUs, and related elements of procedure	609
21.1	DLPDU granularity.....	609
21.2	Length octet (LE, LEr)	610
21.3	Address octet	610
21.4	Control octet (FC).....	613
21.5	DLPDU content error detection	616
21.6	DATA_UNIT	617
21.7	Error control procedures.....	618
22	Type 3: DLPDU-specific structure, encoding and elements of procedure	619
22.1	DLPDUs of fixed length with no data field	619
22.2	DLPDUs of fixed length with data field	621
22.3	DLPDUs with variable data field length	622
22.4	Token DLPDU	624

22.5	ASP DLPDU	625
22.6	SYNCH DLPDU	625
22.7	Time Event (TE) DLPDU	625
22.8	Clock Value (CV) DLPDU	625
22.9	Transmission procedures	626
23	Type 3: Other DLE elements of procedure	629
23.1	DL-entity initialization	629
23.2	States of the media access control of the DL-entity	629
23.3	Clock synchronization protocol	635
24	Type 4: Data Link Protocol Definition	640
24.1	Overview of the DL-protocol	640
24.2	General structure and encoding of PhIDUs and DLPDUs, and related elements of procedure	651
24.3	DLPDU-specific structure, encoding and elements of procedure	657
24.4	DL-service elements of procedure	661
24.5	Route mechanism	664
24.6	Link-access system	668
24.7	Local variables, counters and queues	668
25	Type 5: <placeholder for the Type 5 DL-protocol definition>	671
26	Type 6: Connection-oriented DL-protocol and related matters	673
26.1	Connection mode data transfer using TDMA principles	673
26.2	Bus access and data transmission	673
26.3	Overview of bus synchronization	675
26.4	Connection mode data transfer	678
26.5	DLM-connectionless service	706
26.6	Other real-time services	720
26.7	Bridge architecture and operation	724
26.8	Bus configuration	726
26.9	Conformance	736
27	Type 6: MAC elements of procedure	738
27.1	Procedures related to bus startup and bus operation	738
27.2	Medium access control (MAC)	743
28	Type 7: Overview of the DL-protocol	746
28.1	Overall description of medium allocation	746
28.2	Types of entities	748
28.3	Addressing	751
28.4	Flow control	757
28.5	Graphical representation	759
29	Type 7: General structure and encoding of PhIDUs and DLPDUs and related elements of procedure	760
29.1	DLPDU formats and components	760
29.2	Description of each DLPDU component	760
29.3	PhIDU structure and encoding	762
29.4	Common DLPDU structure, encoding and elements of procedure	763
29.5	Valid DLPDU types	763
29.6	DLL timers	765

30	Type 7: DLPDU-specific structure, encoding and element of procedure.....	771
30.1	General	771
30.2	Buffer read	771
30.3	Buffer write.....	771
30.4	Buffer transfer	771
30.5	Specified explicit request.....	772
30.6	Free explicit request.....	778
30.7	Messaging.....	781
30.8	Acknowledged messaging	786
30.9	Numbering of acknowledged messages	790
30.10	Behavior with mismatched parameters	792
31	Type 7: DL-service elements of procedure, interfaces and conformance	795
31.1	General	795
31.2	Producer/consumer entity	795
31.3	Protocol elements by service.....	798
31.4	Bus arbitrator operation	806
31.5	Bridges.....	813
31.6	Interfaces	820
31.7	Conformance	822
32	Type 8: DL-protocol	825
32.1	Overview	825
32.2	DL-service Interface (DLI)	825
32.3	Peripherals data link (PDL).....	829
32.4	Basic Link Layer (BLL)	864
32.5	Medium Access Control (MAC).....	880
32.6	Peripherals network management for layer 2 (PNM2)	912
32.7	Parameters and monitoring times of the DLL	920
Annex A	(informative) – Types 1, 2, 3 (synchronous), 5 (first FCS), 6, 7 and 8: Exemplary FCS implementations.....	926
Annex B	(informative) – Type 1: Formal protocol finite state machines.....	928
B.1	Basic reception and transmission FSMs	928
B.2	FSMs for DLCs.....	940
B.3	FSMs for scheduling.....	946
B.4	FSMs for bridges	946
Annex C	(informative) – Type 1: DLPDU and DL-addressing short-form summaries	947
C.1	Fields used in short-form summaries	947
C.2	DLPDU short-form summary grouped by function	949
C.3	DLPDU short-form summary in alphabetic order of DLPDU names	951
C.4	DLPDU short-form summary in alphabetic order of DLPDU acronyms.....	952
C.5	DLPDU FC code-point assignment matrix – overview and detail	953
C.6	SD-parameters (status and data-description parameters) of CA, CD, ED and DT DLPDUs.....	956
C.7	EC parameters of EC DLPDUs	959
C.8	Parameters of DC and RC DLPDUs.....	961
C.9	Parameters of TD, RQ and RR DLPDUs	963
C.10	Parameters of PN, PT, ES and RI DLPDUs	965
C.11	Addressing summary extracted from figures and tables of 6.3	967

Annex D (informative) – Type 2: Indicators and Switches	971
D.1 Purpose.....	971
D.2 General indicator requirements.....	971
D.3 Common indicator requirements	971
D.4 Module status indicator	972
D.5 Network status indicators	972
D.6 Switches.....	975
Annex E (normative) – Type 3: DL-Protocol state machines	977
E.1 Overall structure.....	977
E.2 Variation of state machines in different devices	979
E.3 DL Data Resource	979
E.4 FLC / DLM.....	984
E.5 MAC	1009
E.6 SRU	1038
Annex F (informative) – Type 3: Exemplary token procedure and message transfer periods.....	1056
F.1 Procedure of token passing	1056
F.2 Examples for token passing procedure	1057
F.3 Examples for message transfer periods – asynchronous transmission	1061
F.4 Examples for message transfer periods – synchronous transmission	1062
Annex G (informative) – Type 7: Object Modeling	1063
G.1 Modeling of the IDENTIFIER object	1063
G.2 Description of the IDENTIFIER object attributes	1064
G.3 Modeling of the QUEUE object	1068
G.4 Description of the QUEUE object attributes	1068
G.5 Modeling of the BUFFER object.....	1069
G.6 Description of the BUFFER object attributes.....	1069
Annex H (informative) – Type 7: Topology of multi-segment DL-subnetwork	1070
H.1 Introduction	1070
H.2 Global specification	1070
H.3 Local specification.....	1071
H.4 Properties.....	1071
H.5 Methods	1072
Annex I (informative) – Type 7: Management of transmission errors	1075
I.1 Transmission of RP_DAT_XX.....	1075
I.2 Transmission of a free RP_RQ(1/2).....	1075
I.3 Transmission of the specified RP_RQ1.....	1076
I.4 Transmission of RP_MSG_NOACK.....	1077
I.5 Transmission of RP_MSG_ACK.....	1079
Annex J (informative) – Type 8: Implementation possibilities of definite PNM2 functions	1082
J.1 Acquiring the current configuration	1082
J.2 Comparing the acquired and stored configurations prior to a DL-subnetwork error	1085
BIBLIOGRAPHY	1092
INDEX	1094

Figure 1 – Relationship of IEC 61158-4 to other fieldbus layers and to users of the fieldbus Data Link service	27
Figure 2 – Relationships of DLSAPs, DLSAP-addresses and group DL-addresses	38
Figure 3 – Relationships of DLCEPs and DLCEP-addresses to DLSAPs, DLSAP-addresses and group DL-addresses	40
Figure 4 – Basic Structure of a DL-address	98
Figure 5 – Basic Structure of a Sublink Selector	98
Figure 6 – DL-address Alternative Structures	99
Figure 7 – Basic Structure of MAC-addresses	109
Figure 8 – Representation of a DL-address as a MAC-address	109
Figure 9 – Linear relationships of sending and receiving DLCEP sequence-number variables	122
Figure 10 – DL-address alternative structures	133
Figure 11 – SHORT DL-address field – alternative implicit structures	133
Figure 12 – NODE DL-address field – implicit structure	134
Figure 13 – State transition diagram for a DLCEP	247
Figure 14 – Projection of the sending and receiving DLCEP sequence-number variables of Figure 9 onto the cyclic sequence-number parameters of CA, CD, DT, ED and RC DLPDUs, with consequent determination of required actions	264
Figure 15 – State transitions of a DLE	347
Figure 16 – Bridged network topology	364
Figure 17 – Spanning tree representation	365
Figure 18 – DLSDU transit delay, DLPDU lifetime and bridge forwarding delay	369
Figure 19 – Forwarding and delivering a received DLPDU	373
Figure 20 – Forwarding a locally-originated DLPDU	374
Figure 21 – Republishing a DLSDU received from another link	375
Figure 22 – Bridge architecture	376
Figure 23 – Replacement for [IL] Fig 3-2 Bridge ports	385
Figure 24 – Replacement for [IL] Fig 3-3 Bridge architecture	386
Figure 25 – Data Link Layer internal architecture	435
Figure 26 – Basic Structure of a MAC ID address	437
Figure 27 – Basic Structure of a Generic tag address	437
Figure 28 – Basic Structure of a Fixed tag address	438
Figure 29 – M_symbols and Manchester encoding at 5 MHz (informative)	439
Figure 30 – NUT structure	443
Figure 31 – Media access during scheduled time	443
Figure 32 – Media access during unscheduled time	444
Figure 33 – DLPDU format	445
Figure 34 – Aborting a DLPDU during transmission	446
Figure 35 – Lpacket format	447
Figure 36 – Generic tag Lpacket format	448
Figure 37 – Fixed tag Lpacket format	448
Figure 38 – goodness parameter of TimeDist_Lpacket	463
Figure 39 – Example I'm alive processing algorithm	470
Figure 40 – Keeper CRC algorithm	491
Figure 41 – Keeper object power-up state diagram	503

Figure 42 – Keeper object operating state diagram	504
Figure 43 – Synchronized network change processing	507
Figure 44 – State transition diagram for TCP/IP interface object	524
Figure 45 – NAM state machine	530
Figure 46 – Logical Token Passing ring	585
Figure 47 – PhL data service for asynchronous transmission	589
Figure 48 – Idle time T_{ID1}	592
Figure 49 – Idle time T_{ID2} (SDN, CS)	592
Figure 50 – Idle time T_{ID2} (MSRD)	593
Figure 51 – Slot time T_{SL1}	593
Figure 52 – Slot time T_{SL2}	594
Figure 53 – Slot time T_{SL1}	599
Figure 54 – Slot time T_{SL2}	599
Figure 55 – Token transfer period	605
Figure 56 – Message transfer period.....	606
Figure 57 – UART character	609
Figure 58 – Octet structure	610
Figure 59 – Length octet coding.....	610
Figure 60 – Address octet coding.....	610
Figure 61 – DAE/SAE octet in the DLPDU.....	611
Figure 62 – Address extension octet.....	612
Figure 63 – FC octet coding for send/request DLPDUs	613
Figure 64 – FC octet coding for acknowledgement or response DLPDUs	613
Figure 65 – FCS octet coding.....	616
Figure 66 – Data field	617
Figure 67 – Ident user data.....	617
Figure 68 – DLPDUs of fixed length with no data field.....	619
Figure 69 – DLPDUs of fixed length with no data field.....	620
Figure 70 – DLPDUs of fixed length with data field	621
Figure 71 – DLPDUs of fixed length with data field	622
Figure 72 – DLPDUs with variable data field length.....	623
Figure 73 – DLPDUs with variable data field length.....	624
Figure 74 – Token DLPDU	624
Figure 75 – Token DLPDU	625
Figure 76 – Send/request DLPDU of fixed length with no data	626
Figure 77 – Token DLPDU and send/request DLPDU of fixed length with data.....	626
Figure 78 – Send/request DLPDU with variable data field length.....	627
Figure 79 – Send/request DLPDU of fixed length with no data	627
Figure 80 – Token DLPDU and send/request DLPDU of fixed length with data.....	628
Figure 81 – Send/request DLPDU with variable data field length.....	628
Figure 82 – DL-state-diagram	630
Figure 83 – Overview of clock synchronization.....	636
Figure 84 – Time master state machine	637
Figure 85 – Time receiver state machine	638
Figure 86 – Clock synchronization	639

Figure 87 – Relationship of PhE, DLE and DLS-user.....	640
Figure 88 – DLE state diagram for confirmed and unconfirmed, unacknowledged DLPDUs .	642
Figure 89 – DLE state diagram for confirmed acknowledged DLPDUs.....	643
Figure 90 – DLE state diagram for unconfirmed acknowledged DLPDUs.....	644
Figure 91 – Full duplex DLE receive state diagram	645
Figure 92 – Full duplex DLE transmit state diagram	645
Figure 93 – Link access example	648
Figure 94 – Simple P-NET-route format	654
Figure 95 – Extended P-NET-route format	654
Figure 96 – Complex P-NET-route format	655
Figure 97 – Immediate P-NET-route format.....	655
Figure 98 – Control-status format.....	656
Figure 99 – Data-field-format	657
Figure 100 – Source / destination designator	665
Figure 101 – Simple P-NET-route generation.....	666
Figure 102 – Extended P-NET-route generation.....	666
Figure 103 – Complex P-NET-route generation.....	667
Figure 104 – Simple DL-route generation.....	667
Figure 105 – Extended DL-route generation.....	667
Figure 106 – Complex DL-route generation.....	668
Figure 107 – Connection mode operation using TDMA.....	673
Figure 108 – Fundamental access concepts	674
Figure 109 – EXAMPLE: Architecture of a synchronizer showing one upper port bridge-DLE synchronizing multiple lower port bridge-DLEs	676
Figure 110 – EXAMPLE: Architecture of a forwarding bridge showing one upper port bridge-DLE synchronizing exactly one lower port bridge-DLE	676
Figure 111 – EXAMPLE: Lower-port bridge DLE	677
Figure 112 – EXAMPLE: Bridged network using forwarding bridges	678
Figure 113 – Operation of the SCAN channel-class and its associated primitives.....	679
Figure 114 – Operation of the ExSCAN channel-class and its associated primitives.....	681
Figure 115 – Operation of the GPA channel class and its associated primitives	683
Figure 116 – Data Transfer on the GPC channel-class and its associated primitives	687
Figure 117 – Transfer DLPDU composition on a UNITARY channel.....	691
Figure 118 – Initial-grain DLPDU composition.....	693
Figure 119 – Initial-grain header composition.....	694
Figure 120 – Middle-grain DLPDU composition	695
Figure 121 – Middle-grain header composition.....	696
Figure 122 – Final-grain DLPDU composition.....	696
Figure 123 – Final-grain header composition	697
Figure 124 – DL-status-header and DL-status DLPDU	698
Figure 125 – DLPDU transmission format for 1 WAY or 2 WAY transmission method.....	704
Figure 126 – EXAMPLE: Three level real network with two superimposed virtual networks .	705
Figure 127 – Operation of DLM-connectionless service and its primitives	706
Figure 128 – DLM-connectionless DLPDU composition and encoding	708
Figure 129 – Node DLMSDU composition and encoding	712
Figure 130 – Node-check DLMSDU composition and encoding	715

Figure 131 – Configuration-status parameter composition.....	716
Figure 132 – DLM-connectionless DL-addresses, symbols and their relationships	717
Figure 133 – General form and encoding of DLM-connectionless DL-addresses	717
Figure 134 – Architecture of a forwarding bridge.....	724
Figure 135 – EXAMPLE: Forwarding through a bridge by slot matching	724
Figure 136 – Channel-mode composition and encoding	730
Figure 137 – Half-forwarding-port mode composition and encoding	732
Figure 138 – Configuration-status field composition	733
Figure 139 – Bus-config field composition and encoding	739
Figure 140 – General description of medium allocation	747
Figure 141 – Internal structure of a producer/consumer entity.....	748
Figure 142 – Associated buffers and queues	750
Figure 143 – Internal structure of a bus arbitrator	751
Figure 144 – Polling BA Table	751
Figure 145 – Addressing scheme	752
Figure 146 – Address partitioning	754
Figure 147 – Structure of an individual physical address.....	755
Figure 148 – Structure of an individual logical address	755
Figure 149 – Structure of restricted physical group address.....	755
Figure 150 – Structure of restricted logical group address	756
Figure 151 – Structure of a generalized group address	756
Figure 152 – Summary of address structure.....	757
Figure 153 – Example of an evaluation net	759
Figure 154 – Basic DLPDU structure.....	760
Figure 155 – DLPDU transmission / reception order.....	760
Figure 156 – Identifier DLPDU	764
Figure 157 – Variable response DLPDU.....	764
Figure 158 – Request response DLPDU.....	764
Figure 159 – Message response DLPDU.....	765
Figure 160 – Acknowledgement response DLPDU	765
Figure 161 – End of message transaction response DLPDU	765
Figure 162 – Buffer reading service interactions	771
Figure 163 – Buffer writing service interactions.....	771
Figure 164 – Buffer transfer service interactions	772
Figure 165 – Buffer transfer DLPDU sequence.....	772
Figure 166 – Interactions within the <u>specified explicit request</u> for buffer transfer service in the <u>aperiodic window</u>	774
Figure 167 – Interactions within the <u>specified explicit request</u> for buffer transfer service in the <u>periodic window</u>	775
Figure 168 – DLPDU sequence for an explicit request for a transfer	776
Figure 169 – Evaluation network for a buffer transfer specified explicit request with (RQ_INHIBITED = False).....	777
Figure 170 – Evaluation network for a buffer transfer specified explicit request with (RQ_INHIBITED = True)	777
Figure 171 – Diagram of interactions within the <u>free explicit request</u> for buffer transfer service.....	779
Figure 172 – Evaluation network for a free explicit request	780

Figure 173 – Diagram of interactions within the unacknowledged message transfer request service for an aperiodic transfer	783
Figure 174 – Diagram of interactions within the unacknowledged message transfer request service for a cyclical transfer	784
Figure 175 – DLPDU sequence for an aperiodic message transfer	785
Figure 176 – DLPDU sequence for a cyclical message transfer	786
Figure 177 – Diagram of interactions within the acknowledged message transfer request service for an aperiodic transfer	787
Figure 178 – Diagram of interactions within the acknowledged message transfer request service for a cyclical transfer	788
Figure 179 – DLPDU sequence for an aperiodic message transfer	789
Figure 180 – DLPDU sequence for a cyclical message transfer	790
Figure 181 – Evaluation network for message aperiodic transfer	793
Figure 182 – Evaluation network for message cyclic transfer	794
Figure 183 – Simplified states machine for a producer/consumer entity	795
Figure 184 – Active bus arbitrator's simplified state machine	811
Figure 185 – Typical bridge usage	813
Figure 186 – Architectural placement of bridges in OSI Basic Reference Model (ISO/IEC 7498)	813
Figure 187 – Representation of an extended link communication	814
Figure 188 – Evaluation network for reception of an RP_MSG_ACK DLPDU	819
Figure 189 – Evaluation network for reception of an RP_MSG_NOACK DLPDU	820
Figure 190 – Data Link Layer Entity	825
Figure 191 – Location of the DLI in the DLL	825
Figure 192 – State transition diagram of DLI	827
Figure 193 – Location of the PDL in the DLL	829
Figure 194 – PDL connection between slave and master	830
Figure 195 – Interface between PDL-user (DLI) and PDL in the layer model	830
Figure 196 – Overview of the PDL services	831
Figure 197 – PDL_Data_Ack service between master and only one slave	832
Figure 198 – Parallel processing of PDL_Data_Ack services	833
Figure 199 – PSM and GSM service for buffer access	833
Figure 200 – Buffer_Received service to indicate successful data transfer	833
Figure 201 – Data flow between PDL-user, PDL and BLL of a PDL_Data_Ack service	836
Figure 202 – Interface between PDL and PNM2 in the layer model	837
Figure 203 – Reset, Set Value and Get Value PDL services	838
Figure 204 – Event PDL service	838
Figure 205 – Transmit and receive FCBs on the master and slave sides	842
Figure 206 – Data transmission master → slave with SWA Message	842
Figure 207 – Time sequence of the data transmission master → slave with SWA Message	843
Figure 208 – Data transmission slave → master with SWA/RWA Message	843
Figure 209 – Time sequence of the data transmission slave → master with SWA/RWA Message	844
Figure 210 – Allocation of actions of the PDL protocol machines and data cycles	845
Figure 211 – Message transmission: master → slave	845
Figure 212 – Message transmission: slave → master	845

Figure 213 – Code octet of a PDL PDU	846
Figure 214 – Structure of a message with a size of one word	847
Figure 215 – Structure of a SPA Message	848
Figure 216 – Structure of a SVA Message	848
Figure 217 – Structure of a FCB_SET Message	848
Figure 218 – Structure of a RWA Message	849
Figure 219 – Structure of a SWA Message	849
Figure 220 – Structure of a confirmation for SPA or SVA Messages	849
Figure 221 – Structure of a FCB_SET as confirmation	849
Figure 222 – Structure of the data octet for FCB_SET as requests and confirmations	850
Figure 223 – Structure of a message with a size of more than one word	850
Figure 224 – PDL base protocol machine	852
Figure 225 – Locations of the PDL and the PDL protocol machines in the master and slaves	854
Figure 226 – PDL protocol machine	855
Figure 227 – TRANSMIT protocol machine	858
Figure 228 – RECEIVE protocol machine	861
Figure 229 – Location of the BLL in the DLL	864
Figure 230 – Interface between PDL and BLL in the layer model	865
Figure 231 – BLL_Data service	866
Figure 232 – Interface between PNM2 and BLL in the layer model	868
Figure 233 – Reset, Set Value and Get Value BLL services	869
Figure 234 – Event BLL service	870
Figure 235 – BLL operating protocol machine of the master	874
Figure 236 – BLL-BAC protocol machine	876
Figure 237 – BLL operating protocol machine of the slave	879
Figure 238 – Location of the MAC in the DLL	880
Figure 239 – Model details of layers 1 and 2	880
Figure 240 – DLPDU cycle of a data sequence without errors	881
Figure 241 – DLPDU cycle of a data sequence with errors	881
Figure 242 – Data sequence DLPDU transmitted by the master	882
Figure 243 – Data sequence DLPDU received by the master	882
Figure 244 – Check sequence DLPDU	882
Figure 245 – Loopback word (LBW)	883
Figure 246 – Checksum status generated by the master	883
Figure 247 – Checksum status received by the master	883
Figure 248 – MAC protocol machine of a master: transmission of a message	885
Figure 249 – MAC protocol machine of a master: receipt of a message	888
Figure 250 – MAC sublayer of a master: data sequence identification	892
Figure 251 – Data sequence DLPDU received by a slave	895
Figure 252 – Data sequence DLPDU transmitted by a slave	895
Figure 253 – Checksum status received by the slave	895
Figure 254 – Checksum status generated by the slave	896
Figure 255 – State transitions of the MAC sublayer of a slave: data sequence	897
Figure 256 – State transitions of the MAC sublayer of a slave: check sequence	898
Figure 257 – Interface between MAC-user and MAC in the layer model	903

Figure 258 – Interactions at the MAC-user interface (master)	904
Figure 259 – Interactions at the MAC-user interface (slave)	905
Figure 260 – Interface between MAC and PNM2 in the layer model	908
Figure 261 – Reset, Set Value and Get Value MAC services	910
Figure 262 – Event MAC service	910
Figure 263 – Location of the PNM2 in the DLL	912
Figure 264 – Interface between PNM2-user and PNM2 in the layer model	913
Figure 265 – Reset, Set Value, Get Value and Get Active Configuration services	915
Figure 266 – Event PNM2 service	915
Figure 267 – Set Active Configuration, Get Current Configuration service	915
Figure 268 – The active_configuration parameter	919
Figure 269 – Device code structure	922
Figure 270 – Relations between data width, process data channel and parameter channel	924
Figure 271 – Structure of the control code	925
Figure A.1 – Example of FCS generation for Types 1, 3 (synchronous), 5 (first FCS), 6 and 7	926
Figure A.2 – Example of FCS syndrome checking on reception for Types 1, 3 (synchronous), 5 (first FCS), 6 and 7	926
Figure C.1 – Gross structure of FC code points	953
Figure C.2 [Figure 4] – Basic structure of a DL-address	967
Figure C.3 [Figure 5] – Basic structure of a sublink selector	967
Figure C.4 [Figure 6] – DL-address alternative structures	967
Figure C.5 [Figure 7] – Basic structure of MAC-addresses	967
Figure C.6 [Figure 8] – Representation of a DL-address as a MAC-address	967
Figure D.1 – Non redundant network status indicator labeling	975
Figure D.2 – Redundant network status indicator labeling	975
Figure E.1 – Structuring of the protocol machines	978
Figure E.2 – Structure of the SRU Machine	1039
Figure F.1 – Derivation of the token holding time (T_{TH})	1057
Figure F.2 – No usage of token holding time (T_{TH})	1058
Figure F.3 – Usage of token holding time (T_{TH}) for message transfer (equivalence between T_{TH} of each Master station)	1059
Figure F.4 – Usage of token holding time (T_{TH}) in different working load situations	1060
Figure I.1 – Evaluation DL-subnetwork for transmission of RP_DAT_XX	1075
Figure I.2 – Evaluation DL-subnetwork for transmission of a free RP_RQ(1/2)	1076
Figure I.3 – Evaluation DL-subnetwork for transmission of the specified RP_RQ1	1077
Figure I.4 – Evaluation DL-subnetwork for transmission of RP_MSG_NOACK, first behavior	1078
Figure I.5 – Evaluation DL-subnetwork for transmission of RP_MSG_NOACK, second behavior	1079
Figure I.6 – Evaluation DL-subnetwork for transmission of RP_MSG_ACK, first behavior	1080
Figure I.7 – Evaluation DL-subnetwork for transmission of RP_MSG_ACK, second behavior	1081
Figure J.1 – DL-subnetwork configuration in the form of a tree structure	1082
Figure J.2 – State machine for the acquisition of the current configuration	1084

Figure J.3 – State machine for comparing two configurations 1088

Figure J.4 – State machine for comparing one line of two configuration matrices 1090

Table 1 – FCS length, polynomials and constants by protocol Type	85
Table 2 – Link node selector addressing	101
Table 3 – Link-local node selector addressing.....	103
Table 4 – Link-local node designators.....	104
Table 5 – Node-local selector addressing	106
Table 6 – Predefined flat non-local DL-addresses	107
Table 7 – Predefined flat link-local DL-addresses	108
Table 8 – Predefined node-local DL-addresses	108
Table 9 – Correlation of DLPDUs with functional classes	114
Table 10 – Summary structure of DLPDUs	140
Table 11 – DLPDU restrictions based on dominant token.....	141
Table 12 – Structure of EC DLPDUs	141
Table 13 – Structure of DC DLPDUs	144
Table 14 – Structure of RC DLPDUs	146
Table 15 – Structure of CA DLPDUs	148
Table 16 – Structure of CD DLPDUs	154
Table 17 – Structure of ED DLPDUs	162
Table 18 – Structure of DT DLPDUs	171
Table 19 – Structure of SR DLPDUs	178
Table 20 – Structure of CT DLPDUs	181
Table 21 – Structure of TD DLPDUs	182
Table 22 – Structure of RQ DLPDUs.....	184
Table 23 – Structure of RR DLPDUs	187
Table 24 – Structure of PN DLPDUs	188
Table 25 – Structure of PR DLPDUs	191
Table 26 – Structure of PT DLPDUs.....	193
Table 27 – Structure of ES DLPDUs	202
Table 28 – Structure of RT DLPDUs	208
Table 29 – Structure of RI DLPDUs.....	209
Table 30 – Structure of CL DLPDUs.....	211
Table 31 – Structure of TL DLPDUs	212
Table 32 – Structure of WK DLPDUs.....	216
Table 33 – Structure of IDLE DLPDUs.....	217
Table 34 – Assumed structure of undefined (spare) DLPDUs	219
Table 35 – Assumed structure of RESERVED (NOT TO BE USED) DLPDUs	220
Table 36 – Structure of an EC DLPDU's parameters	221
Table 37 – EC-parameters: 1st octet.....	221
Table 38 – EC-parameters: 2nd octet.....	222
Table 39 – EC-parameters: 3rd and 4th octets	222
Table 40 – EC-parameters: 5th and 6th octets	223
Table 41 – EC-parameters: 7th octet	223
Table 42 – EC-parameters: 8th octet	223
Table 43 – EC-parameters: 9th and 10th octets	224

Table 44 – EC-parameters: 11th octet.....	224
Table 45 – EC-parameters: 12th octet.....	225
Table 46 – EC-parameters: 13th and 14th octets	225
Table 47 – DC-parameters and RC-parameters: 1st octet	226
Table 48 – DC-parameters and RC-parameters: 2nd octet	226
Table 49 – Disconnect reasons	227
Table 50 – Reset reasons	228
Table 51 – RC-parameters: 3rd octet	228
Table 52 – RC-parameters: 4th octet	228
Table 53 – Structure of connectionless-mode CA, CD, DT and ED DLPDUs.....	229
Table 54 – Short format SD-parameters for connectionless transaction initiators	230
Table 55 – Short format SD-parameters for connectionless responders	230
Table 56 – Reply status for unitdata-acknowledgment and exchange-unitdata-reply DT DLPDUs.....	231
Table 57 – Structure of connection-oriented CA, CD, DT and ED DLPDUs.....	233
Table 58 – Short format SD-parameters for DLCEP state.....	234
Table 59 – Long format SD-parameters for DLCEP state: 1st octet	234
Table 60 – Long format SD-parameters for DLCEP state: 2nd octet.....	234
Table 61 – Long format SD-parameters for DLCEP state: 3rd octet.....	235
Table 62 – Reply status for SR DLPDUs	236
Table 63 – Short format SR-parameters.....	236
Table 64 – Structure of TD-parameters	237
Table 65 – Structure and encoding of the DL-time-quality measures	237
Table 66 – Approximate numeric significance of the bits of seven-octet DL-time	238
Table 67 – Approximate numeric significance of the bits of three-octet short time	239
Table 68 – Structure of RQ-parameters	239
Table 69 – Structure of RR-parameters.....	240
Table 70 – Structure and encoding of the RR-time-quality measures	240
Table 71 – Structure of PN-parameters.....	241
Table 72 – PN-parameters: 1st octet.....	241
Table 73 – PN-parameters: 2nd octet.....	241
Table 74 – PN-parameters: 3rd and 4th octets	242
Table 75 – PN-parameters: 5th octet	242
Table 76 – PN-parameters: 6th octet	242
Table 77 – Structure of DD-parameters.....	242
Table 78 – Components of returned DL-time.....	300
Table 79 – Time synchronization computation.....	302
Table 80 – SPDU 1st octet: SPDU class, and protocol version or subclass	311
Table 81 – Probe-response SPDU	312
Table 82 – DL-protocol versions supported.....	312
Table 83 – PR-SPDU: 3rd and 4th octets	313
Table 84 – Node-activation SPDU.....	314
Table 85 – Node-activation SPDU: 4th octet	314
Table 86 – LAS-data-base-status SPDU	315
Table 87 – LAS-data-base-status SPDU: 2nd octet	315
Table 88 – Live-list- change SPDU	316

Table 89 – DLE-status structure.....	316
Table 90 – Live-list-detail SPDU	317
Table 91 – DL-conformance-reply SPDU.....	318
Table 92 – DL-protocol versions supported.....	318
Table 93 – DL-conformance encoding (portion 1).....	319
Table 94 – DL-conformance encoding (portion 2).....	319
Table 95 – DL-conformance encoding (portion 3).....	319
Table 96 – DL-conformance encoding (portion 4).....	320
Table 97 – Link-basic-parameters-reply SPDU.....	320
Table 98 – Link-master-parameters-reply SPDU	321
Table 99 – Token-hold-time-request SPDU	322
Table 100 – Token-hold-time-array SPDU.....	323
Table 101 – Sequence element header encoding.....	325
Table 102 – SHORT DL-address and duration sequence element	325
Table 103 – LONG DL-address and duration sequence element	326
Table 104 – Wakeup request sequence element	326
Table 105 – Schedule-request SPDU	327
Table 106 – Sequence type, schedule type and priority encoding	327
Table 107 – Scheduling-completed SPDU.....	329
Table 108 – Status and reason codes	329
Table 109 – Cancel-schedule SPDU	330
Table 110 – Schedule-cancelled SPDU.....	330
Table 111 – Link-schedule	331
Table 112 – Schedule-summary SPDU	332
Table 113 – Subschedule-SPDU reference	333
Table 114 – Subschedule SPDU	333
Table 115 – Sequence Sub-SPDU	334
Table 116 – Element-description.....	335
Table 117 – Schedule-summary-request SPDU.....	336
Table 118 – Subschedule-request SPDU	336
Table 119 – Parameter-list element-header encoding	337
Table 120 – Begin/end-of-list element.....	337
Table 121 – Continuation-of-list element.....	338
Table 122 – SHORT DL-address list element	338
Table 123 – LONG DL-address element	339
Table 124 – DLSAP-address-characteristics element.....	339
Table 125 – DLCEP-characteristics element	340
Table 126 – Address-query SPDU	340
Table 127 – Address-report SPDU	341
Table 128 – Address-list-query SPDU	342
Table 129 – DL-address selection criteria	343
Table 130 – Address-list-reply SPDU	344
Table 131 – Topology change notification BPDU format.....	393
Table 132 – Configuration BPDU format	393
Table 133 – Maximum permitted phase-tracking error in a DLE's sense of DL-time at the minimum requireable Time Distribution period	402

Table 134 – Data Link Layer components	434
Table 135 – MAC ID addresses allocation.....	437
Table 136 – Fixed tag service definitions	438
Table 137 – Data encoding rules	439
Table 138 – M Data symbols.....	440
Table 139 – Truth table for ph_status_indication.....	441
Table 140 – DLL support services and objects	451
Table 141 – Elementary data types	454
Table 142 – DLL events	459
Table 143 – Time distribution priority	464
Table 144 – Format of the TUI Lpacket.....	467
Table 145 – ControlNet object class attributes	475
Table 146 – ControlNet object instance attributes	475
Table 147 – TUI status flag bits	479
Table 148 – Channel state bits	481
Table 149 – ControlNet object common services.....	482
Table 150 – ControlNet object class specific services	483
Table 151 – Keeper Object revision history.....	485
Table 152 – Keeper object class attributes.....	485
Table 153 – Keeper object instance attributes.....	486
Table 154 – Keeper operating state definitions	489
Table 155 – Port status flag bit definitions	489
Table 156 – TUI status flag bits	490
Table 157 – Keeper attributes.....	492
Table 158 – Memory requirements (in octets) for the Keeper attributes.....	493
Table 159 – Keeper object common services	493
Table 160 – Keeper class specific services.....	494
Table 161 – Service error codes	495
Table 162 – Wire order format of the TUI Lpacket.....	499
Table 163 – Service error codes	500
Table 164 – Keeper object operating states	501
Table 165 – Keeper object state event matrix	505
Table 166 – Scheduling object class attributes	508
Table 167 – Scheduling object instance attributes	509
Table 168 – Services at class level and instance level.....	509
Table 169 – Status error descriptions for Create.....	510
Table 170 – Status error descriptions for Delete and Kick_Timer	511
Table 171 – Status error descriptions for Read	512
Table 172 – Status error descriptions for Conditional_Write.....	513
Table 173 – Status error descriptions for Forced_Write.....	514
Table 174 – Status error descriptions for Change_Start.....	514
Table 175 – Status error descriptions for Break_Connections	515
Table 176 – Status error descriptions for Change_Complete.....	516
Table 177 – Status error descriptions for Restart_Connections	516
Table 178 – TCP/IP interface object class attributes	518
Table 179 – TCP/IP interface object instance attributes	518

Table 180 – Status values.....	520
Table 181 – Configuration capability bits.....	520
Table 182 – Configuration control bits.....	520
Table 183 – Example path	521
Table 184 – Interface configuration components	522
Table 185 – TCP/IP interface object common services.....	523
Table 186 – Get_Attribute_All reply format.....	523
Table 187 – Ethernet link object class attributes	525
Table 188 – Ethernet link object instance attributes	525
Table 189 – Interface flags bits	527
Table 190 – Ethernet link object common services.....	528
Table 191 – Ethernet link object class specific services	528
Table 192 – NAM states.....	529
Table 193 – Default link parameters.....	530
Table 194 – PhL timing characteristics.....	537
Table 195 – Characteristic features of the fieldbus Data Link protocol	582
Table 196 – Transmission function code No.....	614
Table 197 – FCB, FCV in responder	616
Table 198 – Operating parameters.....	629
Table 199 – Summary structure of DLPDUs	657
Table 200 – Structure of Confirmed DLPDUs	658
Table 201 – Structure of Unconfirmed DLPDUs	659
Table 202 – Structure of Acknowledge DLPDU	660
Table 203 – Structure of Immediate-reply DLPDU	660
Table 204 – Primitives and parameters used on SCAN channels	679
Table 205 – Primitives and parameters used on ExSCAN channels	681
Table 206 – Correspondence of max-DLSDU-size and max-data-length for GPA, GPC and ExSCAN channels	682
Table 207 – Primitives and parameters used on GPA channels.....	683
Table 208 – Primitives and parameters used for data transfer on GPC channels.....	687
Table 209 – Primitives and parameters used to disconnect GPC channels.....	688
Table 210 – Primitives and parameters used for DLM-connectionless service.....	707
Table 211 – Predefined functional and group addresses	718
Table 212 – Predefined trigger assignments	719
Table 213 – Format of link-addresses	720
Table 214 – DL-TIME-OFFSET parameter composition.....	721
Table 215 – Time-Offset primitive and parameters	721
Table 216 – DL-time classes.....	721
Table 217 – Time stamp primitives and parameters	722
Table 218 – Time stamp parameter composition	722
Table 219 – DL-Event-time primitive and parameters.....	723
Table 220 – Channel DLMSDU composition and encoding.....	729
Table 221 – Trigger DLMSDU composition and encoding.....	731
Table 222 – Half-forwarding-port DLMSDU composition and encoding.....	732
Table 223 – Channel check DLMSDU composition and encoding.....	733
Table 224 – Trigger-check DLMSDU composition and encoding.....	734

Table 225 – Half-forwarding-port DLMSDU composition and encoding	735
Table 226 – Bus-PDU composition and encoding	738
Table 227 – Individual and group address encoding	754
Table 228 – DLPDU control-field coding	761
Table 229 – Correspondence between name and coding of 8 bits in the control field	762
Table 230 – DL-Timers	768
Table 231 – Bus arbitrator state transition table	812
Table 232 – Bridge object description	815
Table 233 – Channel object description	816
Table 234 – Segment directory object description	817
Table 235 – Network directory object description	817
Table 236 – Service primitives by type	821
Table 237 – Conformance classes	824
Table 238 – Primitives issued by DLS-/DLMS-user to DLI	826
Table 239 – Primitives issued by DLI to DLS-/DLMS-user	826
Table 240 – DLI state table – sender transactions	827
Table 241 – DLI state table – receiver transactions	828
Table 242 – Function GetOffset	829
Table 243 – Function GetLength	829
Table 244 – Function GetRemAdd	829
Table 245 – Function GetDlsUserId	829
Table 246 – PDL_Data_Ack	834
Table 247 – PDL_Data_Ack L_status values	834
Table 248 – PSM	834
Table 249 – GSM	835
Table 250 – PDL_Reset	839
Table 251 – PDL_Set_Value	839
Table 252 – PDL variables	839
Table 253 – PDL_Get_Value	840
Table 254 – PDL_Event	840
Table 255 – Events	840
Table 256 – Encoding of the L_status	846
Table 257 – FCT code (PDL PDU-Types)	847
Table 258 – State transitions of the PDL base protocol machine	852
Table 259 – Counters of the PDL protocol machines	854
Table 260 – Meaning of the "connection" flag	855
Table 261 – State transitions of the PDL protocol machine	856
Table 262 – State transitions of the TRANSMIT protocol machine	859
Table 263 – State transitions of the RECEIVE protocol machine	862
Table 264 – BLL_Data	867
Table 265 – BLL_Data	870
Table 266 – BLL_Reset	871
Table 267 – BLL_Set_Value	871
Table 268 – BLL variables	871
Table 269 – BLL_Get_Value	872
Table 270 – BLL_Event	872

Table 271 – BLL_Event.....	872
Table 272 – State transitions of the BLL operating protocol machine of the master	875
Table 273 – State transitions of the BLL-BAC protocol machine.....	877
Table 274 – State transitions of the BLL operating protocol machine of the slave	879
Table 275 – MAC_Reset	910
Table 276 – MAC_Set_Value	910
Table 277 – MAC variables	911
Table 278 – MAC_Get_Value.....	911
Table 279 – MAC_Event	911
Table 280 – MAC_Event	912
Table 281 – PNM2_Reset	916
Table 282 – M_status values of the PNM2_Reset	916
Table 283 – PNM2_Set_Value	916
Table 284 – M_status values of the PNM2_Set_Value	917
Table 285 – PNM2_Get_Value.....	917
Table 286 – M_status values of the PNM2_Get_Value	917
Table 287 – PNM2_Event	918
Table 288 – MAC Events	918
Table 289 – PNM2_Get_Current_Configuration	918
Table 290 – PNM2_Get_Active_Configuration	919
Table 291 – PNM2_Set_Active_Configuration.....	920
Table 292 – Data direction.....	922
Table 293 – Number of the occupied octets in the parameter channel.....	923
Table 294 – Device class	923
Table 295 – Control data	923
Table 296 – Data width	924
Table 297 – Medium control.....	925
Table C.1 – Generic assignment of FC code points.....	954
Table C.2 – Individual assignment of FC code points	955
Table C.3 – Reply status for SR DLPDUs	958
Table C.4 – Reply status for unitdata-acknowledgment and exchange-unitdata-reply DT DLPDUs.....	959
Table C.5 – Approximate numeric significance of the bits of seven-octet DL-time	964
Table C.6 – Approximate numeric significance of the bits of N(NT), A...A, and three-octet C(NT)	964
Table C.7 [Table 2] – Link node selector addressing	968
Table C.8 [Table 3] – Link-local node selector addressing	968
Table C.9 [Table 6] – Predefined flat non-local DL-addresses	969
Table C.10 [Table 7] – Predefined flat link-local DL-addresses	969
Table C.11 [Table 4] – Link-local node designators.....	970
Table C.12 [Table 5] – Node-local selector addressing	970
Table C.13 [Table 8]– Predefined node-local DL-addresses	970
Table D.1 – Network status indicators.....	973
Table E.1 – Assignment of state machines.....	979
Table E.2 – Data resource	980
Table E.3 – Primitives issued by DL-User to FLC	984

Table E.4 – Primitives issued by FLC to DL-User	985
Table E.5 – Primitives issued by DL-User to DLM	986
Table E.6 – Primitives issued by DLM to DL-User	987
Table E.7 – Parameters used with primitives exchanged between DL-User and FLC	988
Table E.8 – Parameters used with primitives exchanged between DL-User and DLM	988
Table E.9 – FLC / DLM state table	990
Table E.10 – FLC / DLM function table	1003
Table E.11 – Primitives issued by DLM to MAC	1010
Table E.12 – Primitives issued by MAC to DLM	1010
Table E.13 – Parameters used with primitives exchanged between DLM and MAC	1010
Table E.14 – Local MAC variables	1011
Table E.15 – MAC state table	1012
Table E.16 – MAC function table	1034
Table E.17 – Primitives issued by DLM to SRC	1041
Table E.18 – Primitives issued by SRC to DLM	1041
Table E.19 – Primitives issued by MAC to SRC	1041
Table E.20 – Primitives issued by SRC to MAC	1042
Table E.21 – Parameters used with primitives exchanged between MAC and SRC	1042
Table E.22 – FC structure	1042
Table E.23 – Local variables of SRC	1043
Table E.24 – SRC state table	1043
Table E.25 – SRC functions	1055
Table J.1 – DL-subnetwork configuration in the form of a matrix	1083
Table J.2 – Acquire_Configuration	1083
Table J.3 – State transitions of the state machine for the acquisition of the current configuration	1085
Table J.4 – Check_Configuration	1086
Table J.5 – Compare_Slave	1087
Table J.6 – State transitions of the state machine for comparing two configurations	1089
Table J.7 – State transitions of the state machine for comparing one line of two configuration matrixes	1091

INTERNATIONAL ELECTROTECHNICAL COMMISSION

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELD BUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 4: Data Link protocol specification

FOREWORD

- 1) The IEC (International Electrotechnical Commission) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of the IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, the IEC publishes International Standards. Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. The IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of the IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested National Committees.
- 3) The documents produced have the form of recommendations for international use and are published in the form of standards, technical specifications, technical reports or guides and they are accepted by the National Committees in that sense.
- 4) In order to promote international unification, IEC National Committees undertake to apply IEC International Standards transparently to the maximum extent possible in their national and regional standards. Any divergence between the IEC Standard and the corresponding national or regional standard shall be clearly indicated in the latter.
- 5) The IEC provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with one of its standards.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this International Standard may involve the use of patents as follows, where the [xx] notation indicates the holder of the patent right:

Type 1 time synchronization and possibly other Types:

JP Hei4-35096 [YC] Communication control method

Type 2 and possibly other Types:

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

Type 3 and possibly other Types:

DE 36 43 979 C2	[SI]	Deterministisches Zugriffsverfahren nach dem Tokenprinzip für eine Datenübertragung
DE 36 43 979 A1	[SI]	Deterministisches Zugriffsverfahren nach dem Tokenprinzip für eine Datenübertragung

Type 8 and possibly other Types:

DE 41 00 629 C1 [Px] Steuer- und Datenübertragungsanlage

The IEC takes no position concerning the evidence, validity and scope of these patent rights.

The holders of these patent rights have assured the 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 the IEC. Information may be obtained from:

[AB]: Rockwell Technologies, LLC
Allen-Bradley Co, LLC
1201 S. Second Street
Milwaukee, WI 53204
USA
Attention: Intellectual Property Dept.

[Px]: Phoenix Contact GmbH & Co. KG
Referat Patente / Patent Department
Postfach 1341
D-32819 Blomberg
Germany

[SI]: SIEMENS AG
Ludwig Winkel
Siemensallee 84
D-76181 Karlsruhe
Germany

[YC]: Jun-ichi Tsuboi, Mr.
Yamatake Corporation
Shibuyaku Shibuya, 2-12-19
Tohtate International Building
150-8316 Tokyo
Japan

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

International Standard IEC 61158-4 has been prepared by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

This third edition cancels and replaces the second edition published in 2000. This third edition constitutes a technical revision.

The text of this standard is based on the following documents:

FDIS	Report on voting
65C/291/FDIS	65C/299/RVD

Full information on the voting for the approval of this standard can be found in the report on voting indicated in the above table.

This edition includes the following significant changes from the prior edition:

- the removal of PhL-related material from the Annexes, which has now been incorporated in the concurrent new edition of IEC 61158-2;
- substantial additions and revisions to the Type 3 protocol specification;
- the addition, as Annex E, of normative state machines for most of the Type 3 protocol;
- a substantial rewrite of the Type 6 protocol specification.

This publication has been drafted in accordance with ISO/IEC Directives, Part 2.

The committee has decided that the contents of this publication will remain unchanged until 2007. At this date, the publication will be

- reconfirmed;
- withdrawn;
- replaced by a revised edition, or
- amended.

IEC 61158 consists of the following parts, under the general title *Digital data communications for measurement and control — Fieldbus for use in industrial control systems*:

Part 1: Overview and guidance for the IEC 61158 series

Part 2: Physical Layer specification and service definition

Part 3: Data Link Service definition

Part 4: Data Link Protocol specification

Part 5: Application Layer Service definition

Part 6: Application Layer protocol specification

The contents of the corrigendum of July 2004 have been included in this copy.

0 Introduction

0.1 General

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, which is based in part on the Basic Reference Model for Open Systems Interconnection. Both Reference Models subdivide the area of standardization for interconnection into a series of layers of specification, each of manageable size.

The Data Link Protocol provides the Data Link Service by making use of the services available from the Physical Layer. The relationship between the International Standards for Fieldbus Data Link Service, Fieldbus Data Link Protocol, Fieldbus Physical Service and Systems Management is illustrated in Figure 1.

NOTE Systems Management, as used in this standard, is a local mechanism for managing the layer protocols.

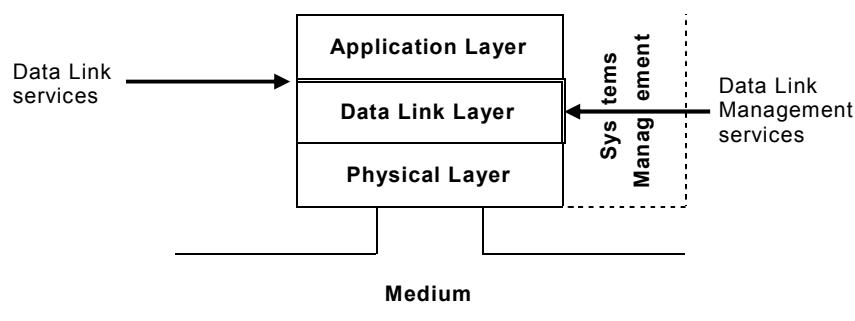


Figure 1 – Relationship of IEC 61158-4 to other fieldbus layers and to users of the fieldbus Data Link service

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.

0.2 Nomenclature for references within this standard

Clauses, including annexes, can be referenced in their entirety, including any subordinate subclauses, as “clause N” or “Annex N”, where N is the number of the clause or letter of the annex.

Subclauses can be referenced in their entirety, including any subordinate subclauses, as “N.M” or “N.M.P” and so forth, depending on the level of the subclause, where N is the number of the subclause or letter of the annex, and M, P and so forth represent the successive levels of subclause up to and including the subclause of interest.

When a clause or subclause contains one or more subordinate subclauses, the text between the clause or subclause heading and its first subordinate subclause can be referenced in its entirety as “N.0” or “N.M.0” or “N.M.P.0” and so forth, where N, M and P are as above. Stated differently, a reference ending with “.0” designates the text and figures between a clause or subclause header and its first subordinate subclause.

NOTE This nomenclature provides a means of referencing text in hanging clauses. Such clauses existed in earlier editions of this standard. Some of those hanging clauses are maintained in this edition to minimize the disruption to existing national and multi-national standards and consortia documents which reference that prior subclause numbering.

DIGITAL DATA COMMUNICATIONS FOR MEASUREMENT AND CONTROL – FIELDBUS FOR USE IN INDUSTRIAL CONTROL SYSTEMS –

Part 4: Data Link protocol specification

1 Scope and object

1.1 Overview

The Fieldbus Data Link Layer provides basic time-critical messaging communications between devices in an automation environment.

The purpose of this part of IEC 61158 is to define eight distinct and non-interoperable fieldbus Data Link protocols. Each protocol defined in this part of this standard is most closely related to, and lies within the field of application of, the corresponding services of IEC 61158-3, the fieldbus Data Link Service Definition. The eight distinct protocols are the following:

Type 1 — A DL-protocol for the Type 1 DL-service. The maximum system size is 56k links of 232 nodes, each with 240 DLSAP-addresses and related peer and publisher DLCEPs, plus another 768 link-wide group DL-addresses per link, plus another 256 DLSAP-addresses and 2816 peer and publisher DLCEPs per link which can be allocated among the link's nodes, plus 2^{27} group DL-addresses for the extended link. Fewer links or nodes permits an increase in the number of DLSAP-addresses and related DLCEPs.

Type 2 — A DL-protocol for the Type 2 DL-service. 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.

Type 3 — A DL-protocol for the Type 3 DL-service. The maximum system size is an unlimited number of links of 127 nodes, each with 66 DLSAP-addresses.

Type 4 — A DL-protocol for the Type 4 DL-service. The maximum system size is a single link with a virtually unlimited number of nodes, each with 256 DLSAP-addresses connected to a single DLS-user.

Type 5 — A second DL-protocol for the Type 1 DL-service. The maximum system size for this protocol is 240 links of 240 nodes, each with 2^{16} DLSAP-addresses and related peer and publisher DLCEPs, plus another 2^{27} link-wide group DL-addresses per link, plus 2^{27} group DL-addresses for the extended link. A normative specification of this protocol was not available at the time of compilation of this edition.

Type 6 — A DL-protocol for the Type 6 DL-service. The maximum system size is a six-deep hierarchy of bridges for a total of 240^6 links, each of 240 nodes, where each node has 240 DLSAP-addresses and over 60k related peer and publisher DLCEPs, plus 16 group-DL-addresses and 4k peer and publisher DLCEPs per link which can be allocated among the link's nodes, plus a very large number of group DL-addresses for each hierarchical subtree of the extended link.

Type 7 — A DL-protocol for the Type 7 DL-service. The maximum system size is 126 links of 256 nodes, each with 16 DLSAP-addresses and 16 group DL-addresses, plus another 28k DLSAP-addresses and related peer and publisher DLCEPs per link which can be allocated among the link's nodes, plus another 28k link-wide group DL-addresses per link, plus 2^{23} group DL-addresses for the extended link.

Type 8 — A DL-protocol for the Type 8 DL-service. The maximum system size is 1 link of 256 nodes with preconfigured DLCEPs.

1.2 Specifications

This standard specifies:

- a) procedures for eight distinct and largely-incompatible protocols 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 Data Link (DL) Protocol Data Units used for the transfer of data and control information by each of those eight protocols, and their representation as Physical Interface Data Units.

NOTE Throughout IEC 61158-4, gray boxes have been used in the tables to indicate that the specified field is not a conceptual part of the specific DLPDU.

1.2.1 Type 1: Additional characteristics

This protocol provides communication opportunities to all participating data-link entities

- a) in a cyclic asynchronous manner, sequentially to each of those data-link entities, and
- b) in a synchronous manner, either cyclically or acyclically, according to a pre-established schedule.

The specified protocol also provides means of changing the set of participating data-link entities and of modifying the set of scheduled communications opportunities. When the set of scheduled communications opportunities is null, the distribution of communication opportunities to the participating data-link entities is completely asynchronous.

Thus this protocol can be characterized as one which provides access asynchronously but with a synchronous overlay.

1.2.2 Type 2: Additional characteristics

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 µs.

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

1.2.3 Type 3: Additional characteristics

This protocol provides communication opportunities to a pre-selected “master” subset of data-link entities in a cyclic asynchronous manner, sequentially to each of those data-link entities. Other data-link entities communicate only as permitted and delegated by those master data-link entities.

For a given master, its communications with other data-link entities can be cyclic, or acyclic with prioritized access, or a combination of the two.

This protocol provides a means of sharing the available communication resources in a fair manner. There are provisions for time synchronization and for isochronous operation.

1.2.4 Type 4: Additional characteristics

This protocol provides a means of connecting devices through a partial mesh network, such that most failures of an interconnection between two devices can be circumvented. In common practice the devices are interconnected in a non-redundant hierarchical manner reflecting application needs

1.2.5 Type 5: Additional characteristics

This protocol is an adaptation of the Type 1 protocol for a PhL consisting of high-speed counter-rotating prioritized buffer insertion rings. It provides the identical Type 1 DL-service. The Type 1 and Type 5 protocols are interconnectable through bridges, seamlessly providing the entire Type 1 DL-service. These bridges forward Type 1 DLPDUs between Type 1 and Type 5 links without change, providing end-to-end error detection through a frame check sequence appended by the originating DLE.

This protocol is anticipated to be most appropriate for connecting complex control systems which must exchange large amounts of information. It provides means to maintain clock synchronization across an extended link with a precision better than 100 ns.

A normative specification of this protocol was not available at the time of compilation of this edition.

1.2.6 Type 6: Additional characteristics

This protocol provides a means of interchanging fixed-length input/output data and variable-length segmented messages between peer devices interconnected in a multidrop bus topology. The exchange of input/output data and selected messaging data is totally synchronous by configuration, and each exchange is unaffected by other traffic.

1.2.7 Type 7: Additional characteristics

This protocol provides communication opportunities to all participating data-link entities

- a) in a synchronously-starting cyclic manner, according to a pre-established schedule, and
- b) in a cyclic or acyclic asynchronous manner, as requested each cycle by each of those data-link entities.

Thus this protocol can be characterized as one which provides cyclic and acyclic access asynchronously but with a synchronous restart of each cycle.

1.2.8 Type 8: Additional characteristics

This protocol provides a highly-optimized means of interchanging fixed-length input/output data and variable-length segmented messages between a single master device and a set of slave devices interconnected in a loop (ring) topology. The exchange of input/output data is totally synchronous by configuration, and is unaffected by the messaging traffic.

Devices are addressed implicitly by their position on the loop. The determination of the number, identity and characteristics of each device can be configured, or can be detected automatically at start-up.

1.3 Procedures

The procedures are defined in terms of

- a) the interactions between peer DL-entities (DLEs) through the exchange of fieldbus Data Link Protocol Data Units;
- 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 Physical 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. Clause 12 specifies profiles for the Type 1 fieldbus DL-protocol.

1.5 Conformance

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

The supplier of a Type 1 protocol implementation which is claimed to conform to this standard should complete a copy of the PICS proforma (see clause 13), including the information necessary to identify both the supplier and the implementation.

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 61131-3:1993, *Programmable controllers – Part 3: Programming languages*

IEC 61158, *Digital data communications for measurement and control – Fieldbus for use in industrial control systems*

IEC 61158-2:2003, *Part 2: Digital data communications for measurement and control – Fieldbus for use in industrial control systems — Physical layer specification and service definition*

IEC 61158-3:2003, *Part 3: Digital data communications for measurement and control – Fieldbus for use in industrial control systems — Data link service definition*

IEC 61158-5:2003, *Part 5: Digital data communications for measurement and control – Fieldbus for use in industrial control systems — Application layer service definition*

IEC 61158-6:2003, *Part 6: Digital data communications for measurement and control – Fieldbus for use in industrial control systems — Application layer protocol specification*

ISO/IEC 7498 (all parts), *Information technology – Open Systems Interconnection – Basic Reference Model*

ISO/IEC 7498-1:1994, *Information technology – Open Systems Interconnection – Basic Reference Model — Basic Reference Model: The Basic Model*

ISO/IEC 7498-3:1997, *Information technology – Open Systems Interconnection – Basic Reference Model — Basic Reference Model: Naming and addressing*

ISO/IEC 8886:1996, *Information technology – Open Systems Interconnection – Data link service definition for Open Systems Interconnection*

ISO/IEC 10038:1993, *Information technology – Telecommunications and information exchange between systems – Local area networks – Media Access Control (MAC) bridges*

ISO/IEC 10731:1994, *Information technology – Open Systems Interconnection – Basic Reference Model – Conventions for the definition of OSI services*

CENELEC EN 50170, *General purpose field communication system*

CENELEC EN 50170-3-1:1996, *General purpose field communication system — Data link layer definitions*

CENELEC EN 50170-3-7:1996, *General purpose field communication system — Network management*

3 Terms and definitions

For the purposes of this document, the following terms and definitions 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	[7498-3]
3.1.2	calling-DL-address	[7498-3]
3.1.3	centralized multi-end-point-connection	[7498-1]
3.1.4	correspondent (N)-entities	[7498-1]
	correspondent DL-entities (N=2)	
	correspondent Ph-entities (N=1)	
3.1.5	demultiplexing	[7498-1]
3.1.6	DL-address	[7498-3]
3.1.7	DL-address-mapping	[7498-1]
3.1.8	DL-connection	[7498-1]
3.1.9	DL-connection-end-point	[7498-1]
3.1.10	DL-connection-end-point-identifier	[7498-1]
3.1.11	DL-connection-mode transmission	[7498-1]

3.1.12 DL-connectionless-mode transmission	[7498-1]
3.1.13 DL-data-sink	[7498-1]
3.1.14 DL-data-source	[7498-1]
3.1.15 DL-duplex-transmission	[7498-1]
3.1.16 DL-facility	[7498-1]
3.1.17 DL-local-view	[7498-3]
3.1.18 DL-name	[7498-3]
3.1.19 DL-protocol	[7498-1]
3.1.20 DL-protocol-connection-identifier	[7498-1]
3.1.21 DL-protocol-control-information	[7498-1]
3.1.22 DL-protocol-data-unit	[7498-1]
3.1.23 DL-protocol-version-identifier	[7498-1]
3.1.24 DL-relay	[7498-1]
3.1.25 DL-service-connection-identifier	[7498-1]
3.1.26 DL-service-data-unit	[7498-1]
3.1.27 DL-simplex-transmission	[7498-1]
3.1.28 DL-subsystem	[7498-1]
3.1.29 DL-user-data	[7498-1]
3.1.30 flow control	[7498-1]
3.1.31 layer-management	[7498-1]
3.1.32 multiplexing	[7498-3]
3.1.33 naming-(addressing)-authority	[7498-3]
3.1.34 naming-(addressing)-domain	[7498-3]
3.1.35 naming-(addressing)-subdomain	[7498-3]
3.1.36 (N)-entity	[7498-1]
DL-entity	
Ph-entity	
3.1.37 (N)-interface-data-unit	[7498-1]
DL-service-data-unit (N=2)	
Ph-interface-data-unit (N=1)	
3.1.38 (N)-layer	[7498-1]
DL-layer (N=2)	
Ph-layer (N=1)	
3.1.39 (N)-service	[7498-1]
DL-service (N=2)	
Ph-service (N=1)	

3.1.40 (N)-service-access-point	[7498-1]
DL-service-access-point (N=2)	
Ph-service-access-point (N=1)	
3.1.41 (N)-service-access-point-address	[7498-1]
DL-service-access-point-address (N=2)	
Ph-service-access-point-address (N=1)	
3.1.42 peer-entities	[7498-1]
3.1.43 Ph-interface-control-information	[7498-1]
3.1.44 Ph-interface-data	[7498-1]
3.1.45 primitive name	[7498-3]
3.1.46 reassembling	[7498-1]
3.1.47 recombining	[7498-1]
3.1.48 reset	[7498-1]
3.1.49 responding-DL-address	[7498-3]
3.1.50 routing	[7498-1]
3.1.51 segmenting	[7498-1]
3.1.52 sequencing	[7498-1]
3.1.53 splitting	[7498-1]
3.1.54 synonymous name	[7498-3]
3.1.55 systems-management	[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.

For the purpose of this part of IEC 61158, the following definitions also apply:

3.3.1 DL-address user-request queue

multi-priority FIFO-within-priority queue, associated with a specific DL-address of a DLE, of in-process DLS-user-requests. This queue is partitioned into three disjoint sections:

1) those requests which have already been transmitted but remain available for retransmission, confirmation, or both — that is, which have not reached their maximum confirm delay and which

— are still within the DLCEP's transmit window, or

— are awaiting an acknowledging response DLPDU at the DLSAP;

2) those requests which are ready for transmission but have not been completely transmitted;

3) those requests which are either awaiting a DL-COMPEL-SERVICE request to release them for transmission, or are outside the DLCEP's transmit window, or both.

NOTE 1 Section 1 of the queue contains DLS-user requests which may have been successfully communicated to another DLS-user. Retransmission of all or part of the DLS-user data associated with these requests may be required, and can be attempted before the requests are purged from this partition (by reset or peer acknowledgment on a peer DLC; by reset or user-request timeout or the need for sequence-number reuse on a multi-peer DLC).

NOTE 2 Section 2 of the queue contains DLS-user requests which are ready for transmission, but could not have been completely communicated to another DLS-user.

NOTE 3 Section 3 of the queue can be non-empty only when the DL-scheduling-policy for the DL-address is EXPLICIT, or when the number of queued DLSDUs exceeds the DLCEP's transmit window, or both.

NOTE 4 Members of a given priority are advanced from one section to another until they are removed from the queue. (In practice, the section partitions may be moved, rather than the members.) The FIFO ordering within each priority is strictly maintained.

One such queue is associated with each DLSAP-address, each peer or publisher DLCEP-address, and each subscriber's DLCEP (which can be considered to be associated with an implicit DLCEP-address) of the DLE, and with the DLE's NODE DL-address (see 3.3.1 and 7.2.2.3)

3.3.2 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

3.3.2.1 DLE unscheduled-service queue

multi-priority FIFO-within-priority queue of

a) **references** to DL-address user-request queues (see 3.3.1);

NOTE See 6.7.1.17a) for a more elaborate definition.

b) **references** to locally-scheduled active sequences (see clause 11) resulting from DL-SCHEDULE-SEQUENCE requests (see 10.4.3.1);

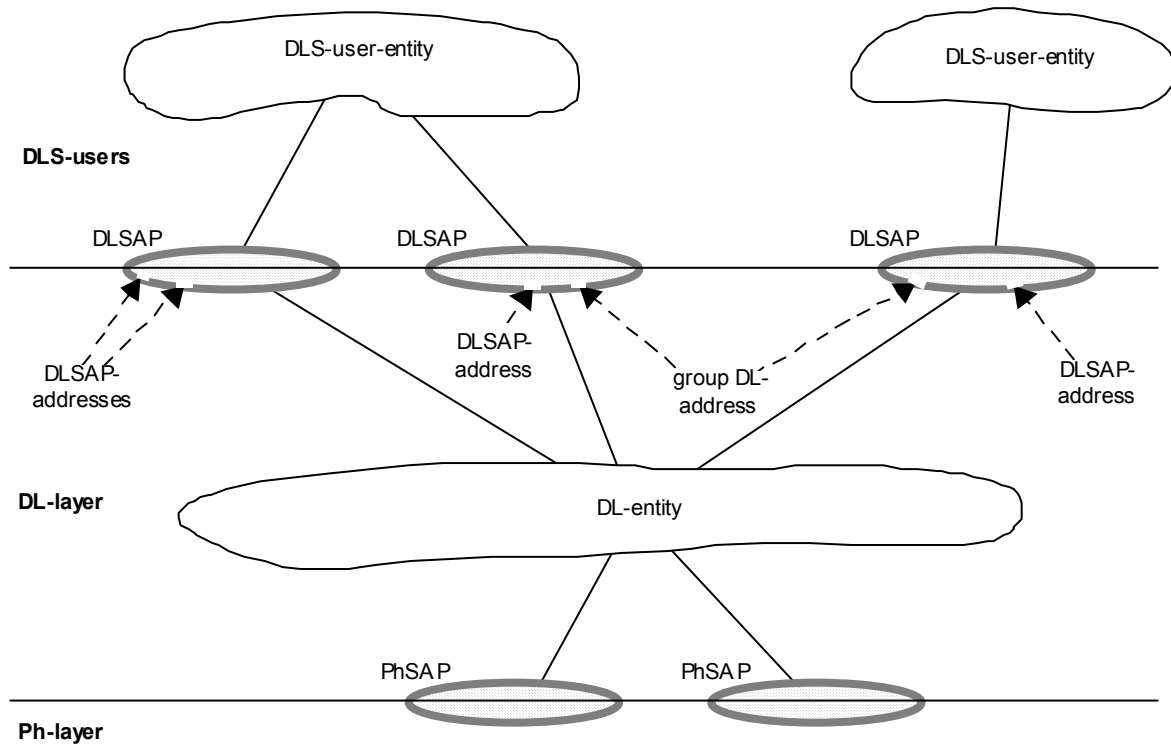
c) DT DLPDUs containing DLS-user data which are delayed responses to received CD and ED DLPDUs, queued in support of the DL-UNITDATA-EXCHANGE service

NOTE Since this is a multi-priority FIFO-within-priority queue, members are removed in priority order, and within a single priority, in FIFO order.

3.3.3 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 2.)



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 2 – Relationships of DLSAPs, DLSAP-addresses and group DL-addresses

3.3.4 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

3.3.5 (individual) DLSAP-address

DL-address that designates only one DLSAP within the extended link. A single DL-entity may have multiple DLSAP-addresses associated with a single DLSAP.

3.3.6 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

3.3.7 frame

denigrated synonym for DLPDU

3.3.8 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

3.3.9 node

single DL-entity as it appears on one local link

3.3.10 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

3.3.11 sending DLS-user

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

3.4 Type 1: Additional terms and definitions

NOTE For historic reasons, some of the following definitions are more extensive than is common to the Definition clauses of other IEC standards.

3.4.1 bridge, DL-router

DL-relay entity which performs selective store-and-forward and routing functions.

- a) to connect two or more separate DL-subnetworks (links) to form a unified DL-subnetwork (the extended link); and
- b) to provide a means by which two end systems can communicate, when at least one of the end systems is periodically inattentive to the interconnecting DL-subnetwork,

and also provides time synchronization among the links to which it is forwarding

3.4.2 delocalization

DLE-internal process of converting synonymous DL-addresses to a canonical form for transmission or during reception (see 7.2.2)

3.4.3 dominant token

unique right to initiate the next transmission on the local link (see token)

NOTE 1 A reply token is the dominant token on the local link during the period after its creation and before its expiration or return, and the reply token holder (the responder) holds the unique right to transmit.

NOTE 2 Otherwise, when no reply token exists, a delegated token is the dominant token on the local link during the period after its creation and before its expiration or return, and its token holder (the initiator) holds the unique right to transmit.

NOTE 3 At all other times the scheduler token is the dominant token on the local link, and the LAS DLE (which functions as the initiator) holds the unique right to transmit

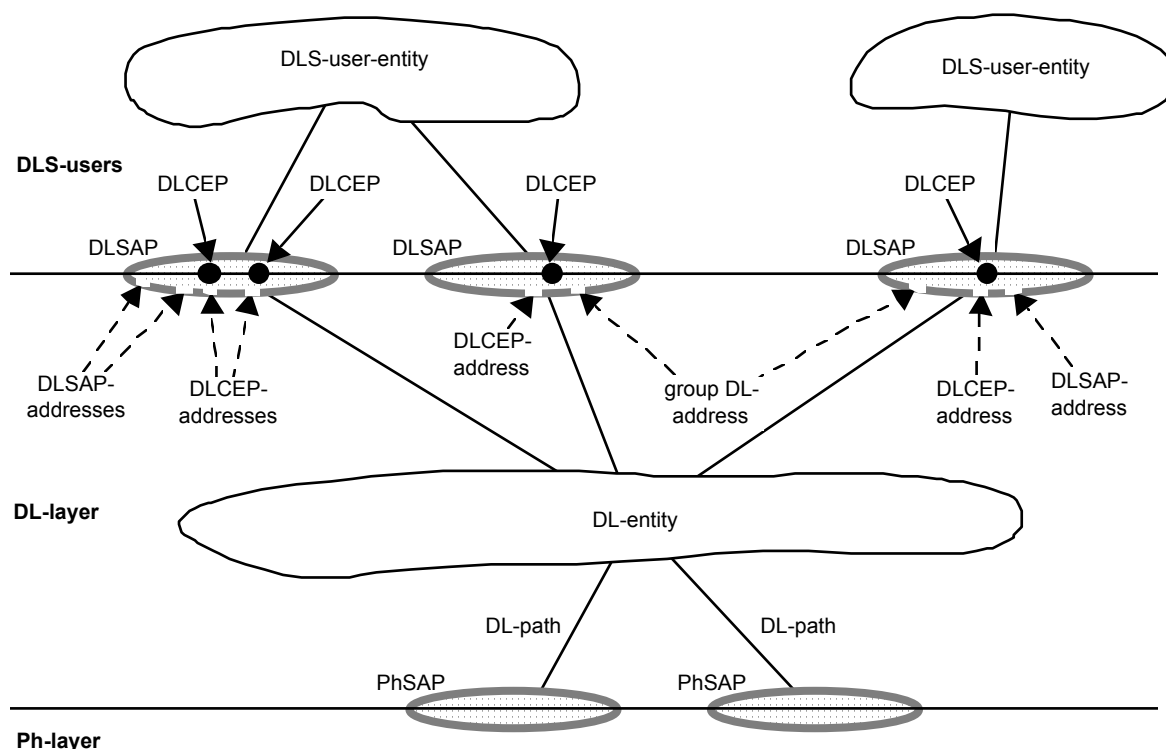
3.4.4 DLCEP-address

DL-address which designates either

- a) one peer DL-connection-end-point; or
- b) one multi-peer publisher DL-connection-end-point, and implicitly the corresponding set of subscriber DL-connection-end-points

where each DL-connection-end-point exists within a distinct DLSAP and is associated with a corresponding distinct DLSAP-address

NOTE This is an extension of the use of DL-addresses beyond that specified in ISO/IEC 7498-3. (See Figure 3.)



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. A DLCEP-address also designates a specific point of information flow (its DLCEP) within the DLSAP.

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

NOTE 4 This figure also shows the relationships of DL-paths and PhSAPs.

Figure 3 – Relationships of DLCEPs and DLCEP-addresses to DLSAPs, DLSAP-addresses and group DL-addresses

3.4.5 DLSEP-address

DL-address which designates a DL-scheduling-end-point within a DLE

NOTE This is an extension of the use of DL-addresses beyond that specified in ISO/IEC 7498-3.

3.4.6 fractional-duty-cycle (FDC)

DLE which is not continuously attentive to received signaling, usually found in an end-system with an austere electrical power budget

NOTE 1 Such a DLE generally requires the assistance of a bridge (DL-relay) DLE to communicate with other DLEs on the same link, to provide for cases where the two DLEs are not concurrently active.

Three classes of FDC DLE have been identified:

class A — FDC DLEs which are totally inattentive to their local links during their “sleep” periods, and control the timing of their temporary “re-awakening” to link communications solely on the basis of internal DL-time or some other measurement;

class B — FDC DLEs which monitor the link during their periods of “sleep” for a particular wakeup DLPDU (see 8.21) addressed specifically to the DLE’s node DL-address, “re-awakening” upon receipt of such a DLPDU;

class C — FDC DLEs which monitor the link during their periods of “sleep” for any DLPDUs addressed specifically to the DLE’s node DL-address, “re-awakening” upon receipt of such a DLPDU.

NOTE 2 DLEs which “sleep” between DLPDUs, but that respond to all DLPDUs addressed to any of their DL-addresses, are not FDC DLEs

3.4.7 immediate reply

DLPDU sent by the responder in a transaction as a solicited reply to the immediately-prior DLPDU

NOTE The original DLPDU sent by the initiator of the transaction specifies the constraints on the replying DLPDU.

3.4.8 immediate-response-recovery-delay

measure, similar to maximum-response-delay, of the worst-case period of **peer** PhE-inactivity which can be observed

- a) by a DLE holding a delegated token while awaiting an immediate response from another correctly-functioning DLE on the local link;
- b) by any other DLE on the local link, after receiving the initiating transaction of a two-DLPDU transaction, before detecting the link activity caused by transmission of that immediate response DLPDU;
- c) by the LAS DLE while waiting for the initial link activity after
 - a reply token (via a PROBE NODE-ADDRESS (PN) DLPDU);
 - a delegated token (via a PASS TOKEN (PT) or EXECUTE SEQUENCE (ES) DLPDU);
 - the scheduler token (via a TRANSFER LAS (TL) DLPDU)

to another correctly functioning DLE on the local link

NOTE 1 More formally, immediate-response-recovery-delay provides a bound for the worst-case delay between receipt of the Ph-Data confirm primitive for a Ph-Data request primitive whose PhIDU specified end-of-data-and-activity (see 6.4.3), and subsequent receipt of the next start-of-activity PhIDU (see 6.4.3), that can be observed by the sending DLE on the local link during

- 1) an initiator-responder interaction, in which a transaction-initiating DLE sends a DLPDU requiring an immediate reply to a responding DLE, and that responding DLE replies by sending a second DLPDU;
- 2) a token-passing interaction, in which the LAS DLE sends a delegated token DLPDU to the next intended token-holding DLE, or the scheduler token to a successor (as LAS) DLE, and that addressed DLE responds by sending a second DLPDU.

NOTE 2 The value of immediate-response-recovery-delay, in units of one slot-time, is (maximum-response-delay + 1)

3.4.9 initiator

DLE role in which a DLE sends a DLPDU to a peer responder DLE, which immediately sends a reply DLPDU back to the initiator DLE (and potentially to other DLEs) as part of the same transaction

NOTE Some prior national standards have referred to this role as a “master” role.

3.4.10 Link Active Scheduler (LAS)

special-purpose elective role of a DLE which schedules the local link and serves as the local source of DL-time for the link

NOTE The LAS functions exist within each link master (LM) DLE, which contends for and then activates those LAS functions after detecting the absence on the link of a DLE whose LAS functions are active. The DLE serving as LAS, sometimes referred to as the LAS DLE, receives, and responds to, scheduling requests from all DLEs on the link, including itself. It also receives and responds to requests for the current DL-time.

3.4.11 link-id

two-octet primary identifier for the local link, within the extended link, whose values are constrained as specified in 6.3.2.1

3.4.12 Link Master (LM)

DLE which can also provide the LAS functions for the link, including initializing and scheduling the link

3.4.12.1 maximum-inactivity-to-claim-LAS-delay

configuration parameter of each local link, its minimum value is the worst-case internal delays of any Link Master DLE and associated PhEs connected to that link

NOTE This delay is computed as the sum of the following two components:

- a) the internal delay between
 - 1) the moment of presentation of the last non-silent PhPDU of a transmission to the DLE's associated PhE at that PhE's point of connection to the local link;
 - 2) the resulting start of the DLE's internal timer which monitors link inactivity;
- b) the internal delay between
 - 1) the expiration of that timer;
 - 2) the resulting moment of presentation of the first non-silent PhPDU of a transmission of the required Claim Las (CL) DLPDU by the DLE's associated PhE at that PhE's point of connection to the local link

The unit of measurement of this aggregate delay is one eighth of the transmission period of one octet (that is, one nominal "bit-period"). The range of values of this parameter is 1 to 4 095 nominal "bit-periods"

3.4.13 maximum-response-delay

measure, in units of one slot-time, greater than the worst-case period of **local** PhE-inactivity which can be observed by a DLE

- a) which has just received a delegated token, before initiating transmission of a DLPDU;
- b) which has just received a reply token, before initiating transmission of the required immediate reply DLPDU.

NOTE 1 A DLE's minimum required value for maximum-response-delay is determined by measuring at the responding DLE the delay between concluding the reception of a requesting DLPDU and initiating the transmission of the immediately-subsequent responding DLPDU.

NOTE 2 Maximum-response-delay is a configuration parameter of each local link, and has a value of 1 to 11. When multiplied by the duration of one slot-time, the product must be at least as large as the largest of the maximum-response-delay values required by each of the current or anticipated DLEs on the local link

3.4.14 multi-peer DLC

centralized multi-end-point DL-connection offering DL-duplex-transmission between a single distinguished DLS-user, known as the **publisher** or **publishing** DLS-user, and a set of peer but undistinguished DLS-users, known collectively as the **subscribers** or **subscribing** DLS-users, where the publishing DLS-user can send to the subscribing DLS-users as a group (but not individually), and the subscribing DLS-users can send to the publishing DLS-user (but not to each other)

NOTE 1 A multi-peer DLC always provides asymmetrical service. It may also be negotiated to provide only DL-simplex service, either from the publisher to the subscribers, or from the subscribers to the publisher. In this last case, the characterizations as publisher and subscriber are misnomers.

NOTE 2 The publishing DLS-user may need to employ control of its publishing rate, because a subscribing DLS-user cannot exert either flow or rate control on its publishing peer entity. Similar considerations apply to subscribing DLS-users with respect to their sending DLSUs to the publishing DLS-user.

3.4.15 NODE DL-address

DL-address which designates the (single) DL-entity associated with a single node (see 3.3.9) on a specific local link

3.4.16 node-id

one-octet primary identifier for the DLE on the local link, whose values are constrained as specified in 6.3.2.2. A value of zero is also permitted, which inhibits all transmission by this DLE

3.4.17 node-timer

frequency-adjustable counter, maintained by the local DLE, which is used to provide the DLS-user with a local multi-partite sense of DL-time (see 6.5.4 and 10.4.1.1) such that one component of that DL-time forms a monotonically-increasing sense of local time available for use within the attached end system

NOTE 1 Conceptually, the node-timer counts in nominal units of 2-13 ms and has a period of over 100 years. Therefore, any actual counter shall be a binary counter whose least significant bit has a nominal weight of $2^{\pm N}$ ms, with a rollover period greater than the expected maximum interval between resets of the DLE

When no information about that maximum interval is available, an interval of five years may be assumed.

NOTE 2 The node-timer is also used within the DL-protocol to provide a shared sense of DL-time which is used both to synchronize DLE scheduling actions, where appropriate, and to synchronize the rate of drift of all of the node-timers on the extended link. This latter is achieved by adjusting the frequency of each DLE's node-timer such that the DLE's monotonic sense of local time maintains an approximately constant phase relationship with that of the DLE serving as the time-reference DLE

This adjustment is the reason why the weight of the node-timer unit of counting is only nominally $2^{\pm N}$ ms

3.4.18 peer DLC

point-to-point DL-connection offering DL-duplex-transmission between two peer DLS-users where each can be a sending DLS-user, and each as a receiving DLS-user may be able to exert flow control on its sending peer

NOTE A peer DLC is negotiated to provide either symmetrical service or asymmetrical service. A peer DLC may also be negotiated to provide only DL-simplex service.

3.4.19 responder

DLE role in which a DLE sends a DLPDU as an immediate reply to a DLPDU received from a peer initiator DLE, all as part of a single transaction

NOTE Some prior national standards have referred to this role as a "slave" role.

3.4.20 slot-time

configuration parameter of each local link, measured as an integral multiple of the transmission period of one complete octet, with an integral value of between 1 and 4 095 octets:

- 1) Slot-time is a fundamental link parameter with multiple uses:
 - i) Slot-time is used by each Link Master DLE connected to the link in determining the amount of time for which that DLE monitors the link for inactivity before sending a CLAIM LAS (CL) DLPDU. Slot-time is defined such that the nominal link-inactivity monitoring periods of two DLEs which
 - have consecutive NODE DL-addresses;
 - which do not hold any token
 differ by exactly one slot-time.
 - ii) Slot-time is used by each DLE on the local link to compute the durations of all other periods of link inactivity which the DLE must monitor. All such durations are specified as standardized or configured multiples of one slot-time.
- 2) Slot-time is a configured minimum upper bound on the maximum two-way asynchronism in immediate communications among interacting DLEs on the local link when trying to (re)initialize the link, maximized across all pairs of DLEs on that local link. Given this viewpoint, slot-time is an aggregate measure of the worst-case implementation delays within the intervening media, the PhL, and the PhL/DLL interfaces, all of which limit the rapidity of two-way DLE interaction on the local link.

Slot-time is computed as the sum of

- a) the worst-case two-way propagation delays through the intervening media and intervening PhEs, such as repeaters, between any two PhEs associated with their respective DLEs on the local link, measured between the point of connection to the local link of each of those PhEs, including worst-case internal logic and analog circuit delays in any intermediate repeaters;
- b) the maximum-inactivity-to-claim-LAS-delay as defined in 3.4.21;
- c) a safety factor, which is used to account for
 - 1) the difference in two relevant internal delays consisting of
 - i) the delay between
 - A) the presentation of the first non-silent PhPDU to the DLE's associated PhE at that PhE's point of attachment
 - B) the indication of start of activity from that PhE to that DLE;
 - ii) the delay between
 - A) the presentation of the last non-silent PhPDU to the DLE's associated PhE at that PhE's point of attachment
 - B) the indication of end of activity from that PhE to that DLE;
 - 2) the rate differences in the internal timer clocks among the DLEs on the local link;
 - 3) the limited resolution of the measurements and potential measurement errors;
 - 4) any extra delay needed to make the sum of a), b) and c) round to an integral multiple of the transmission period of one octet

NOTE 1 These delays are defined in a manner that permits their measurement.

NOTE 2 Finer resolution than one octet is not possible without knowledge of the specific associated PhL, because Ph-timing comes from the PhL and the formal PhL-DLL interface provides only octet timing (see 6.4).

NOTE 3 The definitions of 3.4.20 to 3.4.23 are not recursive, because 3.4.20 is based on a different measurement than 3.4.13 to 3.4.23. The definitions of 3.4.13 to 3.4.23 are in units of one slot-time to reduce the complexities of implementations of this protocol. As a result, a single 8-bit hardware timer, prescaled by the slot-time, enabled on bus inactivity, with usage-specific stopping and retriggering conditions, can provide the required functionality for all of the slot-time-based timers of this protocol.

3.4.21 timeliness, DL-timeliness

attribute of a datum which provides an assessment of the temporal currency of that datum. This attribute is of particular importance in sampled-data systems, which may need to make decisions based on the timeliness, or lack of timeliness, of current data samples

NOTE 1 As a general rule, timeliness is a user attribute which can be affected negatively by the various layers of the data transport system. That is, a datum which was timely when the requesting user presented it to a data communications subsystem for transmission may become untimely due to delays in the communications subsystem.

NOTE 2 DL-timeliness is an attribute of a DLS-user datum relating the timing of a DLS-user/DLE interaction which writes or reads that datum to one or more other DLS-user/DLE interactions.

NOTE 3 These concepts also support migration from previous national standards.

3.4.22 token

right to transmit on the local link

NOTE 1 This right is assumed by a DLE when it activates its LAS functions. This right may be delegated to individual DLEs, subject to specified constraints on its usage. In all cases, this right ultimately reverts to the DLE which has activated its LAS functions (the LAS DLE). Each token is implicitly qualified by the method of its transmission or assumption:

- a) A **scheduler token** is assumed and held by the LAS DLE, and can be sent to another LM DLE on the local link to transfer the activation of LAS functions to that receiving DLE.
- b) A **delegated token** is created by the LAS DLE and sent to a DLE on the local link, and is returned upon completion of its use, or assumed by the LAS DLE at its expiration.

- c) A **reply token** is created by the current delegated token holder (or scheduler token holder if there is no delegated token holder) and sent to a DLE on the local link, requesting an immediate reply; it is returned with that immediate reply, or assumed by the current delegated token holder at the expiration of the reply period

NOTE 2 It is possible, though not absolutely required, for a DLE to function simultaneously as an LAS, an initiator, and a responder, delegating a token and receiving that token, requesting an immediate reply and receiving that request, and then replying to itself as requestor and returning the token to itself as LAS. This would require the transmission of at least three DLPDUs — one from each role.

3.4.23 token-recovery-delay

measure, similar to maximum-response-delay, greater than the worst-case period of **peer** PhE-inactivity which can be observed by the LAS DLE while another correctly-functioning DLE is using a token.

NOTE 1 More formally, token-recovery-delay provides a bound greater than the worst-case delay between receipt of an end-of-activity or end-of-data-and-activity PhIDU (see 6.4.4) and subsequent receipt of the next start-of-activity PhIDU, that can be observed by any DLE on the local link during normal link operation. This delay can be exceeded only when a DLE fails while holding a delegated or scheduler token.

NOTE 2 The value of token-recovery-delay, in units of one slot-time, may be any value between (maximum-response-delay + 3) and 14

3.4.24 transaction

single DLPDU, or a sequence of two immediately consecutive related DLPDUs, resulting from a single DLS-user request

NOTE 1 The DLE sending the first DLPDU of the transaction is known as the initiator; the DLE which sends the second DLPDU of the transaction, if any, is known as the responder.

NOTE 2 A DL-entity can be both an initiator and a responder in the same transaction.

3.5 Type 2: Additional terms and definitions

3.5.1 actual packet interval (API)

measure of how frequently a specific connection produces its Lpacket data

3.5.2 allocate

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

3.5.3 application

function or data structure for which data is consumed or produced

3.5.4 application objects

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

3.5.5 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.

3.5.6 behavior

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

3.5.7 bit

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

3.5.8 blanking or blanking time

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

3.5.9 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

3.5.10 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)

3.5.11 connection

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

3.5.12 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.

3.5.13 cyclic redundancy check (CRC)

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

3.5.14 cyclic

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

3.5.15 deafness

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

3.5.16 device

physical hardware connection to the link

NOTE A device may contain more than one node.

3.5.17 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.

3.5.18 end delimiter

unique set of M_symbols that identifies the end of a DLPDU

3.5.19 error

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

3.5.20 extended link

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

3.5.21 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

3.5.22 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

3.5.23 frame

denigrated synonym for DLPDU

3.5.24 generic tag

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

3.5.25 guardband

time slot allocated for the transmission of the moderator DLPDU

3.5.26 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

3.5.27 implicit token register

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

3.5.28 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.

3.5.29 instance attribute

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

3.5.30 instantiated

object that has been created in a device

3.5.31 keeper

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

3.5.32 link

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

3.5.33 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

3.5.34 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

3.5.35 M_symbol(s)

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

3.5.36 maximum scheduled node

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

3.5.37 maximum unscheduled node

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

3.5.38 message router

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

3.5.39 moderator

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

3.5.40 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

3.5.41 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.

3.5.42 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

3.5.43 network access monitor (NAM)

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

3.5.44 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

3.5.45 network address or node address

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

3.5.46 network status indicators

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

3.5.47 network update time (NUT)

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

3.5.48 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.

3.5.49 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

3.5.50 non-data symbol

PhL symbol which violates the requirements of Manchester biphase L encoding

3.5.51 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.

3.5.52 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.

3.5.53 originator

client responsible for establishing a connection path to the target

3.5.54 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

3.5.55 produce

act of sending data to be received by a consumer

3.5.56 producer

node that is responsible for sending data

3.5.57 redundant media

system using more than one medium to help prevent communication failures

3.5.58 requested packet interval (RPI)

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

3.5.59 rogue

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

3.5.60 scheduled

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

3.5.61 server

object which provides services to another (client) object

3.5.62 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.

3.5.63 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

3.5.64 start delimiter

unique set of M_symbols that identifies the beginning of a DLPDU

3.5.65 supernode

DLE with a MAC ID of zero

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

3.5.66 table unique identifier (TUI)

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

3.5.67 tag

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

3.5.68 target

end-node to which a connection is established

3.5.69 temporary node

same as transient node

3.5.70 tMinus

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

3.5.71 tone

instant of time which marks the boundary between two NUTs

3.5.72 tool

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

EXAMPLE link scheduling software (LSS).

3.5.73 transaction id

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

3.5.74 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

3.5.75 unconnected message manager (UCMM)

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

3.5.76 unconnected service

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

3.5.77 unscheduled

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

3.6 Type 3: Additional terms and definitions

3.6.1 acknowledge DLPDU

reply DLPDU that contains no DLSDU

3.6.2 address extension

DLSAP address or region/segment address

3.6.3 bit time

time to transmit one bit

3.6.4 confirmed message exchange

complete data transfer with request and acknowledgement or response DLPDU

3.6.5 controller_type

hardware class of the communications entity

3.6.6 current master

token holder

3.6.7 data DLPDU

DLPDU that carries a DLSDU from a local DLS-user to a remote DLS-user

3.6.8 DL_status

status that specifies the result of the execution of the associated request

3.6.9 GAP

range of station (DLE) DL-addresses from this station (TS) to its successor (NS) in the logical token ring, excluding stations above HSA

3.6.10 GAP maintenance

registration of new Master and Slave stations

3.6.11 isochronous mode

special operational mode that implies both a constant (isochronous) cycle with a fixed schedule of high and low priority messages, and the synchronization of the DLS-users with this constant (isochronous) cycle

3.6.12 local DLS-user

DLS-user that initiates the current service

3.6.13 message exchange

complete confirmed or unconfirmed data transfer

3.6.14 region/segment address

address extension that identifies a particular fieldbus subnetwork

NOTE This supports DL-routing between fieldbusses.

3.6.15 request data

DLSDU provided by the remote DLS-user to the local DLS-user

3.6.16 remote DLE

addressed DLE of a service request (that is, the intended receiving DLE of any resulting send/request DLPDU)

3.6.17 remote DLS-user

addressed DLS-user of a service request (that is, the intended receiver of any resulting indication primitive)

3.6.18 reply DLPDU

DLPDU transmitted from a remote DLE to the initiating (local) DLE, and possibly other DLEs

NOTE When the remote DLE is a Publisher, the reply DLPDU also can be sent to several remote DLEs.

3.6.19 response DLPDU

reply DLPDU that carries a DLSDU from a remote DLS-user to local DLS-user

3.6.20 send data

DLSDU provided by a local DLS-user to a remote DLS-user

3.6.21 send/request DLPDU

DLPDU that carries either a request for data or a DLSDU or both from a local DLS-user to a remote DLS-user

3.6.22 time master

device which is able to send clock synchronization DLPDUs

NOTE Link devices have time master functionality.

3.6.23 time receiver

device which is able to be time synchronized by a time Master

3.6.24 token holder

Master station that controls bus access

3.6.25 token passing

medium access method, in which the right to transmit is passed from master station to master station in a logical ring

3.7 Type 4: Additional terms and definitions

3.7.1 broadcast-Node-address

address used to send broadcasts to all DLEs on a Link

NOTE All DLEs on a Link receive all DLPDUs where the first Node-address is equal to the Broadcast-Node-Address. Such DLPDUs are always Unconfirmed, and their receipt is never acknowledged. The value of a Broadcast-Node-address is 126.

3.7.2 destination-DL-route

holds a sequence of DL-route-elements, describing the complete route to the destination

NOTE This includes both the destination DLSAP and a local component meaningful to the destination DLS-user.

3.7.3 DL-route

combination of a Destination-DL-route and a Source-DL-route

3.7.4 DL-route-element

octet holding a Node-address or an address used by the DLS-user

3.7.5 no-Confirm-Node-address

address used to indicate that a request or response is Unconfirmed

NOTE The value of a No-Confirm-Node-address is 0.

3.7.6 node-address

address which uniquely identifies a DLE on a Link

NOTE The value of a Node-address can be in the range of 0 to 127, with the values 0, 126 and 127 reserved for special purposes.

3.7.7 normal class device

device which replies to requests from other normal class devices, and initiates transmissions

NOTE Such a device can act as a server (responder) and as a client (requestor) - this is also called a peer

3.7.8 P-NET-route

holds a sequence of P-NET-route-elements

NOTE A P-NET-route is defined as an encoded DL-route, with one of the formats used when transmitting the DLPDU on the Link. The P-NET-route format can be Simple, Extended, Complex or Immediate.

3.7.9 P-NET-route-element

an octet, holding a 7-bit DL-route-element or Remaining-route-length, and a 1-bit source/destination designator

3.7.10 service-Node-address

address reserved for service purposes only

NOTE All DLEs on a Link receive all DLPDUs where the first Node-address is equal to the Service-Node-Address. Such DLPDUs can be Confirmed or Unconfirmed, and their receipt may or may not be acknowledged. The Service-Node-Address can be used on Links with only two DLEs - the requesting Normal class DLE and the responding Simple or Normal class DLE. The value of the Service-Node-Address is 127.

3.7.11 source-DL-route

holds a sequence of DL-route-elements, describing the complete route back to the source

3.7.12 simple class device

device which replies to requests from normal class devices, and can act as a server or responder only

3.7.13 virtual Link-access token

basis for the Link-access system

NOTE It is called Virtual because the token is not explicitly sent from one Normal class DLE to another, but implicitly passed as the Link is idle.

3.8 Type 5: Additional terms and definitions

A normative specification of this protocol was not available at the time of compilation of this edition. See 3.4 for a preliminary set of additional terms and definitions.

3.9 Type 6: Additional terms and definitions

3.9.1 Additional definitions related to IEEE Std 1451.2

3.9.1.1 Network Capable Application Processor (NCAP)

processor in a smart field device

3.9.1.2 Smart Transducer Interface Module (STIM)

"device" that conforms to IEEE Standard 1451.2

3.9.1.3 transducer

sensor or actuator

3.9.1.4 Transducer Electronic Data Sheet (TEDS)

non-volatile memory region that stores the configuration of a smart field device.

3.9.1.5 trigger

signal from a smart NCAP that can trigger synchronized internal or external actions on the local or extended link

3.9.1.6 Universal Unit Identifier (UUI)

12-octet "globally – world-wide" unique identifier for a DLE, based on the 3-dimensional-geographic-location and time of manufacture of the referenced DLE

NOTE Its value is an inherent property of the DLE and is stored in the DLE's TEDS.

3.9.2 Variations on the common definitions of 3.1 through 3.4

3.9.2.1 bridge, DL-relay

see 3.4.1

3.9.2.2 DLCEP-address

see 3.4.2.

3.9.2.3 DLC-ID, channel-ID

identifies one Channel (DLC) from within all Channels on a local link

NOTE The Channel-ID is the number of the first slot (after the Bus-Cycle-End slot) assigned to this channel.

3.9.2.4 peer DLC, GP channel

point-to-point DL-connection offering DL-duplex-transmission between two peer DLS-users where each can be a Calling DLS-user and each can be a Called DLS-user.

NOTE A peer DLC is configured to provide either Confirmed or Acknowledged service.

3.9.2.5 multi-peer DLC

multi-end-point DL-connection offering DL-simplex-transmission between a single distinguished DLS-user, known as the **publisher** or **publishing** DLS-user, and a set of one or more peer distinguished DLS-users, known collectively as the **subscribers** or **subscribing** DLS-users, where the publishing DLS-user can send to the subscribing DLS-users as a group (but not Individually)

NOTE A multi-peer DLC always provides DL-simplex service, from the publisher to the subscribers.

3.9.2.6 master DLCEP

peer DLCEP which controls the direction of data transfers on a GPC channel

3.9.2.7 slave DLCEP

peer DLCEP which may only initiate data transfers on a GPC channel when authorized by the Master peer DLCEP

3.9.3 Additional definitions

3.9.3.1 1WAY method

channel which can only support a one-way flow of data

3.9.3.2 2WAY method

channel which can support a two-way flow of data

3.9.3.3 active nodes and channels

nodes and channels that are able to communicate on the local link

NOTE Active nodes have No-Comm-Node equal 0. Active channels have No-Comm-Node and No-Comm-Channel both equal 0.

3.9.3.4 aliasing

any difference between the time stamps placed on an Event by two different DLEs

3.9.3.5 assembly (reassembly)

process of combining separate grains in the proper order to recreate a DLSDU as it was presented at the remote sending DLS

3.9.3.6 log bus length

base-2 logarithm of the number of slots in one Bus-cycle

3.9.3.7 BL ≠ BSS

referenced parameter can take on all values from 0 up to Bus Length, but excluding all Bus-Sync-slot numbers

3.9.3.8 bogus DLE

DLE found by the Conductor during startup that is not listed on the Node List

3.9.3.9 buffer-queue

memory area used for communicating DLS-user-data between the DLS-user and the DLE in a local node for one direction of a channel, which may take the form of either an overwriteable buffer or a FIFO (first-in first-out) limited-depth queue

3.9.3.10 bus config

dynamic set of values that determine the timing of the bus in one installation

3.9.3.11 bus configuration

dynamic set of values of Bus-Config, node, channel, trigger and forwarding-entity parameters that applies to one installation

3.9.3.12 bus-configuration DLMS-user

System Management entity which configures each DLE on the local link by exchanging DLMSDUs with the DLE's associated Node-configuration DLMS-user

3.9.3.13 bus-cycle

uniform progression of slots from 0 to the Bus-Cycle-Length

NOTE Each active DLE maintains a sense of the current slot (a counter), which it uses to synchronize all real-time actions.

3.9.3.14 bus-cycle-length

number of slots in a Bus-Cycle

NOTE It is always a power of 2 equal to 2^{LBL} . It's value is a configuration issue.

3.9.3.15 bus-end-sync-slot

last or "cycle-closing" slot of a bus cycle

NOTE The Conductor uses it to synchronize the bus cycle counter of all other active DLEs with the Conductor's own bus cycle.

3.9.3.16 bus-sync-period

number of slots between bus sync slots

3.9.3.17 bus-sync-slot

variable duration slot used by the Conductor to fine tune synchronization of the slot clocks of all other active DLEs with the Conductor's slot-clock

NOTE The first Bus-Sync-slot is at Bus-Sync-Period slots.

3.9.3.18 bus-time

common sense of communications system time that is shared by all DLEs on the extended link

3.9.3.19 called DLS-user

higher-layer entity that accepts the transfer of DLS-User-Data

3.9.3.20 calling DLS user

higher-layer entity that initiates the transfer of DLS-User-Data

3.9.3.21 CD-semaphore

interlocked signaling mechanism that controls which DLS-user can transmit next on a GPC channel, that is, which specifies the Channel Direction

3.9.3.22 channel (DLC)

set of equally spaced slots per bus cycle that conveys either:

- a) measurements or control signals that can be received by one or more DLS-users using the SCAN or ExSCAN channel classes, or
- b) interactions between two specific DLS-users using the GPC or GPA channel classes, or
- c) messages between DLMS-users using the DLM-connectionless service

3.9.3.23 channel-class

collection of channel-methods that together define the total operation and QoS of a channel

3.9.3.24 channel direction method

attribute that determines the allowed direction(s) of data flow on the channel

3.9.3.25 channel method (QoS building block)

attribute that determines one aspect of the fundamental operation of a channel, applying to all the slots of one channel on a local or extended link

3.9.3.26 channel selector

identifies any slot during which data for a particular channel is or should be present on the local link

3.9.3.27 conductor

the DLE that is synchronizing all traffic on a bus, and which may also configure all DLEs and channels on that bus

NOTE The Conductor does not schedule traffic as does a Type 1 Link Active Scheduler (LAS).

3.9.3.28 configuration database

contains the configured values of the DLE parameters for each DLE channel, trigger and half-forwarding-port for one usage (experiment/batch/process) of the bus

3.9.3.29 configuration object

configurable object in a non-conductor node

NOTE Configuration-objects include channels, triggers and half-forwarding-ports.

3.9.3.30 configuration results-list

list of the results of the configuration of each node and each configuration object

3.9.3.31 configured nodes

DLEs that have been successfully configured per the Node List

3.9.3.32 conformance-class

fixed set values of parameters that are defined as an integrated set and which apply to multiple installations

3.9.3.33 connection-oriented service

service provided between any configured pair of peer DLCEPs. or from one configured DLCEP to any configured group of multi-peer DLCEPs

3.9.3.34 data

generic term used to refer to any information carried over a fieldbus

3.9.3.35 data field

atomic DLSDU carried by the UNITARY method

3.9.3.36 data-packaging method

attribute that determines whether DLSDUs sent on the channel will be granulated or not

3.9.3.37 device

DLE which cannot become a Host or Conductor

3.9.3.38 destination DLSAP

DLSAP that accepts the transfer of a message using the DLM-connectionless method

3.9.3.39 directory

human and machine-readable file that gives the identity and functional usage/location of each Node and Channel on a single local link or an extended link

3.9.3.40 DL-buffer identifier

local identifier for the data passed in a buffer by the associated primitive, shared by the DLS provider and DLS-user

3.9.3.41 DL-event-time

time at which an Event occurred, as calculated by the Conductor

3.9.3.42 DL-queue identifier

local identifier for the data passed in a queue by the associated primitive, shared by the DLS provider and DLS-user, or by the DLMS provider and DLMS-user

3.9.3.43 DLC-identifier

Identifier for one Channel (DLC) from within all Channels on a local link

NOTE The value of the Channel-ID is equal to the relative index of the first slot (after the Bus-cycle-End slot) assigned to this channel.

3.9.3.44 DLM-connectionless DLPDU

DLPDU transmitted using DLM-connectionless procedures and format

3.9.3.45 DLM-connectionless service

service used for bus configuration with less integrity than GPC or GPA connection mode services, in which the DLS does not relate any DLM-connectionless DLPDU to any other DLPDU or make any retry attempt on detection of an error

3.9.3.46 DLMS-user

user of the DL-Management service, generally, of the DLM bus configuration service

3.9.3.47 DLMSDU

service data unit sent or received by a DLMS-user, generally, with structure and content specified by this standard

3.9.3.48 DLPDU user-data field

DLPDU field which conveys all or part (one grain) of a DLSDU

3.9.3.49 DLSAP role

role a DLSAP plays in the current transaction

3.9.3.50 DLSDU length

length in octets of an item of DLS-user-data

3.9.3.51 EXSCAN class

channel capable of transferring long, scanned, state data, whose length exceeds Max-data-length, with minimal DL-overhead

3.9.3.52 forwarding-bridge-DLE

DLE that synchronizes the bus cycles and slot clocks for two or more connected local links, and which can be configured to forward DLSDUs among the connected links

3.9.3.53 functional DLEA

group DLEA that identifies a DLE that is now performing a specific function (such as the active Conductor)

3.9.3.54 general purpose acknowledged (GPA) class

channel which provides reliable communications for higher layer protocols that depend on classical one-way DL-data-delivery

3.9.3.55 general purpose confirmed (GPC) class

channel which provides reliable communications for higher layer protocols that depend on classical two-way DL-data-delivery

3.9.3.56 grain

partial DLSDU

NOTE One of the elements of granulated data that comprise a DLSDU on a GPC or GPA channel. The order of transmission of grains is, the Initial-grain, zero or more Middle-grains, followed by the Final-grain.

3.9.3.57 GRANULAR channel

channel which can perform Granulation and Assembly

3.9.3.58 GRANULAR method

channel on which each DLSDU is transferred in one or more grains

NOTE When the Calling DLS-user on a GRANULAR channel submits DLSDUs longer than the data "payload" of the DLPDUs now in use on this bus (that is, the DLSDU-length exceeds the configured max-data-length of the bus), the DLE **granulates** (segments) each DLSDU into multiple DLPDUs. Each grain carries a one octet header that allows reconstruction by the called DLE of the original submitted DLSDU. The receiving DLE assembles the grains and validates the DLSDU before delivery to the Called DLS-user.

3.9.3.59 granulation (segmentation)

see GRANULAR method

3.9.3.60 high channel count architecture

DLE architecture that is typically used for a Host, typically with a large number of DLSAPs

3.9.3.61 high order part of a data item

most significant (written as leftmost) octets/bits of the item

3.9.3.62 host

DLE that can become a Conductor and synchronize operation of a bus, and which may source and/or sink "data" on a local or extended link

3.9.3.63 I/O channel

hardware interface to a "real world" transducer

NOTE This concept maps directly onto the "data transfer" channels of this protocol.

3.9.3.64 jitter

any deviation from a regular synchronous pattern in either

- a) sampling a transducer
- b) transmission of data values for the SCAN and EXSCAN classes
- c) imposition of a trigger signal.

NOTE Jitter can be measured both from instant to instant within one Node and across multi-nodes on a local or extended link.

3.9.3.65 local DLSAP

DLSAP that initiates the transfer of a DLSDU using the DLM-connectionless method

3.9.3.66 log scan-class

$\log_2$ of the number of slots between slot selectors, as used for channel configuration

3.9.3.67 low channel count architecture

DLE architecture that is typically used for sensors and actuators with one or a few channels, typically using a small number of DLSAPs

3.9.3.68 low-order part of a data item

least significant (written as rightmost) octets/bits of the item

3.9.3.69 max-data-length

maximum length of the data field of any DLPDU which may be transmitted on an extended link

NOTE max-data-length is an element of Bus-Config

3.9.3.70 max-DLSDU-buffer-size

maximum buffer size for the conformance-class in use on a bus

3.9.3.71 max-DLSDU-length

maximum length of any DLSDU which may be transmitted on a bus

3.9.3.72 max-DLE-group-address

maximum number of group DLE addresses per local link

3.9.3.73 max-net-level

maximum depth of the nest of real links

3.9.3.74 max-time-stamp

maximum number of time-stamp buffers and messages that can be present in one DLSAP

3.9.3.75 missing node

DLE that is listed on the Node-List that cannot communicate on the local link

3.9.3.76 max-trigger

maximum number of triggers that can be present in one DLSAP

3.9.3.77 net-ID, link-address

hierarchical identification of the subnetwork on which a DLE resides in the real network topology, identifying the local link as seen from the TOP level (L0) of the extended link looking DOWN, used by DLM-connectionless services

3.9.3.78 node

generic term for a DLE, sometimes also including its associated DLS- or DLMS-user

3.9.3.79 node-configuration DLMS-user

local System Management agent which configures a DLE according to DLMSDUs received from the Bus-configuration DLMS-user

3.9.3.80 node-list

list that contains the bus configuration information for each node, channel, half-forwarding-port and trigger which should be active on the local link

3.9.3.81 non-retentive buffer (queue of depth one)

buffer (equivalently, a queue of depth one) for which DLSDU (or DLMSDU) presence is cleared immediately after the contained DLS-user-data has been transmitted by the DLE or read by the DLS-user (or DLMS-user)

3.9.3.82 num-time-stamp

number of time-stamp-buffers present in a DLSAP

3.9.3.83 offset slot, channel-ID

starting slot number for scanning the specified channel

3.9.3.84 DLPDU-length

actual length of the contents of the DLPDU

3.9.3.85 publisher DLCEP

DLCEP that attempts to transmit a DLPDU in each slot assigned to a specific channel on a multi-peer DLC

3.9.3.86 QoS-set

selector that identifies, for one DLCEP, a specific set of QoS options selected from those specified for the conformance-class in use, thereby implicitly restricting the set of DLCEPs that can support a specific DLS-user in a matching role, that is, those DLCEPs that can participate in a specified DLS-user protocol

3.9.3.87 retentive-buffer

buffer for which transmission or reading of the DLSDU does not cause the DLE to empty the buffer, where the sending DLS-user clears the buffer by writing to it and the receiving DLE overwrites the value in the buffer when it receives a new DLSDU

3.9.3.88 remote DLSAP

DLSAP that receives a DLSDU using the DLM-connectionless method

3.9.3.89 SCAN channel

channel which transfers short, scanned, state data with minimal DL-overhead

3.9.3.90 scan period (of a channel)

instance to instance increment in the slot index of slots assigned to the channel

3.9.3.91 scan rate

number of slots per second assigned to a channel

3.9.3.92 short DLSDU

DLSDU whose length is known by configuration to be less than or equal to max-data-length, so that it can always be conveyed by a single DLPDU

3.9.3.93 slot

basic unit of time allocation on a bus

NOTE All slots are the same lengths for a given (changeable) bus configuration.

3.9.3.94 source DLSAP

DLSAP that initiates the transfer of a message using the DLM-connectionless method

3.9.3.95 subscriber DLCEP

DLCEP that attempts to receive the DLPDUs present in the slots assigned to a specific channel on a multi-peer DLC

3.9.3.96 synchronizer DLE

DLE that synchronizes the bus cycles and slot clocks between two or more local links, but does not support data transfers between local links (and thus is not also a Bridge DLE)

3.9.3.97 system

extended link with all of its attached DLEs

3.9.3.98 system manager

agent which sets up the system configuration in the node list and monitors configuration and operation of the system

3.9.3.99 time stamp

attribute of a datum which provides the bus time at which the underlying event was detected by the submitting DLS-user

3.9.3.100 time-stamping of events

recording a time-stamp concurrent with an external signal and appending the recorded time-stamp to a later transmitted Event message

NOTE The time stamp appended to the message may be approximately concurrent with the time the relevant Event occurred, even if the Event message is transmitted much later.

3.9.3.101 transducer

sensor or actuator

3.9.3.102 transfer DLPDU

DLPDU that carries all of, or a grain of, a DLSDU, that is, a data DLPDU on a DLC

3.9.3.103 transmission delay

any unwanted delay in transmitting transducer readings

3.9.3.104 transparent data conveyance

guarantee that for each DLPDU transmitted, the called DLS-user receives either an error indication or exactly the “data” submitted by the calling DLS-user

3.9.3.105 UNITARY method

channel on which a complete DLSDU is always transferred in a single Transfer DLPDU

3.9.3.106 VID (of a DLE)

externally visible identification of a DLE and its contained objects, identical to the DLE's DLEA and used by DLM-connectionless services

3.9.3.107 virtual topology

connection mode topology assigned by the Bus-configuration DLMS-user, which need not map directly to physical topology

3.10 Type 7: Additional terms and definitions**3.10.1 acknowledgement response DLPDU**

information that the recipient of an acknowledged message emits in order to signal either the proper reception of the message or the lack of available resources to store the message, received by the DLE on the local link that emitted the message which requested the acknowledgement

3.10.2 basic cycle

sequence of scanning by the Bus Arbitrator of:

- a) a set of DLCEP-identifiers for variables, requests, and cyclical application messages,
- b) plus the window provided for aperiodic exchanges,
- c) plus the window provided for message services,
- d) plus the window provided for synchronization

3.10.3 basic transaction

succession of DLPDUs related to a single DL-service instance

3.10.4 Bus Arbitrator (BA)

DLE that controls each data producer's right to access the medium

NOTE At any given instant one and only one Bus Arbitrator is active in each DL-segment of a DL-subnetwork.

3.10.5 consumed identified variable

identified variable that corresponds to a DLCEP-identifier for which the entity in question receives data

3.10.6 control field

portion of an emitted or received DLPDU that gives the nature of the data exchanged and the type of exchange

3.10.7 destination address

three octets specifying the DL-segment of the DLE to whom the message is sent, and the destination DLSAP's sub-address within the local link DL-segment

3.10.8 DLCEP-address

information that the Bus Arbitrator emits to allocate the medium to a data producer for the purpose of exchanging a variable

3.10.9 end of message transaction indication DLPDU

information that the source entity of a message emits in order to return link access control to the Bus Arbitrator at the end of a message transaction

3.10.10 identified variable (or simply "variable")

DLL variable (buffer) for which an associated DLCEP-identifier has been defined

3.10.11 identifier

16-bit word associated with a system variable. A DLCEP-identifier uniquely designates a single variable within the DL-subnetwork

3.10.12 invalid DLCEP-identifier

identifier not recognized locally

3.10.13 local link

set of devices that respect the DL-protocol and that are interconnected through a medium . Only one Bus Arbitrator is active on a single local link

3.10.14 macrocycle

set of basic cycles needed for all cyclical DLCEP-identifiers to be scanned

3.10.15 message DL-address

information that the Bus Arbitrator emits to allocate the medium to a source entity for a message transfer

3.10.16 message response DLPDU

information that a data producer emits in response to a message DLCEP-identifier DLPDU

NOTE The desired destination entity or entities pick up this information.

3.10.17 periodic scanning of variables

action by the Bus Arbitrator that guarantees the cyclical exchange of variables

NOTE This is the basic principle of the Type 7 DL-service and protocol.

3.10.18 produced identified variable

identified variable that corresponds to a DLCEP-identifier for which the DLE emits data

3.10.19 request DLCEP-address

information that the Bus Arbitrator emits to allocate the medium to the initiator of an explicit request for a variable exchange

3.10.20 request response DLPDU

information that the initiator of an explicit request for a variable exchange emits in response to a request DLCEP-identifier DLPDU, and to whose transmittal the Bus Arbitrator also responds

3.10.21 source address

24-bit word including the DL-segment number of the entity sending the message, and the entity's sub-address within the DL-segment

3.10.22 timers

see 29.6

3.10.23 triggered message scanning

function of a Bus Arbitrator that makes it possible to transfer messages

3.10.24 triggered scanning of variables

function of a Bus Arbitrator that makes possible the non-cyclical exchange of variables

3.10.25 triggered periodic scanning of messages

function of a Bus Arbitrator that makes it possible to request triggered message exchanges cyclically

3.10.26 triggered periodic scanning of variables

function of a Bus Arbitrator that makes it possible to request triggered variable transfers cyclically

3.10.27 turnaround time

time interval between reception or emission of the last MAC symbol of a DLPDU, signaled by a SILENCE indication from the PhL, and the reception or emission of the first MAC symbol of the subsequent DLPDU, signaled by an ACTIVITY indication from the PhL, both as measured in a given station

3.10.28 variable response DLPDU

information that a data producer emits in response to a DLCEP-identifier DLPDU, which also alerts data consumers to the relevance of the immediately time-proximate DLPDU

3.11 Type 8: Additional terms and definitions**3.11.1 bus coupler**

PhL entity which includes or excludes Ph-segments into or from the network

3.11.2 device

slave or master

3.11.3 device code

two octets which characterize the properties of a slave

3.11.4 DLPDU cycle

transaction initiated from the master in which user data or identification/status information is sent to all slaves and – within the same cycle - received from all slaves

3.11.5 IN data

data received by the master and sent by the slaves

3.11.6 master

DL-entity controlling the data transfer on the network and initiating the media access of the slaves by starting the DLPDU cycle

3.11.7 OUT data

data sent by the master and received by the slaves

3.11.8 parameter channel

acyclic transmission path using a client/server communication model

3.11.9 process data channel

conveyance path allowing a very efficient, high-speed and cyclic transmission of process-relevant data, between slaves and master

3.11.10 receive update memory

memory area containing the data, which was received from the network

3.11.11 ring segment

group of slaves in consecutive order

3.11.12 ring segment level

nesting level number of a ring segment

3.11.13 slave

DL-entity accessing the medium only after being initiated by the preceding slave or master

3.11.14 transmit update memory

memory area containing the data to be sent across the network

3.11.15 update time

time which passes between two consecutive starts of DLPDU cycles used for data transfer

4 Symbols and abbreviations

4.1 Common symbols and abbreviations

4.1.1 Data units

4.1.1.1 DLPDU DL-protocol data unit

4.1.1.2 DLSDU DL-service data unit

4.1.1.3 PhIDU Ph-interface data unit

4.1.2 Miscellaneous

4.1.2.1 DL- data link layer (as a prefix)

4.1.2.2 DLCEP DL-connection endpoint

4.1.2.3 DLE DL-entity (the local active instance of the Data Link layer)

4.1.2.4 DLL DL-layer

4.1.2.5 DLM- DL-management (as a prefix)

4.1.2.6 DLMS DL-management-service

4.1.2.7 DLS DL-service

4.1.2.8 DLSAP DL-service access point

4.1.2.9 FCS frame check sequence

NOTE For Type 3, this also means "frame checksum"

4.1.2.10 FIFO first-in first-out (queuing method)

4.1.2.11 LLC logical link control

4.1.2.12 MAC medium access control

4.1.2.13 OSI open systems interconnection

4.1.2.14 Ph- physical layer (as a prefix)

4.1.2.15 PhE Ph-entity (the local active instance of the Physical layer)

4.1.2.16 PhL Ph-layer

4.1.2.17 PhS Ph-service

4.1.2.18 PhSAP Ph-service access point

4.1.2.19 QoS quality of service

4.2 Type 1: Additional symbols and abbreviations

4.2.1 Data units

4.2.1.1 BPDU bridge protocol data unit, used in the ISO/IEC 10038 inter-bridge protocol

4.2.1.2 pDLSDU partial DL-service data unit — one segment of a multi-segment DLSDU

4.2.1.3 SPDU support protocol data unit, used to support the full DL-protocol

4.2.2 Local variables, timers, counters and queues

4.2.2.1	V(ST)	slot time	See 6.7.1.1
4.2.2.2	V(PhLO)	per DLPDU PhL overhead	See 6.7.1.2
4.2.2.3	V(MRD)	maximum response delay	See 6.7.1.3
4.2.2.4	V(IRRd)	immediate response recovery delay	See 6.7.1.4
4.2.2.5	V(MRC)	maximum retry count	See 6.7.1.5
4.2.2.6	V(NRC)	network retry count	See 6.7.1.6
4.2.2.7	V(NDL)	network DLPDU lifetime	See 6.7.1.7
4.2.2.8	V(TN)	this node	See 6.7.1.8
4.2.2.9	V(TL)	this link	See 6.7.1.9
4.2.2.10	V(MEP)	DL MAC address embedding prefix	See 6.7.1.10
4.2.2.11	C(RD)	remaining duration down-counter	See 6.7.1.11
4.2.2.12	V(MID)	minimum inter-DLPDU delay	See 6.7.1.12
4.2.2.13	T(IRRd)	immediate response recovery delay monitor	See 6.7.1.13
4.2.2.14	V(RA)	reply address	See 6.7.1.14
4.2.2.15	V(OTA)	outstanding transaction array	See 6.7.1.15
4.2.2.16	V(LTI)	last transaction index	See 6.7.1.16
4.2.2.17	Q(US)	unscheduled service queue	See 6.7.1.17
4.2.2.18	V(RID)	random identifier	See 6.7.1.18
4.2.2.19	C(NT)	node time up-counter	See 6.7.1.19
4.2.2.20	V(LSTO)	local link scheduling time offset	See 6.7.1.20
4.2.2.21	V(DLTO)	DL-time offset	See 6.7.1.21
4.2.2.22	V(TQ)	time quality	See 6.7.1.22
4.2.2.23	V(MD)	measured delay	See 6.7.1.23
4.2.2.24	V(LN)	LAS node	See 6.7.1.24
4.2.2.25	V(TSC)	time synchronization class	See 6.7.1.25
4.2.2.26	T(TDP)	time distribution period monitor	See 6.7.1.26
4.2.2.27	V(TSL)	time source link	See 6.7.1.27
4.2.2.28	P _U (SDUL)	DLSDU length request parameter	See 6.7.2.1
4.2.2.29	P _U (SDU)	DLSDU request parameter	See 6.7.2.2
4.2.2.30	P _U (MCD)	maximum confirm delay parameter	See 6.7.2.3
4.2.2.31	T _U (MCD)	maximum confirm delay monitor	See 6.7.2.4
4.2.2.32	Q _A (UR)	user request queue	See 6.7.3.1

4.2.2.33	V_C(ST)	DLCEP state	See 6.7.4.1
4.2.2.34	V_C(NP)	negotiated DLCEP parameters	See 6.7.4.2
4.2.2.35	V_C(N)	next sequence number to assign to a DLSDU	See 6.7.4.3
4.2.2.36	V_C(R)	maximum non-transmittable DLSDU sequence number	See 6.7.4.4
4.2.2.37	V_C(A)	maximum acknowledged DLSDU sequence number	See 6.7.4.5
4.2.2.38	V_C(M)	minimum untransmitted DLSDU sequence number	See 6.7.4.6
4.2.2.39	V_C(MS)	minimum untransmitted segment number	See 6.7.4.7
4.2.2.40	V_{C,K}(SS)	to-be-sent segments of a DLSDU	See 6.7.4.8
4.2.2.41	T_{C,K}(SS)	sent segments monitor for a DLSDU	See 6.7.4.9
4.2.2.42	T_C(SS)	simplified sent segments monitor for a DLSDU	See 6.7.4.9.1
4.2.2.43	V_C(L)	last reported DLSDU sequence number	See 6.7.4.10
4.2.2.44	V_C(H)	highest detected DLSDU sequence number	See 6.7.4.11
4.2.2.45	V_C(HS)	highest detected segment number of the highest detected DLSDU sequence number	See 6.7.4.12
4.2.2.46	V_{C,K}(MRS)	missing received segments of a DLSDU	See 6.7.4.13
4.2.2.47	V_{C,K}(RRS)	retransmission request required segments of a DLSDU	See 6.7.4.14
4.2.2.48	T_{C,K}(RRS)	retransmission request monitor for a DLSDU	See 6.7.4.15
4.2.2.49	T_C(RAS)	residual activity stimulus	See 6.7.4.16
4.2.2.50	T_C(RAM)	residual activity monitor	See 6.7.4.17
4.2.2.51	V_C(TNA)	DL-time of last network access	See 6.7.4.18
4.2.2.52	V_B(TW)	DL-time of last buffer write	See 6.7.4.19
4.2.2.53	V_B(TP)	DL-time of production	See 6.7.4.20
4.2.2.54	V_B(TS)	timeliness status of buffer write	See 6.7.4.21
4.2.2.55	V(DTA)	delegation address	See 6.7.5.1
4.2.2.56	V(LL)	local link live list	See 6.7.5.2
4.2.2.57	V(TCL)	token circulation list	See 6.7.5.3
4.2.2.58	V(ENRL)	expected non-response list	See 6.7.5.4
4.2.2.59	V(MST)	maximum scheduled traffic	See 6.7.5.5
4.2.2.60	V(MSO)	maximum scheduling overhead	See 6.7.5.6

4.2.2.61	V(DMDT)	default minimum token delegation time	See 6.7.5.7
4.2.2.62	V(DTHT)	default token holding time	See 6.7.5.8
4.2.2.63	V(LTHT)	link maintenance token holding time	See 6.7.5.9
4.2.2.64	V(MTHA)	maximum token holding time array	See 6.7.5.10
4.2.2.65	V(TTRT)	target token rotation time	See 6.7.5.11
4.2.2.66	V(ATRT)	actual token rotation time	See 6.7.5.12
4.2.2.67	V(RTHA)	remaining token holding time array	See 6.7.5.13
4.2.2.68	V(NTHN)	next token holding node	See 6.7.5.14
4.2.2.69	V(FUN)	first unpolled node id	See 6.7.5.15
4.2.2.70	V(NUN)	number of consecutive unpolled node ids	See 6.7.5.16
4.2.2.71	P(TRD)	token recovery delay	See 6.7.5.17
4.2.2.72	V(TDP)	time distribution period	See 6.7.5.18
4.2.2.73	V(MICD)	maximum inactivity to claim LAS delay	See 6.7.5.19
4.2.2.74	V(LDDP)	LAS data base distribution period	See 6.7.5.20
4.2.2.75	V(ML)	maximum link	See 6.7.6.1
4.2.3 DLPDU classes			
4.2.3.1	CA	compel acknowledgment	See 8.4
4.2.3.2	CD	compel data	See 8.5
4.2.3.3	CL	claim LAS	See 8.19
4.2.3.4	CT	compel time	See 8.9
4.2.3.5	DC	disconnect connection	See 8.2
4.2.3.6	DT	data	See 8.7
4.2.3.7	EC	establish connection	See 8.1
4.2.3.8	ED	exchange data	See 8.6
4.2.3.9	ES	execute sequence	See 8.16
4.2.3.10	IDLE	idle	See 8.22
4.2.3.11	PN	probe node address	See 8.13
4.2.3.12	PR	probe return	See 8.14
4.2.3.13	PT	pass token	See 8.15
4.2.3.14	RC	reset connection	See 8.3
4.2.3.15	RI	request interval	See 8.18
4.2.3.16	RQ	round trip delay query	See 8.11
4.2.3.17	RR	round trip delay reply	See 8.12

4.2.3.18	RT	return token	See 8.17
4.2.3.19	SR	status response	See 8.8
4.2.3.20	TD	time distribution	See 8.10
4.2.3.21	TL	transfer LAS	See 8.20
4.2.3.22	WK	wakeup	See 8.21

4.2.4 Miscellaneous

4.2.4.1	BME	bridge management entity	
4.2.4.2	DLC	DL-connection	
4.2.4.3	DLP-	DL-protocol (as a prefix)	
4.2.4.4	DLPCI	data link protocol control information	
4.2.4.5	DLSEP	DL-schedule endpoint	
4.2.4.6	FC	frame control (first octet of each DLPDU)	
4.2.4.7	FDC	fractional duty cycle (type of DLE)	
4.2.4.8	LAS	link active scheduler	
4.2.4.9	LM	link master	
4.2.4.10	PhICI	physical interface control information	
4.2.4.11	PhID	physical interface data	
4.2.4.12	SLAE	systems load application entity	
4.2.4.13	SMAE	systems management application entity	

4.3 Type 2: Additional symbols and abbreviations

4.3.1	ACM	access control machine	
4.3.2	CA	clock accuracy	
4.3.3	COP	connection originator password	
4.3.4	LED	light emitting diode.	
4.3.5	LSS	link scheduling software	
4.3.6	MAC ID	the address of a node	
4.3.7	NAM	network attachment monitor	
4.3.8	NAP	network access port	
4.3.9	ND	non-data symbol	
4.3.10	NUT	network update time	
4.3.11	PT	programming terminal (a temporary network connection)	
4.3.12	Rcv	receive	
4.3.13	RPI	requested Lpacket interval	

4.3.14 Rx	receive
4.3.15 RxLLC	receive logical link control
4.3.16 RxM	receive machine
4.3.17 SEM	state event matrix
4.3.18 SMAX	MAC ID of the maximum scheduled node
4.3.19 STD	state transition diagram, used to describe object behavior
4.3.20 TUI	table unique identifier
4.3.21 Tx	transmit
4.3.22 TxLLC	transmit logical link control
4.3.23 TxM	transmit machine
4.3.24 UCCM	unconnected message manager
4.3.25 UMAX	MAC ID of maximum unscheduled node
4.3.26 USR	unscheduled start register
4.4 Type 3: Additional symbols and abbreviations	
4.4.1 ACK	acknowledge(ment) DLPDU
4.4.2 ASM	active spare time message
4.4.3 ASP	active spare time period
4.4.4 Bus ID	bus identification, an address extension (region/DL-segment address) that identifies a particular bus as supporting routing between DL-segments
4.4.5 CRX	character receive execution
4.4.6 CS	clock synchronization
4.4.7 CTX	character transmit execution
4.4.8 DA	destination address of a DLPDU
4.4.9 DAE	destination address extension(s) of a DLPDU, which convey D_SAP_index and/or destination bus ID
4.4.10 D_SAP	destination service access point, the DLSAP associated with the remote DLS-user
4.4.11 D_SAP_index	destination service access point index – that component of a DLSAP address which designates a DLSAP and remote DLS-user within the remote DLE
4.4.12 DXM	data exchange multicast
4.4.13 ED	end delimiter of a DLPDU
4.4.14 EOA	END-OF-ACTIVITY
4.4.15 EOD	END-OF-DATA
4.4.16 EODA	END-OF-DATA-AND-ACTIVITY

4.4.17 EXT	address extension bit of a DLPDU
4.4.18 FC	frame control (frame type) field of a DLPDU
4.4.19 FCB	frame count bit of a DLPDU (FC field) used to eliminate lost or duplicated DLPDUs
4.4.20 FCV	frame count bit valid bit of a DLPDU, indicates whether the FCB is to be evaluated
4.4.21 FCS	frame check sequence (synchronous) or frame checksum (asynchronous)
4.4.22 FLC	fieldbus link control
4.4.23 G	GAP update factor, the number of token rotations between GAP maintenance (update) cycles
4.4.24 GAPL	GAP list containing the status of all stations in this station's GAP
4.4.25 IsoM	isochronous mode
4.4.26 Hd	Hamming distance, a measure of DLPDU integrity, the minimum number of bit errors that can cause acceptance of a spurious DLPDU
4.4.27 HSA	highest station address installed (configured) on this fieldbus
4.4.28 L	length of the information field, the part of a DLPDU that is checked by the FCS
4.4.29 LE	field giving the length of a DLPDU beyond the fixed part
4.4.30 LEr	field that repeats the length to increase DLPDU integrity
4.4.31 LMS	list of master stations
4.4.32 LR	local resource not available or not sufficient (DL/DLM_status of the service primitive)
4.4.33 LS	local service not activated at DL-service access point or local DLSAP not activated (DL/DLM_status of the service primitive)
4.4.34 Isb	least significant bit of a field or octet
4.4.35 max	the arithmetic maximum function
4.4.36 MCT	multicast
4.4.37 MP	message transfer message retry transfer periods
4.4.38 mr	number of retries
4.4.39 msb	most significant bit of a field or octet
4.4.40 MSRD	DLS: Send and Request Data with Multicast reply
4.4.41 mt	number of retries per token rotation
4.4.42 n	number of stations
4.4.43 NA	no acknowledgement/response (DL/DLM_status of the service primitive)

4.4.44 na	number of active stations
4.4.45 NIL	locally determined value
4.4.46 NO	not ok (DL/DLM_status of the service primitive)
4.4.47 np	number of passive stations
4.4.48 NR	no response DL-data acknowledgement negative and send data ok (DL_status of the service primitive)
4.4.49 NRZ	non-return-to-zero (PhL), an encoding technique where transitions occur only when successive data bits have different values
4.4.50 NS	next station, the station to which this master will pass the token
4.4.51 OK	service finished according to the rules (DL/DLM_status of the service primitive)
4.4.52 PhICI	PhL Interface Control Information [ISO/IEC 7498-1]
4.4.53 PhPCI	PhL Protocol Control Information [ISO/IEC 7498-1]
4.4.54 PON	power on transition occurs at a station
4.4.55 PS	previous station, the station which passes the token to this master station
4.4.56 PSP	passive spare time period
4.4.57 RDH	response DL-data high and no resource for send data (DL/DLM_status of the service primitive)
4.4.58 RDL	response DL/DLM-data low and no resource for send data (DL/DLM_status of the service primitive)
4.4.59 Res	reserved
4.4.60 RET	retry
4.4.61 RR	no resource for send data and no response DL-data available (negative acknowledgement) (DL/DLM_status of the service primitive)
4.4.62 RS	no service, or no service activated at remote DLSAP (negative acknowledgement) (DL/DLM_status of the service primitive)
4.4.63 RSYS	system message rate (in message transfer periods per second) at which confirmed DL-message exchanges are performed
4.4.64 SA	source address of a DLPDU
4.4.65 SAE	source address extension(s) of a DLPDU, which convey S_SAP_index and/or source bus ID
4.4.66 SC	single character acknowledge DLPDU
4.4.67 SD1 to SD4	start delimiters of asynchronous DLPDU transmission
4.4.68 SDA	Send Data with Acknowledge (DL-service)
4.4.69 SDA_H/L	Send Data with Acknowledge high/low (DLPDU Function)

4.4.70 SDX	start delimiter of asynchronous or synchronous DLPDU transmission
4.4.71 SDL1 to SDL5	start delimiters of synchronous DLPDU transmission
4.4.72 SDN	Send Data with No Acknowledge (DL-service)
4.4.73 SDN_H/L	Send Data with No Acknowledge high/low (DLPDU Function)
4.4.74 SM	state machine
4.4.75 SOA	START-OF-ACTIVITY
4.4.76 SRC	send receive control
4.4.77 SRD	Send and Request Data with Reply (DL-service)
4.4.78 SRD_H/L	Send and Request Data with Reply high/low (DLPDU Function)
4.4.79 SRU	send receive unit
4.4.80 S_SAP	source service access point, the DLSAP associated with the initiating local DLS-user
4.4.81 S_SAP_index	source service access point index — a component of a DLSAP-address which designates that DLSAP within the DLE at which the transaction is being initiated
4.4.82 Stn	station, a device implementing a DLE with a fieldbus DL-address
4.4.83 SYN	synchronizing bits of a DLPDU (period of idle), which guarantees the specified DLPDU integrity and allows for receiver synchronization
4.4.84 TA/R	time to transmit an acknowledgement/response DLPDU
4.4.85 TASM	ASM message time
4.4.86 tBIT	bit time
4.4.87 TCSI	clock synchronization interval time
4.4.88 TCT	isochronous cycle time
4.4.89 TGUD	GAP update time
4.4.90 TID	idle time
4.4.91 TIM	timer state machine
4.4.92 TMP	message transfer period
4.4.93 TP	token transfer period
4.4.94 TpSP	passive spare time
4.4.95 TPTG	post transmission gap time
4.4.96 TQUI	quiet time
4.4.97 TRCT	real isochronous cycle time

4.4.98	T_{RD}	receive delay time
4.4.99	T_{RDY}	ready time
4.4.100	T_{RES}	spare time
4.4.101	T_{RR}	real rotation time
4.4.102	T_S	this station
4.4.103	T_{S/R}	send/request DLPDU time
4.4.104	T_{SD}	send delay time
4.4.105	T_{SDI}	station delay of initiator
4.4.106	T_{SDR}	station delay of responder
4.4.107	T_{SET}	setup time
4.4.108	T_{SH}	time shift
4.4.109	T_{SL}	slot time
4.4.110	T_{SM}	safety margin time
4.4.111	T_{SR}	system reaction time
4.4.112	T_{SYN}	synchronization time
4.4.113	T_{SYNI}	synchronization interval time
4.4.114	T_{TC}	token cycle time
4.4.115	T_{TD}	transmission delay time
4.4.116	T_{TF}	token DLPDU time
4.4.117	T_{TH}	token holding time
4.4.118	T_{TO}	timeout time
4.4.119	T_{TP}	token transfer period
4.4.120	T_{TR}	target rotation time
4.4.121	UART	universal asynchronous receiver/transmitter
4.4.122	UC	UART character
4.4.123	UE	negative acknowledgement, remote user interface error (DL/DLM_status of the service primitive)

4.5 Type 4: Additional symbols and abbreviations

4.5.1 Constants, variables, counters and queues

4.5.1.1	BNA	broadcast node address
4.5.1.2	C(LAC)	link access counter

4.5.1.3	C(LIC)	link idle counter
4.5.1.4	SNA	service node address
4.5.1.5	NCNA	no confirm node address
4.5.1.6	Q(UR)	user request queue
4.5.1.7	V(ACPDU)	acknowledge confirmed PDU
4.5.1.8	V(AUPDU)	acknowledge unconfirmed PDU
4.5.1.9	V(BR)	bit rate
4.5.1.10	V(DC)	device class (simple or normal)
4.5.1.11	V(DMRT)	default max retry time
4.5.1.12	V(MID)	max indication delay
4.5.1.13	V(NA)	node address
4.5.1.14	V(NDLE)	number of DLEs
4.5.1.15	V(PNR)	permitted number of retries

4.5.2 Miscellaneous

4.5.2.1	RCL/ACK	response comes later / acknowledge

4.6 Type 5: Additional symbols and abbreviations

A normative specification of this protocol was not available at the time of compilation of this edition. See 4.1.1 for a preliminary set of additional symbols and abbreviations.

4.7 Type 6: Additional symbols and abbreviations

4.7.1	B or Q	buffer or queue
4.7.2	BL	bus length
4.7.3	BSS	bus sync slot
4.7.4	CEP-sel	DLCEP selector within a DLE
4.7.5	ConfigDB	configuration data base
4.7.6	DLC	DL-connection, Channel
4.7.7	DLCid, Channel VID	DLC-identifier, DL-channel identifier
4.7.8	DLEA	a DLE address
4.7.9	DLE-sel	a DLE selector (=0)
4.7.10	DP	parameter is accessible to both DLE and DLS-user
4.7.11	ED	End Delimiter of a DLPDU
4.7.12	FSM	Finite State Machine
4.7.13	HCC	high channel count architecture

4.7.14 HO	high order part of a data item
4.7.15 LBL	Log bus length
4.7.16 LCC	low channel count architecture
4.7.17 LO	low order part of a data item
4.7.18 NCAP	network capable application processor
4.7.19 PI	protocol information field
4.7.20 ppm	parts per million
4.7.21 QoS-set	set of relevant QoS parameters
4.7.22 SAP-sel	DLSAP selector within a DLE
4.7.23 spc	scans per bus cycle
4.7.24 spp	slots per scan period
4.7.25 sps	scans per second
4.7.26 STIM	smart transducer interface module
4.7.27 TDMA	time division multiple access
4.7.28 TEDS	transducer electronic data sheet
4.7.29 UUID	universal unit identifier

4.8 Type 7: Additional symbols and abbreviations

4.8.1 BA	bus arbitrator
4.8.2 B_Dat_Cons	buffer which contains the value of the Data Consumed
4.8.3 B_Dat_Prod	buffer which contains the value of the Data Produced
4.8.4 B_Req1/2	buffer containing the list of DLCEP-identifiers that are the objet of a specified explicit request for a transfer at the priority 1 (urgent) or 2 (normal)
4.8.5 ID_DAT	DLPDU used to allocate the medium to a buffer transfer
4.8.6 ID_MSG	DLPDU used to allocate the medium to a message exchange
4.8.7 ID_RQ1/2	DLPDU used to allocate the medium to a request for a buffer transfer
4.8.8 PRT	physical reaction time
4.8.9 Q_IDMSG	queue of requested DLCEP-identifiers received by the BA for message transfer
4.8.10 Q_IDRQ1/2	queue of requested DLCEP-identifiers received by the BA at the priority 1 (urgent) or 2 (normal)
4.8.11 Q_Msg_Cyc	queue which contains messages to be emitted that are associated with cyclical exchanges
4.8.12 Q_Msg_Aper	queue which contains messages to be emitted that are associated with aperiodic exchanges

4.8.13 Q_Req1/2	queue containing the list of DLCEP-identifiers that are the object of a free explicit request for a transfer at the priority 1 (urgent) or 2 (normal)
4.8.14 Q_RPRQ	queue for aperiodic transfers in progress
4.8.15 RQ_Inhibit	Indicator used to manage explicit request for variable exchanges
4.8.16 RP_ACK	DLPDU used to transfer an acknowledgement of message exchange, Transfer OK
4.8.17 RP_DAT	DLPDU used to carry the value of the identified variable previously requested.
4.8.18 RP_DAT_MSG	DLPDU used to carry the value of the identified variable previously requested with a request for a message exchange.
4.8.19 RP_DAT_RQ1/2	DLPDU used to carry the value of the identified variable previously requested with a request for an explicit transfer of variables.
4.8.20 RP_DAT_RQ1/2_MSG	DLPDU used to carry the value of the identified variable previously requested with a request for an explicit transfer of variables and a request for a message transfer.
4.8.21 RP_MSG_ACK	DLPDU used to carry the message to be exchanged with a request of acknowledgement of this exchange.
4.8.22 RP_MSG_NOACK	DLPDU used to carry the message to be exchanged without a request of acknowledgement of this exchange.
4.8.23 RP_NAK	DLPDU used to transfer an acknowledgement of message exchange, that the message was received correctly, but was not stored by the DLE.
4.8.24 RP_END	DLPDU used to terminate a message exchange.
4.8.25 STT	station turnaround time
4.8.26 TI	latency time

4.9 Type 8: Additional symbols and abbreviations

4.9.1 Basic reference model terms

4.9.1.1	BLL	basic link layer
4.9.1.2	BLLSDU	BLL service data unit
4.9.1.3	BLL_TSDU	BLL transmit service data unit
4.9.1.4	BLL_RSDU	BLL receive service data unit
4.9.1.5	MACSDU	MAC service data unit
4.9.1.6	PDL	peripherals data link
4.9.1.7	PDLSDU	PDL service data unit
4.9.1.8	PhMS	Ph-management service

4.9.2 Local variables, timers, counters and queues

4.9.2.1	add_wait	See Table 252
4.9.2.2	BLL_access_control	See Table 268
4.9.2.3	bus_timeout	See Table 268
4.9.2.4	Ccerr	See Table 259
4.9.2.5	Cconf	See Table 259
4.9.2.6	Ccycle	See Table 259
4.9.2.7	Creq_retry	See Table 259
4.9.2.8	Cswa	See Table 259
4.9.2.9	configuration_valid	See Table 268
4.9.2.10	loopback_word (LBW)	See Table 277
4.9.2.11	max_dlsdu_size_from_req	See Table 252
4.9.2.12	max_dlsdu_size_from_res	See Table 252
4.9.2.13	max_receiving_queue_depth	See Table 252
4.9.2.14	max_sending_queue_depth	See Table 252
4.9.2.15	max_spa_retry	See Table 252
4.9.2.16	max_swa_count	See Table 252
4.9.2.17	start_bus_cycle	See Table 252
4.9.2.18	time_timeout	See Table 277
4.9.2.19	trigger_mode	See Table 252
4.9.2.20	update_time	See Table 268

4.9.3 DLPDU classes

4.9.3.1	DATA	data	See Table 257
4.9.3.2	FCB_SET	frame count bit	See Table 257
4.9.3.3	IDL	idle	See Table 257
4.9.3.4	RWA	read word again	See Table 257
4.9.3.5	SPA	send parameter with acknowledge	See Table 257
4.9.3.6	SVA	send value with acknowledge	See Table 257
4.9.3.7	SWA	send word again	See Table 257

4.9.4 Miscellaneous

4.9.4.1	AT	application triggered
4.9.4.2	BAC	basic access control
4.9.4.3	CO	confirmation

4.9.4.4	CRC	cyclic redundancy check
4.9.4.5	DL-Ph	Data Link-Physical (interface)
4.9.4.6	DLI	DL-interface
4.9.4.7	DSAP	destination service access point
4.9.4.8	FCB	frame count bit
4.9.4.9	FCT	function
4.9.4.10	FMS	fieldbus message specification
4.9.4.11	GSM	get shared memory
4.9.4.12	IN	input
4.9.4.13	L_status	link status
4.9.4.14	LBW	loopback word
4.9.4.15	lsb	least significant bit
4.9.4.16	M, (m)	mandatory
4.9.4.17	msb	most significant bit
4.9.4.18	NT	network triggered
4.9.4.19	O, (o)	optional
4.9.4.20	OUT	output
4.9.4.21	PDL	peripherals data Link
4.9.4.22	PM	protocol machine
4.9.4.23	PNM1	peripherals network management of Layer 1
4.9.4.24	PNM2	peripherals network management of Layer 2
4.9.4.25	PSM	put shared memory
4.9.4.26	RUM	receive update memory
4.9.4.27	S	selection
4.9.4.28	SM	state machine
4.9.4.29	TUM	transmit update memory

5 DL-protocol elements common to multiple DL-protocol Types

5.1 Frame check sequence

Within this clause, 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 most of the protocol Types in 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"¹⁾ of length n consisting of k DLPDU message bits followed by $n - k$ 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:

The generic form of the generator polynomial for this FCS construction is specified in equation (6) and the polynomial for the receiver's expected residue is specified in equation (11). The specific polynomials for each DL-protocol type are specified in Table 1. An exemplary implementation is discussed in Annex A.

¹⁾ W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes* (2nd edition), MIT Press, Cambridge, 1972.

Table 1 – FCS length, polynomials and constants by protocol Type

Protocol	Item	Value
Type 1, Type 5, Type 6, Type 7 — Type 3 synchron- ous	$n-k$	16
	$G(x)$	$x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^6 + x^3 + x^2 + x + 1$ (notes 1, 2, 3)
	$R(x)$	$x^{15} + x^{14} + x^{13} + x^9 + x^8 + x^7 + x^4 + x^2$ (note 4)
	NOTE 1 Code words $D(X)$ constructed from this $G(X)$ polynomial have Hamming distance 4 for lengths ≤ 344 octets and Hamming distance 5 for lengths ≤ 15 octets. NOTE 2 This $G(X)$ polynomial is relatively prime to all, and is thus not compromised by any, of the polynomials commonly used in DCEs (modems): the differential encoding polynomial $1 + X^{-1}$ and all primitive scrambling polynomials of the form $1 + X^{-j} + X^{-k}$. NOTE 3 This $G(X)$ polynomial is the optimal 16-bit polynomial for burst error detection over DLPDUs of 300 octets or less when the statistics of the error burst have a Poisson distribution (as is the usual case). NOTE 4 The remainder $R(x)$ should be 1110 0011 1001 0100 (X^{15} to X^0, respectively) in the absence of errors.	
Type 2	$n-k$	16
	$G(x)$	$x^{16} + x^{12} + x^5 + 1$ (notes 5, 6, 7)
	$R(x)$	$x^{12} + x^{11} + x^{10} + x^8 + x^3 + x^2 + x + 1$ (note 7, 8)
	NOTE 5 Code words $D(X)$ constructed from this $G(X)$ polynomial have Hamming distance 4 for lengths ≤ 4095 octets, and all errors of odd weight are detected. NOTE 6 This $G(X)$ polynomial is relatively prime to all, and is thus not compromised by any, of the primitive scrambling polynomials of the form $1 + X^{-j} + X^{-k}$ sometimes used in DCEs (modems). However, it is severely compromised by use of differential coding, which uses an encoding polynomial of $1 + X^{-1}$ (a factor of $G(X)$), and therefore it should be used with PhLs which do not employ differential coding. NOTE 7 These are the same polynomials and method as specified in ISO/IEC 3309 (HDLC). NOTE 8 The remainder $R(x)$ should be 0001 1101 0000 1111 (X^{15} to X^0, respectively) in the absence of errors.	
Type 8	$n-k$	16
	$G(x)$	$x^{16} + x^{12} + x^5 + 1$ (note 9)
NOTE 9 This $G(X)$ polynomial is the same as used by Type 2. However, it is used to compute and check a message which is transmitted separately from the redundant check bits. As a consequence, the error detection properties which hold for Type 2's use apply only approximately to Type 8's use.		

5.1.1 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 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 $n-k-1$

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

That is,

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

NOTE 1 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 2 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

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

NOTE 3 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.

5.1.2 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

$$V(X) = v_1X^{u-1} + v_2X^{u-2} + \dots + v_{u-1}X + v_u \quad (8)$$

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 5.1.1) 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

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

whose value is independent of $D(X)$. Unfortunately $R(X)$ will also equal $R_{ok}(X)$ 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 $R_{ok}(X)$; 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).

5.1.3 Modification within bridges

When forwarding a Type 1 or Type 7 DLPDU, it is sometimes necessary for a bridge to alter one or more subfields of a DLPDU's frame control field. When making these modifications, the bridge can be required to modify the received FCS to compensate for changes in the frame control octet; in this case the bridge does not discard the received FCS and recompute a new FCS after the DLPDU's frame control field has been altered²⁾.

When the received DLPDU's length, plus that of its FCS field, is N octets, then the bridge can compensate for a change in bit K in the first octet by computing the residual of the polynomial

$$X^{8N+K-8} \text{ (modulo } G(X)) \quad (12)$$

and then updating the DLPDU's FCS field by exclusive-ORing the computed residual into that field.

NOTE When the bridge initializes, it can precompute the residuals for all permissible DLPDU lengths and bit positions potentially needing alteration — for Type 1, this is all values of N between 3 and 272 and of K equal to 2; for Type 7, this is all values of N between 3 and 272 and of K equal to 7. Then for any DLPDU the bridge need only apply to the DLPDU's FCS that residual which corresponds to the change actually made in the DLPDU's frame control octet.

²⁾ D. R. Irvin, *Preserving the integrity of cyclic-redundancy checks when protected text is intentionally altered*, IBM Journal of Research and Development, Vol. 33, No. 6, November 1989, pp. 618-626.

6 Type 1: Overview of the DL-protocol

6.1 Three-level model of the DLL

The DLL is modeled as

- a) a low-level of path-access-and-scheduling functions, which supports
- b) an intermediate-level of bridge-operation functions, which in turn supports
- c) a higher-level of connection-mode and connectionless data transfer, bridge-coordination, and DL-service functions.

Interoperating with all three levels are DL-management functions, including bridge and redundant-path management functions where relevant.

NOTE 1 The term “sublayer” is not appropriate for describing these levels, since ISO/IEC 7498 requires that when multiple sublayers are defined, all but one of them must be optional.

NOTE 2 This three-level partitioning closely resembles the partitioning into lower-level “MAC”, intermediate-level “bridge-operation”, and higher-level “LLC” functions found in the ISO/IEC LAN standards (see ISO/IEC TR 8802-1 for the MAC service definition), with two significant differences:

- A) This protocol's low-level functionality is contained entirely within the Data Link layer as specified by ISO/IEC 7498. In contrast, the “MAC” functionality of the ISO/IEC LAN protocols spans the lower part of the OSI Data Link Layer and the upper part of the OSI Physical Layer.
- B) This protocol employs a single level of DL-addressing within the Data Link layer. In contrast, the ISO/IEC LAN protocols employ two levels of DL-addressing, one within the “MAC” and “bridge-operation” functionality and a second within the “LLC” functionality.

6.1.1 Path access and scheduling level

The path-access-and-scheduling level provides basic communication from one DLE to another. The timing of such communication is regulated

- 1) to provide an equitable distribution of opportunities for arbitrary communications to all DLEs on the local link, generally in a cyclic but asynchronous manner, and
- 2) to provide a specific distribution of opportunities for designated communications to DLEs on the local link in accord with a pre-established schedule for the link.

It is the latter requirement for scheduled communications, in support of time-critical communications, which causes this protocol to rely more heavily on centralized real-time management of the communications opportunities than prior protocols such as ISO/IEC 8802-3, ISO/IEC 8802-4 and ISO/IEC 8802-5. The centralized manager is elected from the set of operating full-function DLEs on the local link by a procedure similar to that used in ISO/IEC 8802-4 and ISO/IEC 8802-5. Should that manager fail, a replacement is similarly elected.

NOTE Many traditional control algorithms used in the domain of applicability of this protocol require tight constraints on the cycle-to-cycle timing of information acquisition. In distributed systems such constraints frequently translate to limits on the permissible cycle-to-cycle periodicity of the messaging which conveys that repetitively acquired information.

- a) The path-access-and-scheduling level forms each DLPDU from DL-protocol control information and DLS-user data; computes and appends an appropriate frame check sequence (FCS); and passes the whole as a sequence of PhIDUs (see 6.4) to the PhE for transmission to peer PhEs for reporting to peer DLEs.

In some cases it also appends the low-order three octets either

- of the current value of the local node-timer, C(NT), or
- of a computed value minus the current value of the local node-timer, C(NT),

during DLPDU formation, immediately preceding the appended FCS.

b) The path-access-and-scheduling level receives a sequence of PhIDUs from the PhE, concatenates those PhIDUs into a received DLPDU, computes a frame check sequence over the entire sequence of received data, and checks for the proper residual value. The first octet of the received sequence is examined to determine the type of the DLPDU, and an attempt is made to parse that DLPDU into its DL-protocol control information and DLS-user data components. If the FCS residual was correct and the parse was successful, then the appropriate low-level actions are performed, possibly including reporting the parsed DLPDU to a higher level.

In some cases the value of the low-order three octets of the local node-timer, C(NT), at the time of receipt, is appended to this parsed DLPDU.

c) The path-access-and-scheduling level provides the basic functions of both responder and initiator. As a responder, it provides the sequencing functions necessary

- 1) for receiving a DLPDU, possibly conveying a reply token;
- 2) in that latter case (of the received reply token), for sending a DLPDU as an immediate reply to the just-received DLPDU.

As an initiator, it provides the sequencing functions necessary for

- 3) receiving a delegated token;
- 4) sending one or more DLPDUs, including those requiring an immediate reply;
- 5) receiving such an immediate reply, or inferring its absence;
- 6) returning a delegated token.

d) The path-access-and-scheduling level provides the low-level scheduling functionality required for scheduling DLPDU transmissions on a specific path, including any interactions with the local link's link active scheduler (LAS) to coordinate the schedule with other DLEs or to request needed path transmission capacity.

The actions of a) and b) are augmented within a bridge relay DLE) to permit the retransmission of a received sequence of data octets, including the received FCS, with possible constrained alterations of the first octet and compensating alterations see 5.1.3) of the received FCS, prior to retransmission.

The actions of c) and d) are based in part on two request-management and scheduling queues:

- i) a DL-address-specific request queue see 3.3.1), associated with the sending address, which is used to manage DLS-user requests originated from that DL-address;
- ii) the common DLE unscheduled-service queue see 6.7.1.17), which is used to manage the servicing of unscheduled requests upon receipt of the "circulated" token.

Some of the more complex scheduling functionality of this level requires, and uses, the services of the upper level, see 6.1.3) by acting as a DLE-internal quasi-DLS-user.

6.1.2 Bridge-operation level

The bridge-operation level provides the intermediate-level functionality of

- a) logically inter-connecting multiple local links into a single extended-link by physically interconnecting multiple paths;
- b) serving as a possibly-distributed "time-space-time" switch:
 - 1) providing DLPDU store-and-forward functions to permit communication between

DLEs on the extended link which could not otherwise communicate;

NOTE This includes the coordination with fractional-duty-cycle (FDC) DLEs (see 3.4.21) necessary to permit alternating periods of “FDC-DLE-awake” and “FDC-DLE-asleep” operation.

- 2) providing surrogate functions to permit delayed-reply interactions between DLEs on the extended link as an extrapolation of “immediate-reply” transactions on a local link;
- c) providing a shared sense of DL-time throughout the extended link;
- d) coordinating local-link scheduling among two or more local links to provide any necessary multi-link coordination of scheduling within the extended link.

Much of the bridge-operation level of functionality is active only in “bridge” DLEs (see 6.6).

6.1.3 Connection-mode and connectionless data transfer, bridge-coordination, and DL-service level

The connection-mode and connectionless data transfer, bridge-coordination, and DL-service level provides the higher-level functionality of

- a) managing all DLE interactions with the DLS-user, converting all DLS-user request and response primitives into the necessary sequence of DLE operations, and generating DLS-user indication and confirm primitives where appropriate;
- b) managing the sequencing of each active DLCEP, including
 - 1) determination of the type and sequence of DLPDUs to be transmitted from the DLCEP;
 - 2) QoS negotiation;
 - 3) determination of the DLPCI to be included in each DLPDU;
 - 4) segmentation and reassembly of large DLSDUs;
- c) managing the sequencing of unitdata transactions which require a response from a peer DLE, including
 - 1) determination of the DLPCI to be included in each DLPDU;
 - 2) correlation of a non-immediate reply DLPDU with the requesting transaction;
- d) processing all inter-DLE state-information-distribution DLPDUs, such as TIME DISTRIBUTION (TD) (see 8.10 and 10.4.1) and bridge configuration (see 12.4) DLPDUs and LAS-backup SPDUs (see clause 11);
- e) managing all query-DLPDU/reply-DLPDU interactions with other DLEs, other than those which occur as an “immediate reply” and which are made on a reactive basis by the lower-level functions of 6.1.1.

These query-DLPDU / reply-DLPDU interactions also include computation of round-trip-delays (see 8.11, 8.12, and 10.4.1), support of remotely-initiated DL-SUBSCRIBER-QUERY (see 10.2.3) and DL-LISTENER-QUERY (see 10.3.4) requests, and inter-bridge exchanges of bridge state information (see 12.4).

6.2 Service provided by the DLL

The DLL provides connectionless data transfer services for limited-size DLSDUs, 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, and a DL(SAP)-address, queue and buffer management service.

NOTE IEC 61158-3 shows many of the relationships among DLC QoS attributes.

Some relevant QoS attributes are as follows:

6.2.1 QoS – DLCEP class

Each DLCEP establishment request specifies the class of the DLCEP. The three choices for DLCEP-class are

- a) **PEER** — the DLS-user can exchange DLSDUs with one other peer DLS-user;
- b) **PUBLISHER** — the DLS-user can send DLSDUs to a set of zero or more associated subscribing DLS-users, and receive DLSDUs from any of those subscribing DLS-users;
- c) **SUBSCRIBER** — the DLS-user can receive, and request, DLSDUs from the associated publishing DLS-user, and can send DLSDUs to that publishing DLS-user.

6.2.2 QoS – DLCEP data delivery features

Both members of a peer DLC, or the publishing DLS-user of a multi-peer DLC, specify the data delivery features of the DLC's DLCEP(s). The five choices for DLCEP data delivery features, and their effects, are

- a) **CLASSICAL** — the DLS-user can send data which will be delivered without loss, duplication or misordering. All relevant DLS-users will be notified of any loss of synchronization on the DLC.
- b) **DISORDERED** — the DLS-user can send data which will be delivered immediately upon receipt, without duplication but potentially in a different order than that of the sending DLS-user. All relevant DLS-users will be notified of any unrecoverable loss of DLS-user data or loss of synchronization on the DLC.
- c) **ORDERED** — the DLS-user can send data which will be delivered immediately upon receipt, without duplication or misordering, but with potential loss of some DLS-user data. Loss of DLS-user data will not be reported, and recovery of DLS-user data lost prior to the last-reported DLS-user data will not be attempted.
- d) **UNORDERED** — the DLS-user can send data which will be delivered immediately upon receipt. Loss, duplication and misordering of DLS-user data will not be detected or reported. No attempt will be made by the DLS-provider to recover from such events.
- e) **NONE** — the DLS-user cannot send data in this direction of data transfer.

On a peer DLC, the QoS value for the sending DLCEP data delivery features may be chosen independently for each direction of data transfer. On a multi-peer DLC, the QoS value for the DLCEP data delivery features for the subscribers-to-publisher direction of data transfer is restricted to UNORDERED and NONE. The default QoS value for the DLCEP data delivery features in the publisher-to-subscribers direction is UNORDERED.

6.2.3 QoS – DLL priority

All DLCEP establishment requests and responses, all connectionless data transfer requests, and many DL-scheduling requests, specify an associated DLL priority used in scheduling DLL data transfer services. This DLL priority also determines the maximum amount of DLS-user data that can be conveyed in a single DLPDU. The three DLL priorities with their corresponding ranges of conveyable DLS-user data (per DLPDU) are, from highest priority to lowest priority:

- a) **URGENT** — ≤ 64 DLS-user octets per DLPDU;

- b) **NORMAL** — ≤ 128 DLS-user octets per DLPDU;
- c) **TIME-AVAILABLE** — ≤ 256 DLS-user octets per DLPDU.

NOTE 1 URGENT and NORMAL are considered **time-critical** priority levels; TIME-AVAILABLE is considered a **non-time-critical** priority level.

NOTE 2 DLC establishment and DLC release DLPDUs which are sent at TIME-AVAILABLE priority are restricted to convey no more DLS-user data than would be permitted at NORMAL priority — 128 octets.

6.2.4 QoS – DLPDU authentication

Each DLCEP establishment negotiation, and each connectionless data transfer, uses this attribute to determine

- a) a lower bound on the amount of DL-addressing information used in the DLPDUs that provide the associated DLL data transfer services;

NOTE This has a slight impact on the residual rate of DLPDU misdelivery; more addressing information reduces the potential for misdelivery.

- b) whether the current state of a sending peer or publisher DLCEP should be sent at low-frequency to the DLC's peer or subscriber DLCEP(s) even when there are no unconfirmed DLS-user requests outstanding at the sending DLCEP;

NOTE This continuing background transmission is known as **residual activity**.

- c) whether all related scheduling actions should be executed locally.

NOTE These last two aspects are of particular importance in safety systems.

The three levels specifiable, with their amounts of DL-addressing information, are

- 1) **ORDINARY** — each DLPDU shall include the minimum amount of addressing information necessary;
- 2) **SOURCE** — each DLPDU shall include a source DL-address where possible;
- 3) **MAXIMAL** — each DL-address shall include the maximal amount of addressing information possible. Also, all related scheduling actions should be executed locally; and each sending peer or publisher DLCEP of the DLC should maintain a low-frequency report of state information when there is no DLS-user activity.

6.2.5 QoS – DLL maximum confirm delay

Each DLCEP establishment request, and each response, specifies upper bounds on the maximum time duration permitted for the completion

- a) of a DL-CONNECT, DL-RESET or DL-SUBSCRIBER-QUERY primitive, and, separately;
- b) of a DL-DATA primitive.

Each connectionless service request specifies an upper bound on the maximum time duration permitted for the completion

- c) of a locally-confirmed DL-UNITDATA primitive, and, separately;
- d) of a remotely confirmed DL-UNITDATA primitive, a DL-LISTENER-QUERY primitive, or an instance of the DL-UNITDATA-EXCHANGE service.

Each parameter either has the value **UNLIMITED** or specifies an interval, in units of 1 ms, from 1 ms to 60 s, inclusive. The value **UNLIMITED** provides compatibility with prior OSI protocols, and provides a means for DL-CONNECT requests to remain in a “listening” or “half-open” state. The completion status of “timeout” cannot occur on a DLS-user request which specifies **UNLIMITED**.

The parameters for the DL-DATA and locally-confirmed DL-UNITDATA primitives specify intervals less than or equal to that for the DL-CONNECT, DL-RESET, DL-SUBSCRIBER-QUERY, remotely-confirmed DL-UNITDATA, and DL-LISTENER-QUERY primitives.

The intervals specified are the maximum permissible delays

- 1) between the issuing of the specified request primitives and the issuing of the corresponding confirm primitives;
- 2) between the initiation and completion of a single instance of the specified publishing or unitdata-exchange service.

NOTE For DLEs that do not support a time resolution of 1 ms, the requested time interval may be rounded up to the next-greatest multiple of that resolution that the DLE does support, or to approximately 60 s if the DLE has no sense of time.

Failure to complete a DL-CONNECT or DL-RESET request within the specified interval shall result in a DLS-provider initiated release of the DLCEP, and possibly of the DLC.

6.2.6 QoS – DL-scheduling-policy

For each DLSAP-address, and each DLCEP, the DLS-user can override the normal (implicitly-scheduled) DLL policy of providing the requested DL-service as soon as possible, and instead can defer any inter-DLS-user communication required by a DL-DATA or DL-UNITDATA request DLS-primitive until that deferral is released by an involved DLS-user. Each such release, by execution of a DL-COMPEL-SERVICE request, specifying the DLSAP-address or DLCEP, permits the completion of just a single deferred request DLS-primitive. Only DL-services that provide DLS-user intercommunication are affected by this attribute.

The two choices are:

- a) **IMPLICIT** — any required communications with peer DLS-user(s) from this DLSAP-address, or from this DLCEP, will occur as soon as possible;
- b) **EXPLICIT** — any required data or unitdata communications with peer DLS-user(s) from this DLSAP-address, or from this DLCEP, will occur only when the deferral is explicitly released by an involved DLS-user.

NOTE Scheduling of DLPDU transmission to support the DL-CONNECT, DL-RESET and DL-DISCONNECT services, and to support responder-deferred replies in the DL-UNITDATA-EXCHANGE and remotely-confirmed DL-UNITDATA services, is always IMPLICIT. Scheduling of DLPDU transmission to initiate the DL-UNITDATA-EXCHANGE service is always EXPLICIT.

6.2.7 QoS – maximum DLSDU sizes

Each DLC / DLCEP establishment request, and each response, specifies an upper bound on the size (in octets) of DLSUDs which will be offered for transmission, and an upper bound on the size of DLSUDs which are acceptable for reception.

For peer DLCs, the negotiated maximum DLSDU size shall be determined independently for each direction of data transfer as the smallest of that offered by the sender, that permitted by the sender's local DL-management, that permitted by the receiver's local DL-management, and that permitted by the receiver.

The sender's offered size may be any value between zero and 16 times the maximum number of DLS-user octets per DLPDU, as specified in 6.2.3. The receiving DLE and all DL-management agents shall choose their maximum permitted sizes from the following list of sizes: 0, 64, 128, $256 \times N$ where $1 \leq N \leq 16$.

NOTE 1 The maximum size DLSDU supported by this protocol is 16 times the maximum number of octets of DLS-user data conveyable in a single DLPDU, as determined by the DLC's DLL Priority (see 6.2.3).

NOTE 2 The set of maximum permitted DLSDU sizes is limited to the above small list of values to promote interoperability. The sender's maximum specified size is permitted to take any value within the range permitted by the DLC's DLL priority to facilitate optimization of the shared communications capacity of the DLL.

NOTE 3 A value of zero (0) corresponds to the choice of simplex service, as specified by the DLS-user by the choice NONE as described in 6.2.2e).

For multi-peer DLCs, the negotiated maximum DLSDU size shall be the smaller of that offered by the publisher and that permitted by the publisher's local DL-management. For subscribers of multi-peer DLCs, the DLC shall be refused by the DLS-provider (the subscriber's DLE) if the maximum DLSDU size established by the publisher is larger than the smaller of that permitted by the subscriber and that permitted by the subscriber's local DL-management.

The publisher's offered size in the publisher-to-subscribers direction may be any value between zero and 16 times the maximum number of DLS-user octets per DLPDU, as specified in 6.2.3. The publisher's offered size in the subscriber-to-publisher direction may be any value between zero and the maximum number of DLS-user octets per DLPDU, as specified in 6.2.3.

The subscribers and all DL-management agents shall choose their maximum permitted sizes from the following list of sizes: 0, 64, 128, $256 \times N$ where $1 \leq N \leq 16$.

NOTE 4 The maximum size DLSDU supported by this protocol in the publisher-to-subscribers direction is 16 times the maximum number of octets of DLS-user data conveyable in a single DLPDU, as determined by the DLC's DLL Priority (see 6.2.3).

NOTE 5 The maximum size DLSDU supported by this protocol in the subscriber-to-publisher direction is the maximum number of octets of DLS-user data conveyable in a single DLPDU, as determined by the DLC's DLL Priority (see 6.2.3).

NOTE 6 The set of maximum permitted DLSDU sizes is limited to the above small list of values to promote interoperation. The publisher's maximum specified DLSDU size is permitted to take any value within the range permitted by the DLC's DLL priority to facilitate optimization of the shared communications capacity of the DLL.

NOTE 7 A value of zero (0) corresponds to the choice of simplex service, as specified by the DLS-user by the choice NONE as described in 6.2.2e).

The default value for both the sender's and receiver's maximum DLSDU size is the maximum number of DLS-user octets which can be carried by a single DLPDU of the specified DLL priority. The DLS-provider shall always support this DLSDU size.

6.2.8 QoS – DLCEP and DLSAP-address buffer-and-queue bindings

Each DLCEP establishment request, and each response, can bind one or two local retentive buffers or specified-depth FIFO queues, created by DL-CREATE buffer and queue management primitives (or by DL-management), to the DLCEP:

NOTE When these bindings are made for a DLS-user of a peer DLC, or a publishing DLS-user of a multi-peer DLC, they determine the maximum transmit window (that is, number of transmitted but unacknowledged DL-SDUs) for that direction of DLC data transfer. Since the size of the transmit window can also be limited by DL-management, or by an implementation, the queue depth only imposes an upper limit on the window size.

- a) One queue or retentive buffer can be bound to a DLCEP to convey DLSDUs from the DLS-user to the DLS-provider.
- b) One queue or retentive buffer can be bound to a DLCEP to convey DLSDUs from the DLS-provider to the DLS-user.
- c) It is also possible to bind a queue or retentive buffer to be written at one DLCEP and to source data at another DLCEP. Such an intermediate buffer or queue can serve to cross scheduling boundaries or redistribute received DLS-user data to a second set of DLS-users.

Such a binding is made by specifying, for the appropriate parameter, a buffer-or-queue DL-identifier which resulted from a prior DL-CREATE request (or by DL-management), and which has not yet been deleted.

When the sending DLCEP data delivery features specify UNORDERED or ORDERED, both the sender and receiver(s) may specify a queuing policy of BUFFER-R or QUEUE. When the DLCEP's sending data delivery features specify DISORDERED or CLASSICAL, both the sender and receiver(s) may specify a queuing policy of QUEUE; a queuing policy of BUFFER-R is not permitted. A queuing policy of BUFFER-NR is never permitted.

Each DLSAP-address bind request can bind up to six local retentive buffers or non-retentive buffers or specified-depth FIFO queues, created by DL-CREATE buffer and queue management primitives (or by DL-management), to the DLSAP-address:

- d) One buffer or queue can be bound to the sending direction of a DLSAP-address at each priority to convey DLSDUs from the DLS-user to the DLS-provider. Buffers can be bound only to DLSAP-addresses whose DLSAP-role is INITIATOR or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER. Queues can be bound only to DLSAP-addresses whose DLSAP-role is BASIC.
- e) One buffer or queue can be bound to the receiving direction of a DLSAP-address at each priority to convey DLSDUs from the DLS-provider to the DLS-user. Buffers can be bound only to DLSAP-addresses whose DLSAP-role is INITIATOR or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER. Queues can be bound to all DLSAP-addresses.

6.2.8.1 Binding to a buffer

When a sending buffer is bound to a DLCEP, or to a DLSAP-address and DLL priority, by a DLS-user,

- a) a DL-PUT request primitive overwrites the buffer with a DLSDU, or may set the buffer empty;
NOTE After creation, the buffer is empty.
- b) a DL-COMPEL request primitive, specifying either
 - 1) a DLCEP, or
 - 2) unitdata-exchange at a DLSAP-address

causes the transmission, at the first opportunity, of the then most recent contents of the buffer; the primitive does not itself specify a DLSDU;

- c) a DL-BUFFER-SENT indication primitive notifies the DLS-user of the specific DLCEP on which the retentive buffer was transmitted, and to which the buffer is bound, that the buffer was just transmitted;
- d) a DL-UNITDATA-EXCHANGE indication primitive notifies the single DLS-user attached to the specific DLSAP-address from which the buffer was transmitted (and to which the buffer is bound) that the buffer was just transmitted on that DLSAP-address, and emptied if it is non-retentive (BUFFER-NR).

When a receiving buffer is bound to a DLCEP, or to a DLSAP-address and DLL priority, by a DLS-user,

- e) a DL-BUFFER-RECEIVED or DL-UNITDATA-EXCHANGE indication primitive notifies the DLS-user of the overwriting of the buffer by a newly-received DLSDU; the primitive does not itself specify a DLSDU;
- f) a DL-GET request primitive copies the DLSDU from the buffer, and empties the buffer if it is non-retentive (BUFFER-NR).

Multiple concurrent output bindings to a retentive buffer are permitted as an implementation and conformance (see 12.6.1) option, but are not required by this protocol.

6.2.8.2 Binding to a FIFO queue

When a sending FIFO queue of maximum depth K is bound to a DLCEP, or to a DLSAP-address and DLL priority, by a DLS-user,

- a) a DL-PUT request primitive is not permitted;
- b) a DL-DATA or DL-UNITDATA request primitive attempts to append a DLSDU to the queue, but fails if the queue already contains K DLSDUs. If the append operation was successful, then the DLSDU will be transmitted at the first opportunity, after all preceding DLSDUs in the queue.

When a receiving FIFO queue of maximum depth K is bound to a DLCEP, or to a DLSAP-address and DLL priority, by a DLS-user,

- c) a DL-GET request primitive attempts to remove a DLSDU from the queue, but fails if the queue is empty.
- d) a DL-DATA or DL-UNITDATA or DL-UNITDATA-EXCHANGE indication primitive notifies the receiving DLS-user of the result of appending a newly received DLSDU to the receive queue; the primitive does not itself specify a DLSDU.

Multiple concurrent input bindings to a FIFO queue are permitted as an implementation option, but are not required by this protocol.

6.2.8.3 Default bindings

When these binding options are not specified, or for DLS-primitives for which explicit binding is not applicable, the conventional implicitly-queued sending and direct receiving interfaces between DLS-user and DLS-provider are employed. In this case each DL-DATA and DL-UNITDATA primitive conveys a DLSDU, and each DL-CONNECT, DL-DISCONNECT and DL-RESET primitive may convey a DLSDU:

- a) DL-PUT and DL-GET request primitives are not permitted;
- b) A DL-DATA or DL-UNITDATA or DL-DISCONNECT request primitive, or a DL-CONNECT or DL-RESET request or response primitive, issued by the sending DLS-user attempts to append a DLSDU to the implicit queue, but fails if the queue is full. If the append operation was successful, then the DLSDU will be transmitted at the first opportunity, after all preceding DLSDUs in the queue.
- c) A DL-DATA or DL-UNITDATA or DL-DISCONNECT indication primitive, or a DL-CONNECT or DL-RESET indication or confirm primitive, notifies a receiving DLS-user of a newly received DLSDU, and conveys that DLSDU to the DLS-user. No apparent queuing is provided within the DLL.

6.2.9 DL-timeliness

This attribute applies only to DL-buffers, to DLCEPs at which DL-buffers are bound, and to those DLS-primitives which transfer DLS-user data to or from DL-buffers at such DLCEPs.

Each DLCEP establishment request, and each response, can specify DL-timeliness criteria which are to apply to information sent from, or received into, buffers at that DLCEP. Four types of DL-timeliness can be supported: RESIDENCE timeliness, UPDATE timeliness, SYNCHRONIZED timeliness, and TRANSPARENT timeliness. All four types of timeliness, and the case where there is no timeliness, are shown in IEC 61158-3, Figure 6.

- a) **RESIDENCE** timeliness is an assessment based upon the length of time that a DLS-user datum has been resident in a buffer, which is the time interval between

- 1) the moment when the buffer is written (by a DL-PUT request primitive, or by reception into the buffer at a DLCEP);

- 2) the moment when the buffer is read (by a DL-GET request primitive, or by transmission from the buffer at a DLCEP);

$$\text{DL-timeliness} \equiv 0 \leq (R_T - W_T) < \Delta T \quad (13)$$

NOTE This type of timeliness is also known as “Asynchronous”.

- b) **UPDATE** timeliness is an assessment based upon the time interval between

- 1) the moment of occurrence of a multi-DLE synchronizing event (a DL-BUFFER-RECEIVED indication or DL-BUFFER-SENT indication);

- 2) the moment when the buffer is written (by a DL-PUT request primitive, or by

reception into the buffer at a DLCEP);

$$\text{DL-timeliness} \equiv 0 \leq (WT - ST) < \Delta T \quad (14)$$

NOTE A type of timeliness closely related to this one is also known as “**Punctual**”.

c) **SYNCHRONIZED** timeliness is an assessment based upon the time intervals and timing relationships between

1) the moment of occurrence of a multi-DLE synchronizing event (a DL-BUFFER-RECEIVED indication or DL-BUFFER-SENT indication);

2) the moment when the buffer is written (by a DL-PUT request primitive, or by reception into the buffer at a DLCEP);

3) the moment when the buffer is read (by a DL-GET request primitive, or by transmission from the buffer at a DLCEP);

$$\text{DL-timeliness} \equiv 0 \leq (WT - ST) \leq (RT - ST) < \Delta T \quad (15)$$

NOTE This type of timeliness is also known as “**Synchronous**”.

d) **TRANSPARENT** timeliness occurs when timeliness is selected on a DLCEP but none of the above assessments are performed. In such a case the DLC preserves any prior buffer timeliness, but does not itself invalidate that timeliness. When no prior buffer timeliness exists, the default timeliness value shall be TRUE.

e) **No** timeliness occurs when timeliness is not selected on a DLCEP. In such a case the DL-timeliness attribute of DLS-user data always shall be FALSE.

Where a buffer read or write occurs over a significant time interval, and not just as a momentary event, then the overall timeliness of the read or write operation shall be computed as the timeliness at the beginning of the read or write operation, logically ANDed with the timeliness at the end of the read or write operation, all using the same timeliness criteria.

6.2.10 Remote-DLE-confirmed

Each unitdata transfer request can specify whether confirmation of receipt of the associated DLSDU by the (implicitly addressed) remote DLE is required. Its permissible values are **TRUE** and **FALSE**.

NOTE Selection of the value TRUE inevitably uses more link capacity than does selection of the value FALSE.

6.3 Structure and definition of DL-addresses

DL-addresses are used as DLSAP-addresses and group DL-addresses, as DLCEP-addresses, and as DLSEP-addresses. DL-addresses conform to the following structure and coding limitations.

This subclause defines the form of DL-addresses and the usage of various ranges of DL-address components. It includes specific definitions for some standard DL-addresses.

This DL-protocol uses individual (non-group) DL-addresses for other purposes than simply as DLSAP-addresses. The same terminology and the following considerations apply to those non-DLSAP DL-addresses as well.

NOTE 1 This usage extends the definition of DL-addresses beyond that specified in ISO/IEC 7498-3.

NOTE 2 DL-addresses are also addressed to some extent in 7.1.

6.3.1 Form of DL-addresses

A standard DL-address may be considered to consist of two parts: link-designator and sublink selector. The link designator is an unsigned integer, two octets in length. The sublink selector is also two octets in length.

Link Designator	Sublink Selector
2 octets	2 octets

Figure 4 – Basic Structure of a DL-address

Most values of the link designator specify an individual link. In these cases, the sublink selector specifies a unique DL-address within the designated local link, as further described in 6.3.2.2 and 6.3.2.3. When lacking address-specific forwarding information, bridges (relay DLEs) shall base their forwarding decisions only on the link designator portion of such addresses.

However, many values of the link designator specify that the link designator portion of the address is concatenated with the sublink selector portion to form a non-hierarchical (that is, flat) address. When lacking address-specific information, bridges shall not forward such addresses.

A standard sublink selector itself consists of two subparts: node-designator and subnode selector. Each of these subparts is a small unsigned integer, one octet in length.

implied Link Designator (0000)	Node Designator	Sub-node Selector
	1 octet	1 octet

Figure 5 – Basic Structure of a Sublink Selector

Most values of the node designator specify an individual node. In these cases, the subnode selector specifies a unique DL-address within the designated local node.

However, some values of the node designator specify that the node designator portion of the address is concatenated with the subnode selector portion to form a non-hierarchical (that is, flat) subaddress.

This almost-hierarchical structure of link designators, node designators, and subnode selectors was chosen primarily to facilitate the administration of DL-addresses, and secondarily to reduce the number of entries in bridge address-forwarding tables.

NOTE Protocol implementations may find that this partially hierarchical structure is of use during address recognition.

The resulting structural possibilities for DL-addresses are (where “||” indicates concatenation of the nominal link, node and selector designator subfields):

Link	Node	Selector
2 octets	1 octet	1 octet

or

Link	Node Selector
2 octets	2 octets

or

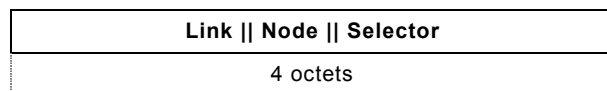


Figure 6 – DL-address Alternative Structures

A DL-address in which the link designator has a value of zero is known as a *local* address because its context for interpretation is limited to the local link. In contrast, other DL-addresses are known as *extended* addresses, because their interpretation context is the entire extended link.

There are four distinct types of DLS-user-visible DL-address:

- a) DLSAP-addresses, which are associated with only one DLSAP in the entire address interpretation context;
- b) Group DL-addresses, which are potentially associated with many DLSAPs in one or more DLEs, or with many DLEs;
- c) DLCEP-addresses, which are associated
 - 1) with only one DLCEP and associated DLSAP of a peer-to-peer DLC, or
 - 2) directly with the publishing DLCEP and associated DLSAP of a multi-peer DLC, and indirectly with all the subscribing DLCEPs and associated DLSAPs of a multi-peer DLC;
- d) DLSEP-addresses, which are associated with one sequence-end-point of a single DLE.

DLSAP-addresses may be used with all of the primitives of the DLS which specify a DLSAP-address. Group DL-addresses shall not be

- 1) bound to a queue as a source DLSAP-address, or
- 2) used as a source DLSAP-address in any connection-oriented or connectionless DLS primitive.

Individual non-"DLSAP" DL-addresses may be used as appropriate within the DL-protocol, and where permitted as specific parameters within the DL-services.

6.3.2 Predefined Values and Ranges for DL-Address Components

Some values of the link-designator or node-designator component of DL-addresses designate a group DL-address. All other combinations of values of the link-designator and node-designator components designate individual DL-addresses. This is summarized in Table 2 through Table 4.

NOTE All link, node and selector values are specified in hexadecimal (radix 16).

6.3.2.1 Link designators

The link designator component of a DL-address usually designates a specific single local link. Values for the link designator component are predefined, or their usage is constrained, as follows:

0000 The local link. The zero (0000) value is provided as a convenience for the unification of service primitives, to enable implementations to refer to the local link's address by a constant zero (0000), and to enable isolated-link operation when no bridge is present to identify the link component of an extended link. When the DLPDU-authentication QoS does not specify *MAXIMAL* and the DL-addresses themselves permit it, the link-designator components of all DL-addresses shall be omitted before transmission. Otherwise, the nominal-link-id of the sourcing local link, if known, shall be substituted for the zero (0000) link value before transmission.

0001 Non-local group DL-addresses. These link designator values, concatenated with the associated node designator values and selector values — *link || node || selector* —
...
003F form an address space of 4 128 768 identifiers usable as group DL-addresses. The resulting DL-addresses do not implicitly designate any specific link.

Non-local group DL-addresses shall be allocated individually or in contiguous blocks. When considered as unsigned 32-bit integers, the first (lowest-valued) eight such addresses are reserved for use by the DL-protocol; the next 56 such addresses are reserved for standardized group DL-addresses. All of the other addresses are available for use as non-standardized group DL-addresses.

0040 Non-local globally-administered group DL-addresses. These link designator values, concatenated with the associated node designator values and selector values — *link*
...
007F *|| node || selector* — form an address space of 4 194 304 identifiers for globally-administered group DL-addresses. These DL-addresses are formed by prefixing the 10-bit string "0000 0000 01" to the lower 22 bits of the Organizationally Unique Identifiers specified in the ISO/IEC 8802 family of Local Area Network standards. These DL-addresses do not implicitly designate any specific link.

Table 2 – Link || node || selector addressing

Link	N S	Assigned use for specified DL-address range
0000 ... FFFF	0000 ... FFFF	local link DL-addresses as specified in Table 3.
0001 ... 0007	0000 ... 0007	flat non-local group DL-addresses reserved for DLL use
0001 ... 003F	0008 ... 003F	flat non-local group DL-addresses reserved for standard AEs
0001 ... 003F	0040 ... FFFF	flat non-local group DL-addresses available for vendor, user and dynamic assignment
0040 ... 007F	0000 ... FFFF	flat non-local globally-administered group DL-addresses — potentially one per vendor (Note 1)
0080 ... 0007	0000 ... 0007	flat non-local individual “DLSAP”-addresses reserved for DLL use
0080 ... 003F	0008 ... 003F	flat non-local individual DLSAP-addresses reserved for standard AEs
0080 ... 00FF	0040	flat non-local individual DLSAP-addresses available for vendor, user and dynamic assignment
0100 ... 0FFF	0000 ... FFBF	flat non-local individual DL-addresses available for vendor, user and dynamic assignment as DLCEP-addresses and DLSEP-addresses
0FFF ... FFF7	FFC0 ... FFF7	flat non-local individual DL-addresses reserved for standard AE DLCEP-addresses and DLSEP-addresses
0FFF ... FFFF	FFF8 ... FFFF	flat non-local individual DL-addresses reserved for DLL use as DLCEP-addresses and DLSEP-addresses
1000 ... FEFF	0000 ... FFFF	Individual links of DL-addresses as specified in Table 3.
NOTE These group DL-addresses are based on the Organizationally Unique Identifiers specified in the ISO/IEC 8802 family of Local Area Network standards.		

0080 Non-local individual DLSAP-addresses. These link designator values,
... concatenated with the associated node designator values and selector values
00FF — *link || node || selector* — form an address space of 8 388 608 identifiers
usable as individual DLSAP-addresses. The resulting DLSAP-addresses do
not implicitly designate any specific link or node.

Non-local individual DLSAP-addresses shall be allocated individually or in contiguous blocks. When considered as unsigned 32-bit integers, the first (lowest-valued) eight such addresses are reserved for use by the DL-protocol as “DLSAP”-addresses; the next 56 such addresses are reserved for standardized individual DLSAP addresses for standard AEs. All of the other addresses are available for use as non-standardized individual DLSAP addresses.

0100 Non-local individual non-DLSAP DL-addresses usable as DLCEP-addresses
... and DLSEP-addresses. These link designator values, concatenated with the
associated node designator values and selector values — *link || node ||*

0FFF *selector* — form an address space of 251 658 240 identifiers usable as individual DLCEP- and DLSEP-addresses. The resulting DL-addresses do not implicitly designate any specific link or node.

Non-local individual non-DLSAP DL-addresses shall be allocated individually or in contiguous blocks. When considered as unsigned 32-bit integers, the last (highest-valued) eight such addresses are reserved for use by the DL-protocol; the next last 56 such addresses are reserved for standardized individual DLCEP-addresses or DLSEP-addresses for standard AEs. All of the other addresses are available for use as non-standardized individual DL-CEP addresses and DLSEP-addresses.

1000 Designators of individual links. Link designators shall be allocated,
... individually or in contiguous blocks, in order of increasing address values
FEFF when considered as unsigned 16-bit integers.

The value of the DL-management configuration parameter V(ML) shall be greater than or equal to the largest of the link-designator values assigned to links of the extended link.

NOTE This parameter is used by bridges to assist in the provision of default forwarding actions for each link.

NOTE 1 The potential link-designator values FF00 – FFFF are reserved to support the implementation of identifier-references to DL-addresses as noted in IEC 61158-3.

In summary, each DLE capable of recognizing LONG DL-addresses (see 7.2.2) recognizes link designator values 0001 – 0FFF as designating the extended group and individual address spaces — *link* || *node* || *selector* — common to the extended link.

Each DLE capable of recognizing LONG DL-addresses recognizes link designator value 0000 as designating the nominal-link, which is a single local link designator value ≥ 1000 . The resulting identifier values — *nominal-link-id* || *node* || *selector* — form an address space of up to 65 280 identifiers usable as individual and group DL-addresses as described in 6.3.2.2.

NOTE 2 The *link* || *node* || *selector* components of these address identifiers take the hexadecimal form *wwwxxyy*, where $1000 \leq www$, $01 \leq xx \leq FF$ and $00 \leq yy \leq FF$. When $01 \leq xx \leq 03$, such addresses are group DL-addresses; all other such addresses are individual DL-addresses. Addresses where $1\ 000 \leq www$ and $xx = 00$ refer to the local node, and are valid only within the software of that node.

When the address space needs of a given link exceed that provided by a single nominal-link-id, secondary link-ids may be allocated to that link. For each such secondary link-id, the resulting values — *secondary-link-id* || *node* || *selector* — form an additional address space of 57 600 identifiers usable as additional individual and group DL-addresses for the link as described in 6.3.2.2.

NOTE 3 The *link* || *node* || *selector* components of these additional address identifiers take the hexadecimal form *wwwxxyy*, where $1\ 000 \leq www$, and either $0140 \leq xxyy \leq 03FF$, or $0440 \leq xxyy \leq 0FBF$, or $10 \leq xx \leq FF$ and $08 \leq yy \leq F7$. When $01 \leq xx \leq 03$, such address are group DL-addresses; all other such addresses are individual DL-addresses.

DLEs which are not capable of recognizing LONG DL-addresses recognize only SHORT and VERY-SHORT addresses (see 7.2.2), which always designate the nominal-link. Such DLEs are not capable of directly communicating or being directly managed through a bridge.

6.3.2.2 Node designators

The node designator component of a DL-address usually designates a single DLE. When the link designator component does not have the value 0001 – 0FFF, then the values for the node designator component are predefined, or their usage is constrained, as follows:

Table 3 – Link-local node || selector addressing

N	S	Assigned use for specified DL-address range
00	00 ... FF	local node DL-addresses as specified in Table 5.
01	00 ... 07	flat link-local group DL-addresses reserved for DLL use
01	08 ... 3F	flat link-local group DL-addresses reserved for standard AEs
01 ... 03	40 ... FF	flat link-local group DL-addresses available for vendor, user and dynamic assignment
04	00 ... 07	flat link-local individual "DLSAP"-addresses reserved for DLL use
04	08 ... 3F	flat link-local individual DLSAP-addresses reserved for standard AEs
04	40 ... FF	flat link-local individual DLSAP-addresses available for vendor, user and dynamic assignment
05 ... 0F	00 ... BF	flat link-local individual DL-addresses available for vendor, user and dynamic assignment as DLCEP-addresses and DLSEP-addresses
0F	C0 ... F7	flat link-local individual DL-addresses reserved for standard AE DLCEP-addresses and DLSEP-addresses
0F	F8 ... FF	flat link-local individual DL-addresses reserved for DLL use as DLCEP-addresses and DLSEP-addresses
10 ... FF	00 ... FF	Individual nodes of DL-addresses as specified in Table 5.

- 00 The local DLE. Equivalent to a DLE-internal address. This designator is provided as a convenience for the unification of service primitives, and to enable implementations to refer to the local DLE address by a constant zero (00). When transmission is appropriate, an implementation shall substitute the DLE's own nominal-node-id for this zero (00) address.

NOTE 1 As a consequence, DLPDUs with a node designator component of zero (00) in a DL-address (other than a flat DL-address) are never transmitted, and are discarded upon reception.

This node designator shall not be used with secondary-link-ids.

- 01 Link-local group DL-addresses. These node designator values, concatenated with the nominal or a secondary link-id of the local link and the associated selector values —
...
03 *link-id || node || selector* — form address spaces of 768 identifiers each, usable as group DL-addresses.

Link-local group DL-addresses shall be allocated, individually or in contiguous blocks, in order of increasing address values when considered as unsigned 16-bit integers. The first (lowest-valued) such addresses are reserved for use by the DL-protocol; the next 56 such addresses are reserved for standardized group DL-addresses. All of the other addresses are available for use as non-standardized group DL-addresses

- 04 Link-local individual DLSAP-addresses. These node designator values, concatenated with the nominal or a secondary link-id of the local link and the associated selector values — *link-id* || *node* || *selector* — form address spaces of 256 identifiers each, usable as individual DLSAP-addresses. The resulting DLSAP-addresses do not implicitly designate any specific node.

Link-local individual DLSAP-addresses shall be allocated, individually or in contiguous blocks, in order of increasing address values when considered as unsigned 16-bit integers. The first (lowest-valued) such addresses are reserved for use by the DL-protocol as “DLSAP”-addresses; the next 56 such addresses are reserved for standardized individual DLSAP addresses for standard AEs.

- 05 Link-local individual non-DLSAP DL-addresses usable as DLCEP-addresses and DL-SEP-addresses. These node designator values, concatenated with the nominal or a secondary link-id of the local link and the associated selector values — *link-id* || *node* || *selector* — form address spaces of 2 816 identifiers usable as individual DLCEP- and DLSEP-addresses. The resulting DL-addresses do not implicitly designate any specific node.

Link-local individual non-DLSAP DL-addresses usable with connected DL-services shall be allocated, individually or in contiguous blocks, in order of decreasing address values when considered as unsigned 16-bit integers. The first (highest-valued) eight such addresses are reserved for use by the DL-protocol; the next 56 such addresses are reserved for standardized individual DLCEP-addresses or DLSEP-addresses for standard AEs. All of the other addresses are available for use as non-standardized individual DLCEP addresses and DLSEP-addresses.

- 10 Designators of individual nodes. These node designator values, concatenated with the nominal link-id of the local link, provide the equivalent of a “unique hardware address” for each individual physical DLE. Node designators shall be allocated, individually or in contiguous blocks, as shown in Table 4.

Table 4 – Link-local node designators

N	Assigned use for specified node designator range
00	local DLE
01 ... 0F	unusable
10 ... 13	bridge-class DLEs
14 ...	link-master-class DLEs
...	unused
... F7	basic-class DLEs
F8 ... FB	non-“visitor” DLEs awaiting proper node designator assignment
FC ... FF	“visitor” DLEs

- 00 The local DLE. Equivalent to a DLE-internal address. This designator is provided as a convenience for the unification of service primitives, and to enable implementations to refer to the local DLE address by a constant zero (00). When transmission is appropriate, an implementation shall substitute the DLE's own nominal-node-id for this zero (00) address.

NOTE 2 As a consequence, DLPDUs with a node designator component of zero (00) in a DL-address (other than a flat DL-address) are never transmitted, and are discarded upon reception.

- 01 Reserved for unrelated use within the DL-protocol - not available for DLE NODE
... DL-addresses.
0F

- 10 Reserved for allocation to DLEs which bridge (interconnect) the local link (whose link-id is the nominal link-id) to other links, and in particular to higher-capacity links.

- 13 NOTE 3 Such bridges are expected to serve as link-local managers for the DL-protocol.

- 14 Reserved for allocation to other DLEs which are capable of initializing and managing the local link, and which are intended to be permanently attached to the local link (not, for example, maintenance DLEs such as "hand-held communicators").

These node designators shall be allocated, individually or in contiguous blocks, in order of increasing address values when considered as unsigned 8-bit integers.

- ... Reserved for allocation to DLEs which require more local DL-addresses than are provided hierarchically by the DLE's primary node designator.

- ... Reserved for allocation to DLEs which are not capable (or are configured to be incapable) of initializing the local link.
F7

These node designators shall be allocated, individually or in contiguous blocks, in order of decreasing address values when considered as unsigned 8-bit integers.

- F8 Reserved for temporary use by other DLEs which are intended to be permanently attached to the local link while they determine their appropriate final node designator.
...
FB

- FC Reserved for allocation to DLEs which are not intended to be permanently attached to the local link (for example, maintenance DLEs such as "hand-held communicators").

- FF NOTE 4 Such "visitor" DLEs normally are capable of initializing and managing the local link, but are the least desirable such DLEs for this purpose since their long-term presence on the link is not expected.

When not used to designate a single DLE, node designator values between those used for DLEs capable of initializing the local link, and those which are incapable (or are configured to be incapable) of initializing the local link, can be concatenated with the nominal or a secondary link-id of the local link, and with selector components whose value is ≥ 08 and $\leq F7$, to form additional identifier spaces of 240 individual DL-addresses each — *link-id* || *node* || *selector*. Each such additional address space may be assigned to a single node, or may be shared among the nodes of the link.

NOTE 5 The *node* || *selector* components of these additional address identifiers take the hexadecimal form *xxyy*, where $10 < xx < FC$ and $08 \leq yy \leq F7$.

In summary, when the link designator value designates the local link, then each DLE recognizes node designator values 01 – 0F as designating link-local group and individual address spaces — *link-id* || *node* || *selector* — common to all nodes on the local link.

Each DLE recognizes node designator value 00 as designating a single local node designator value ≥ 10 (hexadecimal), the nominal-node. The resulting identifier values — *nominal-link-id* || *nominal-node-id* || *selector* — form a space of identifiers usable as individual DL-addresses as described in 6.3.2.3.

When the address space needs of a given node exceed that provided by a single nominal-node-id, secondary node-ids may be allocated to that node. For each such secondary node id, the resulting values — *link-id* || *secondary-node-id* || *selector* — with selector component values ≥ 08 and $\leq F7$, form an additional space of 240 identifiers usable as additional individual DL-addresses for the node as described in 6.3.2.3. The same method may be used to extend the set of non-node specific individual addresses for the link.

6.3.2.3 Selectors

The selector component of a DL-address usually designates a single protocol entity within the DLE.

When the link designator component does not have the value 0001 – 0FFF, and when the node designator component does not have the value 01 – 0F, then the values for the selector component are predefined, and their usage is constrained, as specified in Table 5.

Table 5 – Node-local selector addressing

Selector	Assigned use for specified DL-address range
00 ...	node-local individual "DLSAP"-addresses reserved for DLL use
01	
02 ...	node-local individual DLSAP-addresses reserved for standard AEs
07	
08 ...	node-local individual DLSAP-addresses available for vendor, user and dynamic assignment
1F	
20 ...	node-local individual DL-addresses available for vendor, user and dynamic assignment as DLCEP-addresses and DLSEP-addresses
F7	
F8 ...	node-local individual DL-addresses reserved for use as standard AE DLCEP-addresses and DLSEP-addresses
FF	

00 Reserved for standard node-related DLSAP-addresses, and for intra-DLE
... "DLSAP"-addresses.
07

08 Non-standardized individual DLSAP-addresses.
...
1F

20 Non-standardized DLCEP-addresses and DLSEP-addresses.
...
F7

F8 Reserved for standard node-related DLCEP-addresses and DLSEP-
... addresses.
FF

6.3.3 Predefined DL-addresses

As indicated in Table 2 through Table 4, some specific DL-addresses are defined in this subclause. These DL-addresses are defined to facilitate provision of DL-services and initialization of OSI communications within a physically distributed real system.

6.3.3.1 Predefined flat non-local DL-addresses

A number of flat non-local DL-addresses are defined within this subclause, as specified in Table 6.

NOTE SLAE is the System Load Application Entity; SMAE is the System Management Application Entity.

Table 6 – Predefined flat non-local DL-addresses

Link N S	Assigned use for specified DL-address
0001 0000	the DL-support functions of “all” (see NOTE) DLEs on the extended link
0001 0001	the DL-support functions of “all” (see NOTE) LM DLEs on the extended link
0001 0002	the DL-support functions of “all” (see NOTE) bridge DLEs on the extended link
0001 0003	the DL-bridge functions of “all” (see NOTE) bridge DLEs on the extended link
0001 0008	the SMAEs of “all” (see NOTE) unconfigured DLEs on the extended link
0001 0009	the SMAEs of “all” (see NOTE) DLEs on the extended link
0001 000A	the SMAEs of “all” (see NOTE) LM DLEs on the extended link
0001 000B	the SMAEs of “all” (see NOTE) bridge DLEs on the extended link
0001 000C	the SLAEs of “all” (see NOTE) LoadServers on the extended link
0001 000D	the SLAEs of “all” (see NOTE) LoadableDevices on the extended link
0080 0000	reserved for DLL use for a DL-support “DLSAP”-address (see NOTE)
0080 0004	the “DLSAP”-address for the DL-bridge functions of the bridge DLE on the extended link which is the root bridge of the spanning tree
0FFF FFFE	the DLCEP-address for the DLC from the bridge functions of the bridge DLE on the extended link which is the root bridge of the spanning tree to the bridge functions in all other bridge DLEs on the extended link
0FFF FFFF	reserved for DLL use for a DL-support DLCEP-address or DLSEP-address (see NOTE)
NOTE DLEs which do not recognize LONG DL-addresses are necessarily excluded from these sets.	

6.3.3.2 Predefined flat local DL-addresses

A number of flat local DL-addresses are defined within this subclause, as specified in Table 7. These correspond one-for-one with the predefined flat non-local DL-addresses specified in Table 6. The extent of the addresses in Table 7 is the local link; the extent of the addresses in Table 6 is the entire extended link.

The DL-addresses in Table 7 have a one-for-one correspondence with analogous DL-addresses in Table 6; while the addresses in Table 7 refer to functions local to each link, the analogous addresses of Table 6 refer to similar functions global to the entire extended link. This addressing correspondence is intentional, and may serve to ease implementation complexity. Thus this correspondence should be preserved in future address assignments.

Table 7 – Predefined flat link-local DL-addresses

Node selector		Assigned use for specified DL-address
01	00	the DL-support functions of all DLEs on the link
01	01	the DL-support functions of all LM DLEs on the link
01	02	the DL-support functions of all bridge DLEs on the link
01	03	the DL-bridge functions of all bridge DLEs on the link
01	08	the SMAEs of all unconfigured DLEs on the link
01	09	the SMAEs of all DLEs on the link
01	0A	the SMAEs of all LM DLEs on the link
01	0B	the SMAEs of all bridge DLEs on the link
01	0C	the SLAEs of all LoadServers on the link
01	0D	the SLAEs of all LoadableDevices on the link
04	00	the “DLSAP”-address for the DL-support functions of the DLE on the link which is serving as LAS
04	04	the “DLSAP”-address for the DL-bridge functions of the bridge DLE on the link which is dominant (closest to the root) in the bridge spanning tree
0F	FE	the DLCEP-address for the DLC from the bridge functions of the bridge DLE on the link which is dominant (closest to the root) in the bridge spanning tree to the bridge functions in all other bridge DLEs on the link
0F	FF	the DLCEP-address for the DLC from the DL-support functions of the DLE on the link which is serving as LAS to the DL-support functions of all of the other LM DLEs on the link

6.3.3.3 Predefined node-local DL-addresses

A number of node-local DL-addresses are defined within this subclause, as specified in Table 8.

Table 8 – Predefined node-local DL-addresses

Selector	Assigned use for specified DL-address
00	the “DLSAP”-address for the DL-support functions of the node’s DLE
01	the “DLSAP”-address for the DL-bridge functions of the node’s DLE
02	the DLSAP-address for the node’s SMAE
03	the DLSAP-address for the node’s SLAE

6.3.4 Representation of DL-Addresses as locally-administered 48-bit MAC-addresses

The ISO/IEC Local Area Network protocols (ISO/IEC 8802) provide a foundation for portions of the IEC fieldbus protocols, such as the Ph-redundancy state machine, the inter-DL-relay-(bridge)-protocol, and the system management system-load-protocol. The latter two protocols, which specify six-octet encodings for each ISO/IEC MAC-address, are being adapted for fieldbus use with the minimum of necessary changes; thus there is a need to specify the representation of fieldbus four-octet DL-addresses for use with the fieldbus adaptations of these protocols.

A six-octet ISO/IEC MAC-address, as specified in ISO/IEC 8802-1, contains, in order of transmission

- a) an initial bit, I/G, indicating whether the MAC-address is an individual MAC-SAP-address (I/G=0) designating a single MAC-sublayer-entity, or a group MAC-address (I/G=1) designating a group of zero or more MAC-sublayer-entities;

- b) a second bit, U/L, indicating whether the MAC-address contains a globally administered component (U/L=0), or is completely locally administered (U/L=1);
- c) a 22-bit globally administered component and a 24-bit vendor-administered component when U/L=0, or a 46-bit user-administered component when U/L=1 (see figure 6.3.4).

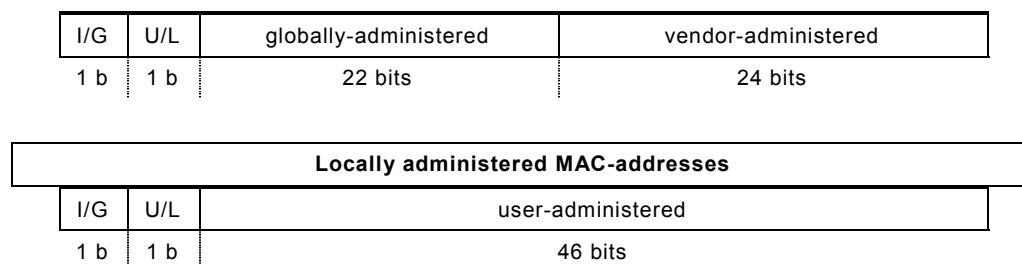


Figure 7 – Basic Structure of MAC-addresses

Since DL-addresses are not administered so as to be globally unique they shall be represented as six-octet locally administered MAC-addresses as shown in Figure 8.

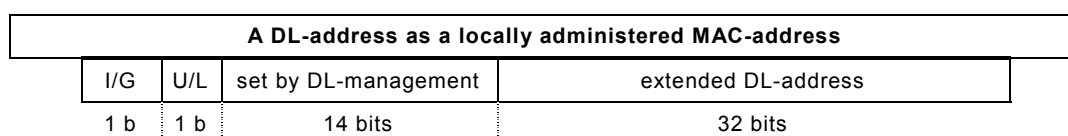


Figure 8 – Representation of a DL-address as a MAC-address

The 14-bit MAC-address component set by DL-management shall be set equal to the value of the extended-link's **DL-MAC-address-embedding-prefix** parameter, V(MEP), which shall be distributed as part of the extended-link's configuration information. The default value for this parameter shall be hexadecimal 3FFF (all ones), but it shall be changeable by DL-management to permit more than one extended link to be directly addressable from inter-working ISO/IEC local area networks.

Each 32-bit DLSAP-, DLCEP- and DLSEP-address shall be represented as a six-octet MAC-address by prefixing the 32-bit non-group DL-address with the binary string "01" followed by the extended-link's 14-bit DL-MAC-address-embedding-prefix, V(MEP).

Each 32-bit group DL-address shall be represented as a six-octet MAC-address by prefixing the 32-bit group DL-address with the binary string "11" followed by the extended-link's 14-bit DL-MAC-address-embedding-prefix, V(MEP).

In addition, should there be a need to specify a six-octet MAC-address for the group of multi-peer subscriber DLCEPs associated with a specified multi-peer publisher DLCEP, then the implicit address of that group of subscriber DLCEPs shall be represented as a six-octet MAC-address by prefixing the publisher's 32-bit individual DLCEP-address with the binary string "11" followed by the extended-link's 14-bit DL-MAC-address-embedding-prefix, V(MEP).

6.4 Service assumed from the PhL

This subclause defines the assumed Physical Service (PhS) primitives and their constraints on use by the DLE.

NOTE Proper layering requires that an (N+1)-layer entity not be concerned with, and that an (N)-service interface not overly constrain, the means by which an (N)-layer provides its (N)-services. Thus the Ph-service interface does not require DLEs to be aware of internal details of the PhE (for example, preamble, postamble and DLPDU delimiter signal patterns, number of bits per baud), and should not prevent the PhE from using appropriate evolving technologies.

6.4.1 Assumed primitives of the PhS

The granularity of transmission in the fieldbus protocol is one octet. This is the granularity of PhS-user data and timing information exchanged at the PhL – DLL interface.

6.4.1.1 PhS characteristics reporting service

The PhS is assumed to provide the following service primitive to report essential PhS characteristics used in DLL transmission, reception, and scheduling activities:

Ph-CHARACTERISTICS indication (minimum-data-rate, framing-overhead)

where

minimum-data-rate — specifies the effective minimum rate of data conveyance in bits/second, including any timing tolerances, of any PhE on the local link.

NOTE 1 A PhE with a nominal data rate of 1 Mbit/s $\pm$ 0,01 % would specify a minimum data rate of 0,9999 Mbit/s.

framing-overhead — specifies the maximum number of bit periods, where $\text{period} = \frac{1}{\text{data rate}}$, used in any transmission for PhPDUs which do not directly convey data (for example, PhPDUs conveying preamble, DLPDU delimiters, postamble, inter-DLPDU “silence”, and so on).

NOTE 2 If the framing overhead is F and two DL message lengths are L₁ and L₂, then the time to send two immediately consecutive messages of lengths L₁ and L₂ will be at least as great as the time required to send one message of length L₁ + F + L₂.

If this framing-overhead is more than the DLE’s configured per-DLPDU-PhL-overhead, V(PhLO), then the DLE shall report this discrepancy to DL-management and shall not issue Ph-DATA requests while the discrepancy exists.

NOTE 3 This restriction prohibits DLE transmission while this discrepancy exists. The DLE’s local station management may remedy this discrepancy by reconfiguring V(PhLO) to a greater value.

6.4.1.2 PhS transmission and reception services

The PhS is assumed to provide the following service primitives for transmission and reception:

Ph-DATA request (class, data);

Ph-DATA indication (class, data);

Ph-DATA confirm (status)

where

class — specifies the Ph-interface-control-information (PhICI) component of the Ph-interface-data-unit (PhIDU). For a Ph-DATA request, its possible values are

START-OF-ACTIVITY — transmission of the PhPDUs which precede Ph-user data should commence;

DATA — the single-octet value of the associated data parameter should be transmitted as part of a continuous correctly-formed transmission;

END-OF-DATA-AND-ACTIVITY — the PhPDUs which terminate Ph-user data should be transmitted after the last preceding octet of Ph-user data, culminating in the cessation of active transmission;

For a Ph-DATA indication, its possible values are:

START-OF-ACTIVITY — reception of an apparent transmission from one or more PhEs has commenced;

DATA — the associated data parameter was received as part of a continuous correctly-formed reception;

END-OF-DATA — the ongoing continuous correctly formed reception of Ph-user data has concluded with correct reception of PhPDUs implying END-OF-DATA;

END-OF-ACTIVITY — the ongoing reception (of an apparent transmission from one or more PhEs) has concluded, with no further evidence of PhE transmission;

END-OF-DATA-AND-ACTIVITY — simultaneous occurrence of END-OF-DATA and END-OF-ACTIVITY;

data — specifies the Ph-interface-data (PhID) component of the PhIDU. It consists of one octet of Ph-user data to be transmitted (Ph-DATA request) or which was received successfully (Ph-DATA indication).

status — specifies either success or the locally detected reason for inferring failure.

The Ph-DATA confirm primitive provides the critical physical timing feedback necessary to inhibit the DLE from starting a second transmission before the first is complete. The final Ph-DATA confirm of a transmission shall not be issued until the PhE has completed the current transmission.

6.4.2 Notification of PhS characteristics

The PhE has the responsibility for notifying the DLE of those characteristics of the PhS which are relevant to DLE operation. This notification is accomplished by the PhE by issuing a single Ph-CHARACTERISTICS indication primitive at each of the PhE's PhSAPs at PhE startup.

6.4.3 Transmission of Ph-user data

The PhE determines the timing of all transmissions. When a DLE has a DLPDU to transmit, and the DL-protocol gives that DLE the right to transmit, then the DLE shall send the DLPDU, including a concatenated FCS, by making a well-formed sequence of Ph-DATA requests, consisting of a single request specifying START-OF-ACTIVITY; followed by three to 300 consecutive requests, inclusive, specifying DATA; and concluded by a single request specifying END-OF-DATA-AND-ACTIVITY.

The PhE signals its completion of each Ph-DATA request, and its readiness to accept a new Ph-DATA request, with a Ph-DATA confirm primitive; the status parameter of the Ph-DATA confirm primitive conveys the success or failure of the associated Ph-DATA request. A second Ph-DATA request should not be issued until after the Ph-DATA confirm corresponding to the first request has been received from the PhE.

6.4.4 Reception of Ph-user data

The PhE reports a received transmission with a well-formed sequence of Ph-DATA indications, which shall consist of either

- a) a single indication specifying START-OF-ACTIVITY; followed by consecutive indications specifying DATA; followed by a single indication specifying END-OF-DATA; and concluded by a single indication specifying END-OF-ACTIVITY, or

- b) a single indication specifying START-OF-ACTIVITY; followed by consecutive indications specifying DATA; followed by a single indication specifying END-OF-DATA-AND-ACTIVITY, or
- c) a single indication specifying START-OF-ACTIVITY; optionally followed by one or more consecutive indications specifying DATA; and concluded by a single indication specifying END-OF-ACTIVITY.

NOTE This last sequence is indicative of an incomplete or incorrect reception. Detection of an error in the sequence of received PhPDUs, or in the PhE's reception process, disables further Ph-DATA indications with a class parameter specifying DATA, END-OF-DATA, or END-OF-DATA-AND-ACTIVITY until after both the end of the current period of activity and the start of a subsequent period of activity have been reported by Ph-DATA indications specifying END-OF-ACTIVITY and START-OF-ACTIVITY, respectively.

In the first two cases, the DLE concatenates the received data and then attempts to parse it into a DLPDU followed by a concatenated FCS. In the last case the DLE discards all reported data and reports the event to DL-management.

6.5 Functions of the DLL

6.5.1 Overview of functions

The functions of the DLL are those necessary to bridge the gap between the services available from the PhL and those offered to DLS-users. When used in a normal OSI environment, the necessary functions of the DLL are those specified in ISO/IEC 8886. When used in a Time Critical OSI environment, the necessary functions are a superset of those specified in ISO/IEC 8886; the enhancements are primarily in

- a) the availability of a confirm primitive and a service confirmation deadline for each connection-oriented and connectionless data-transfer DL-service;
- b) the addition of connectionless receiving-DLE-acknowledgment and two-way data-exchange services to the basic OSI data-transfer DL-service;
- c) the ability to defer and schedule data-transfer DL-services;
- d) the efficient distribution of DLS-user data from a publishing DLS-user to a set of subscribing DLS-users;
- e) the efficient convergence of independent DLS-user data from a set of subscribing DLS-users, delivered to a single publishing DLS-user;
- f) the provision of a synchronized sense of internal time among the DLEs and available to the DLS-users of the extended link;
- g) the standardized availability of local DL(SAP)-address, buffer and queue management capabilities.

6.5.2 Connectionless data transfer functions

The purpose of the connectionless data transfer functions is to transport DLSDUs of limited size between one DLS-user and one or more other DLS-users on the same link, without the requirement for establishing or maintaining a DLC with each of those other DLS-users. This purpose is achieved by means of transmission of DLPDUs providing transfer of a limited amount of user data to one or many DLS-users, with limited protection against loss of the DLSDU, or misordering of successively-transmitted DLPDUs. Both DLSDU transfer and DLSDU exchange services are provided.

6.5.3 Connection-oriented functions

Connection-oriented functions provide for the establishment, use, resynchronization and abrupt termination of a connection between DLS-users on an extended link. The type of the connection may be selected to support user data flow either

- a) bidirectionally between two peer DLS-users,
- b) unidirectionally from one peer DLS-user to another,
- c) bidirectionally between one publishing DLS-user and the set of zero or more subscribing DLS-users,

- d) unidirectionally from one publishing DLS-user to the set of zero or more subscribing DLS-users, or
- e) unidirectionally from the set of zero or more subscribing DLS-users to the publishing DLS-user.

The features of the connection may be selected to support transfer of DLSDUs of a negotiated maximum size, with either

- 1) reliable in-sequence non-duplicated delivery with reset on DLSDU loss;
- 2) reduced delay, potentially out-of-sequence but non-duplicated delivery, and reset on DLSDU loss;
- 3) minimal delay with in-sequence non-duplicated delivery, but with potential DLSDU loss;
- 4) minimal delay, unsequenced delivery, potential duplication, potential but improbable DLSDU misordering, and no notification of DLSDU loss; or
- 5) no transfer in one specified direction.

Only cases 4) and 5) are permitted for the subscriber-to-publisher direction of a multi-peer DLC.

6.5.3.1 Connection establishment phase

The purpose of the connection establishment phase is

- a) to establish a DLC between two DLS-users,

NOTE The establishment of a publishing DLC is best modeled as the concurrent independent pair-wise establishment of the DLC between the common publisher and each separate subscriber.

- b) to determine QoS attributes of the DLC;
- c) to distinguish between DLCs.

6.5.3.2 Data transfer phase

The purpose of the data transfer phase is

- a) to transport DLSDUs between two DLS-users connected by a DLC. This purpose is achieved by transmission of DATA (DT) and related DLPDUs (see 8.7, 8.6 and 8.4), which may involve segmenting of DLSDUs for conveyance in multiple DLPDUs and reassembly by the destination DLE;
- b) to resynchronize the flow of DLSDUs between the DLS-users, and notify those DLS-users of information loss after an unrecovered error.

These concepts are defined in IEC 61158-3.

6.5.3.3 Connection termination phase

The purpose of the connection termination phase is to terminate abruptly a connection between two or more DLS-users and convey the reason for the termination.

6.5.4 Time synchronization function

The purpose of the time synchronization function is to provide a shared approximately-synchronized internal time reference for all DLS-users, consisting of two components:

- a) a component which increases monotonically with time, with a value of zero at the startup of the local end system;
- b) a second component which, when added to the first, causes the sum to be approximately equal to the corresponding sums of other correctly-functioning DLEs on the extended link.

6.6 Functional classes

In this standard, a DLE's functional class determines its capabilities for autonomous DLL activity, and thus the minimum complexity of conforming implementations. Each class includes all lower-numbered classes. The three functional classes, in order of increasing complexity, are

- a) Basic;
- b) Link Master (LM);
- c) Bridge.

All functional classes support all DLS-user services and are completely interoperable. The DLPDUs associated with elements of procedure for each functional class are specified in Table 9.

Table 9 – Correlation of DLPDUs with functional classes

Functional class			Code	DLPDU name	Subclause
Basic	Link Master	Bridge			
R ¹ S ¹	R ¹ S ¹	R ¹ S ¹	EC	ESTABLISH CONNECTION	8.1
R ¹ S ¹	R ¹ S ¹	R ¹ S ¹	DC	DISCONNECT CONNECTION	8.2
R ¹ S ¹	R ¹ S ¹	R ¹ S ¹	RC	RESET CONNECTION	8.3
R S	R S	R S	CA	COMPEL ACKNOWLEDGMENT	8.4
R S	R S	R S	CD	COMPEL DATA	8.5
R S	R S	R S	ED	EXCHANGE DATA	8.6
R S	R S	R S	DT	DATA	8.7
R —	R S	R S	SR	STATUS RESPONSE	8.8
— s ²	R s ²	R s ²	CT	COMPEL TIME	8.9
r ² —	R ² S	R ² S	TD	TIME DISTRIBUTION	8.10
R s ²	R s ²	R s ²	RQ	ROUND-TRIP-DELAY QUERY	8.11
r ² S	r ² S	r ² S	RR	ROUND-TRIP-DELAY REPLY	8.12
R —	R S	R S	PN	PROBE NODE DL-ADDRESS	8.13
— S	R ³ S	R ³ S	PR	PROBE RESPONSE	8.14
R —	R S	R S	PT	PASS TOKEN	8.15
R —	R S	R S	ES	EXECUTE SEQUENCE	8.16
— S	R ³ S	R ³ S	RT	RETURN TOKEN	8.17
— S	R ³ S	R ³ S	RI	REQUEST INTERVAL	8.18
— —	R S	R S	CL	CLAIM LAS	8.19
— —	R S	R S	TL	TRANSFER LAS	8.20
r —	r S	— S	WK	WAKEUP	8.21
— s ⁴	— s ⁴	— s ⁴	IDLE	IDLE	8.22
— —	— —	— —	spare	SPARE	8.23

Key

r The specified class permits the ability to receive and act upon the specified type of DLPDU.

R The specified class always provides the ability to receive and act upon the specified type of DLPDU.

s The specified class permits the ability to send the specified type of DLPDU.

S The specified class always provides the ability to send the specified type of DLPDU.

— The specified class does not permit the specified type of DLPDU.

NOTE 1 Mandated support requires only minimal elements of procedure.

NOTE 2 All shall be supported when the DLE has a time-synchronism class (see 12.6.3 other than NONE).

NOTE 3 Shall respond to DLSDU contents when acting as LAS.

NOTE 4 No class is required to receive the IDLE DLPDU; its intended receiver is a specialized link monitor or analyzer, beyond the scope of this standard.

6.6.1 Basic class

The Basic class includes the basic protocol elements of procedure necessary

- a) to provide interoperability when responding to DLPDUs sent by a DLS-peer or a bridge DLE;
- b) to initiate, reset and terminate DLCs with a peer DLE, to support the orderly conveyance of DLSDUs;
- c) to send and receive connectionless and connection-oriented DLSDUs, and to reply to received DLSDUs where required;
- d) to request services from the LAS;
- e) to execute an uninterrupted sequence of link operations;
- f) to optimize local use of the link.

This class is the minimum necessary for fieldbus interoperability.

6.6.2 Link Master class

The Link Master class includes all of the functions of the Basic class. It also includes the protocol elements of procedure necessary

- a) to cooperate with similar DLEs in establishing and sharing mastership of the link;
- b) to detect the absence of an LAS on the link and activate the LAS functions within its own node,

and when providing the functions of the LAS,

- c) to maintain an ordered access to the shared link communications resource, responding to requests from other DLEs for use of that shared resource;
- d) to serve as the source of internal time for the other DLEs on the link.

This class is necessary for autonomous operation on the link. At least one DLE on the link shall operate in this class.

6.6.3 Bridge (DL-relay) class

The Bridge class includes all of the functions of the Link Master class. It also includes the protocol elements of procedure necessary

- a) to enable communications between DLEs on a single link which are themselves periodically incapable of communicating directly on the link (that is, fractional duty cycle (FDC) DLEs), and in some cases serving as a surrogate for a DLE on the local link;
- b) to interconnect two or more local links, by bridging them into an extended link, and in some cases serving as a surrogate for a remote DLE on one of the bridge's local links;
- c) to provide a common sense of DL-internal time coordinated across the extended link.

This class is necessary when interconnecting two or more local links to form a multi-link extended link, or when one or more DLEs on the local link are fractional duty cycle (FDC) DLEs. When a multi-link extended link exists, the individual local links shall be interconnected only by DLEs which operate in this class.

6.7 Local parameters, variables, counters, timers and queues

This standard uses DLS-user request parameters $P(\dots)$ and local variables $V(\dots)$ as a means of clarifying the effect of certain actions and the conditions under which those actions are valid, local timers $T(\dots)$ as a means of monitoring actions of the distributed DLS-provider and of ensuring a local DLE response to the absence of those actions, and local counters $C(\dots)$ for performing rate measurement functions. It also uses local queues $Q(\dots)$ as a means of ordering certain activities, of clarifying the effects of certain actions, and of clarifying the conditions under which those activities are valid.

Unless otherwise specified, at the moment of their creation or of DLE activation,

- all variables shall be initialized to their default value, or to their minimum permitted value if no default is specified;
- all counters shall be initialized to zero;
- all timers shall be initialized to inactive;
- all queues shall be initialized to empty.

DL-management may change the values of configuration variables.

6.7.1 Parameters, variables, counters, timers and queues to support the Basic class

The parameter, variables, timer, counter and queue defined in 6.7.1.1 to 6.7.1.18 are required in all DLEs. The variables, counter and timer defined in 6.7.1.19 to 6.7.1.27 are required in all DLEs except those with a time synchronism class (see 12.6.3) of NONE.

6.7.1.1 $V(ST)$ slot-time

$V(ST)$ is used by the DLE to record the link's slot-time (see 3.4.20), which is a configuration parameter of the local link. Its range is 1 to 4 095, and its unit is the transmission duration of one octet.

6.7.1.2 $V(PhLO)$ per-DLPDU-PhL-overhead

$V(PhLO)$ is used by the DLE to account for the PhL-induced delay between the end of the last octet of one DLPDU as it appears on the link, and the beginning of the first octet of any other DLPDU as it appears on the link, measured in units of one octet-duration. Its range is 2 to 63.

6.7.1.3 $V(MRD)$ maximum-response-delay

$V(MRD)$ is used by the DLE to record the link's maximum-response-delay (see 3.4.13), which is a configuration parameter of the local link. The default value for this variable is 3. Its range is 1 to 11, and its unit is one slot-time.

NOTE This unit is chosen so that all measurements of an inactive link are for multiples of one slot-time.

6.7.1.4 $V(IRRd)$ immediate-response-recovery-delay

$V(IRRd)$ is used by the DLE to record the link's immediate-response-recovery-delay (see 3.4.8), which is a computed parameter of the local link. The default value for this variable is $V(MRD) + 1$. Its range is 2 to 12, and its unit is one slot-time.

6.7.1.5 $V(MRC)$ maximum-retry-count

$V(MRC)$ is used by the DLE to record the link's maximum-retry-count, which limits the number of immediate retries of a transaction which are permitted. $V(MRC)$ is a configuration parameter of the local link. Its default value is 0, meaning immediate retries are not permitted. Its range is 0 to 7.

6.7.1.6 V(NRC) network-repeat-count

V(NRC) is used by the DLE to record the extended link's network-repeat-count, which specifies the maximum number of deferred retries of a multi-link communication which should be attempted. V(NRC) is a configuration parameter of the extended link, based to some extent on the values of V(MRC) for the various local links of the extended link. Its default value is 1, meaning a single deferred retry should be attempted when other considerations do not intervene. Its range is 0 to 7.

6.7.1.7 V(NDL) network-DLPDU-lifetime

V(NDL) is used by the DLE to record the extended link's network-DLPDU-lifetime, which specifies the maximum period that any DLPDU can remain in transit within the extended link. Its range is 1 ms to 60 s, in units of 1 ms, and its default value is 30 s.

6.7.1.8 V(TN) this-node

V(TN) is used by the DLE to record the local node-id (see 3.4.16), which is a configuration parameter of the local DLE. Its default uninitialized value is zero, which does not permit the DLE to transmit. Its range is 0, and 10_{16} to FF_{16} (see 6.3.2.2).

6.7.1.9 V(TL) this-link

V(TL) is used by the DLE to record the local link-id (see 3.4.11), which is a configuration parameter of the local link. Its default uninitialized value is zero, which does not permit a bridge to forward to another link. Its range is 0, and 1000_{16} to $FEFF_{16}$ (see 6.3.2.1).

6.7.1.10 V(MEP) DL-MAC-address-embedding-prefix

V(MEP) is used by the DLE (see 6.3.4) to embed the extended-link's 32-bit address space within the 48-bit address space used by the ISO/IEC LAN (local area network) standards (ISO/IEC TR 8802-1). The default value for this variable is zero. Its range is 0 to $3FFF_{16}$.

6.7.1.11 C(RD) remaining-duration counter

C(RD) is used by the DLE to record the remaining duration of token delegation when using a token delegated by a PASS TOKEN (PT) or EXECUTE SEQUENCE (ES) DLPDU, or the remaining link capacity until the next scheduled activity when using a scheduler token during link maintenance activity. C(RD) is initialized on receipt of such a DLPDU from the DLPDU's DD-parameter and decrements at the rate of at least one count per octet of elapsed transmission capacity until it reaches the value zero or until the token is returned.

NOTE The phrase "at least" permits implementations to approximate C(RD) provided that they are either accurate or pessimistic in estimating the actual time required for decision-making and for preparing and sending a transmission. This permission may reduce the number of counters required in an actual implementation.

6.7.1.12 V(MID) minimum-inter-DLPDU-delay

V(MID) is used to specify the minimum duration for an interval of non-transmission which a transmitting DLE shall provide after receiving, or transmitting, a DLPDU. This interval is measured in units of the transmission duration of one octet; the measurement interval begins with either

- a) the receipt of a PH-DATA confirm primitive by the DLE which confirms a PH-DATA request primitive which specified END-OF-DATA-AND-ACTIVITY, OR
- b) the receipt of a PH-DATA indication primitive by the DLE specifying END-OF-DATA, END-OF-DATA-AND-ACTIVITY, OR END-OF-ACTIVITY;

whichever occurs first, and ends with the submission of a PH-DATA request primitive by the DLE specifying start-of-activity. The default value for this variable is 0, and its range is 0 to the smaller of 120 or $(V(MRD)-1) \times V(ST)$.

NOTE The PhL may specify a minimum inter-transmission delay of its own. Such a constraint is independent of, and may apply concurrently with, the constraint specified above.

6.7.1.13 T(IRRd) immediate-response-recovery-delay monitor

T(IRRd) is used by the DLE which initiates a two-DLPDU transaction to monitor the local link for an anticipated immediate reply DLPDU, and to reassert its own token as the active token on the link if no reply is detected.

6.7.1.14 V(RA) reply-address

V(RA) is used by a DLE to record the DL-address to which a reply token was last passed by a COMPEL ACKNOWLEDGMENT (CA), COMPEL DATA (CD), or EXCHANGE DATA (ED) DLPDU. It also can be used by the LAS DLE to record the DL-address to which a reply token was last passed by a PROBE NODE-ADDRESS (PN) DLPDU.

6.7.1.15 V(OTA) outstanding-transaction-array

V(OTA) is used by the DLE, in transactions requiring an immediate reply, to correlate an immediate or delayed reply DLPDU with the initiating DLS-user request. It is also used by the DLE to determine, for any given transaction index, whether that index is currently assigned by the DLE to an incomplete outstanding transaction. V(OTA) is an array with an index range of 0..F₁₆.

6.7.1.16 V(LTI) last-transaction-index

V(LTI) is used by the DLE, in transactions requiring an immediate reply, to ensure that a new transaction does not reuse the index of the immediately previous transaction initiated by the same DLE. Its range is 0.. F₁₆, and its initial value is F₁₆.

6.7.1.17 Q(US) unscheduled-service queue

Q(US) is used by the DLE to manage

- a) **references** to the DLE's DL-address user-request queues (Q_A(UR) for all A within the DLE (see 6.7.3.1), including
 - 1) **references** which signal the need to compel a DLSDU transmission from the correspondent peer or publisher DLCEP, when A is a peer or subscriber DLCEP-address;
 - 2) **references** which signal the need to compel an instance of the unitdata-exchange service with the specified DLSAP-address, when A is a DLSAP-address whose DL(SAP)-role is INITIATOR;
 - 3) **references** which signal the need to transmit the contents of a sending buffer bound to a local DLCEP, when A is a local DLCEP-address;
 - 4) **references** which signal the need to send a DLSDU from the sending queue bound to local DLCEP or DLSAP-address, when A is a local DLCEP-address or DLSAP-address;
- b) **references** to locally-scheduled active sequences (see clause 10) resulting from DL-SCHEDULE-SEQUENCE requests (see 10.4.3.1);
- c) DT DLPDUs containing DLS-user data which are delayed responses to received CD and ED DLPDUs, queued in support of the DL-UNITDATA-EXCHANGE service.

This queue is used upon receipt of a PASS TOKEN (PT) DLPDU addressed to the DLE (see 3.4.15 and 7.2.2.3). The structure of this queue is described in 3.3.2.

6.7.1.18 V(RID) random identifier

V(RID) is an identifier, chosen from an approximately uniform random distribution, used by the DLE in responding to a Probe NODE-address (PN) DLPDU addressed to the DLE, to introduce a random element into the response, which in turn is used to enable the DLE to validate a subsequently received node-activation SPDU addressed to the DLE.

Each time it is used in a Probe Reply (PR) DLPDU, before its value is copied into the PR DLPDU, the variable shall be set to a new random value chosen uniformly from the range 0 to FF₁₆. The actual random choice shall be statistically independent of similar choices made by other DLEs.

NOTE This requirement for statistical independence minimizes the probability of repeatedly identical choices by identically-constructed real devices, and thus provides the basis for eventual discrimination among multiple DLEs which happen to respond to the same NODE DL-address, either through fault or misconfiguration.

6.7.1.19 C(NT) node-time counter

C(NT) is used by the DLE to record the monotonically increasing component of the DLE's local node-time (see 3.4.24). In the absence of any corrective frequency adjustment (see 10.4.1.3), its nominal long-term rate of counting shall result in incrementing 8 192 000 units/s ($= 2^{13} \times 10^3$ units/s). Its initial value shall be zero. C(NT) shall be sized not to roll over before reinitialization.

NOTE When other data is lacking, implementors may assume that the maximum period between reinitializations in industrial environments is five years. The specified rate of counting does not imply the granularity of the counting, which depends upon the implementation and the time synchronization capability of the DLE.

6.7.1.20 V(LSTO) local-link-scheduling-time-offset

V(LSTO) is used by the DLE to record the signed offset (difference) between the sum $N(NT) + N(LSTO)$ of the local LAS's local-link-scheduling-time, as received in a TIME DISTRIBUTION (TD) DLPDU, and the receiving DLE's C(NT), so that $V(LSTO) = N_{LAS}(LSTO) + N_{LAS}(NT) - C(NT)$. The initial value of V(LSTO) shall be zero.

6.7.1.21 V(DLTO) DL-time-offset

V(DLTO) is used by the DLE to record the signed offset (difference) between the DL-time and the local-link-scheduling-time, such that $DL-time = C(NT) + V(LSTO) + V(DLTO)$. This variable is also received in a TIME DISTRIBUTION (TD) DLPDU. Its initial value shall be zero.

6.7.1.22 V(TQ) time-quality

V(TQ) is used by the DLE to record the multi-partite quality of both the DLE and the source and path of time distribution on the extended link. Its initial value shall indicate a local time source.

6.7.1.23 V(MD) measured-delay

V(MD) is used by the DLE to record the filtered measured delay in two-way communications between the DLE and the current LAS, as measured with a series of ROUND-TRIP-DELAY QUERY / ROUND-TRIP-DELAY REPLY DLPDUs. Its initial value, and its value when the DLE's own LAS functions are active, shall be zero. Its value is invalidated under the conditions specified in 10.4.1.3.

6.7.1.24 V(LN) LAS-node

V(LN) is used by the DLE to record the node-id (see 3.4.11) of the local link's LAS, as received in a TD (TIME DISTRIBUTION) DLPDU (see 10.4.1.3), or zero if the local link's LAS' node-id is unknown. Its initial value shall be zero. V(LN) is used to determine the validity and relevancy of the computed value of V(MD). Its range is the same as that of V(TN) (see 6.7.1.8).

6.7.1.25 V(TSC) time-synchronization-class

V(TSC) is used by the DLE to record the link's time-synchronization-class and to determine the DLE's own requirement for the minimum time distribution period which the DLE requires to maintain the specified level of time synchronization. V(TSC) shall specify one of the time-synchronization-classes defined in 12.6.3. Its default value shall be the 10 ms time-synchronization class (see 12.6.3).

6.7.1.26 T(TDP) time-distribution-period monitor

T(TDP) is used by the DLE to measure the time elapsed since the last sent or received TIME DISTRIBUTION (TD) DLPDU.

6.7.1.27 V(TSL) time-source-link

V(TSL) is used by the DLE to record the link-id of the source of time distribution on the extended link. Its range is the same as that of V(TL). Its initial value shall be zero.

6.7.2 Parameters and timers to support a DLS-user's request

Each specific instance of a DLS-user service request or response may have associated parameters and timers, depending on the type of request or response.

6.7.2.1 P_U(SDUL) DLSDU-length request parameter

P_U(SDUL) designates the length of the DLSDU associated with the specified DLS-user request or response.

6.7.2.2 P_U(SDU) DLSDU request parameter

P_U(SDU) designates the DLSDU associated with the specified DLS-user request or response. P_U(SDU) is conceptualized as an array of P_U(SDUL) octets, indexed from 1 to P_U(SDUL).

6.7.2.3 P_U(MCD) maximum-confirm-delay parameter

P_U(MCD) designates a DLS-user-established maximum confirm delay for a specific request, where the class of delay is always determined by associated text within this part of IEC 61158.

6.7.2.4 T_U(MCD) maximum-confirm-delay monitor

T_U(MCD) is used by the DLE to monitor the completion of the actions associated with the specified DLS-user request, to ensure that a corresponding DLS confirm is given to the DLS-user within the interval of time allotted by the DLS-user, as specified in the DLS-user-established maximum confirm delay for the specific request.

6.7.3 Queues to support DL-address-based DL-scheduling

Each specific instance of a DLSAP-address or DLCEP-address or subscriber DLCEP within the DLE shall have an associated user-request queue (see 3.3.1 and 6.7.3.1).

6.7.3.1 Q_A(UR) user-request queue

Each specific instance of Q_A(UR) is used by the DLE to manage DLS-user requests and responses which require DLPDU transmissions originating at the associated DL-address or subscriber DLCEP. The structure of this queue is described in 3.3.1.

6.7.4 Variables and timers to support a DLCEP

The state of each DLCEP is maintained in variables and timers specific to that DLCEP. The specific set of variables and timers required for a given DLCEP is dependent on the DLCEP's class and data delivery features.

As shown in Figure 9, the following relationships exist among the variables L (see 6.7.4.10) and H (see 6.7.4.11) and the negotiated receiving window size W_R at a receiving DLCEP, and the variables N (see 6.7.4.3), R (see 6.7.4.4), A (see 6.7.4.5), M (see 6.7.4.6), and the negotiated sending window size W_S (see 6.7.4.2) at the corresponding sending DLCEP:

If "SC"-subscripts designate a sending DLE of a DLC, and "RC"-subscripts designate a receiving DLE of the same DLC, then the relationships among the actual DLCEP variables, $V_C(\dots)$, and negotiated parameters, $P_C(\dots)$, from the point of view of a single direction of data transfer, are

$$P_{RC}(W_{RC}) = P_{SC}(W_{SC}) = W$$

$$V_{RC}(H)-W \leq V_{SC}(M)-W \leq V_{SC}(A) \leq V_{RC}(L) \leq V_{RC}(H) \leq V_{SC}(M) \leq V_{SC}(A)+W \leq V_{RC}(L)+W$$

$$V_{SC}(M)-W \leq V_{SC}(R) \leq V_{SC}(M) < V_{SC}(N)$$

$$V_{RC,K}(RRS) \subseteq V_{RC,K}(MRS)$$

6.7.4.1 $V_C(ST)$ DLCEP state

$V_C(ST)$ is used by a DLE to maintain the current user state of the DLCEP. These states are the same as those shown in Figure 13.

$$W_R = W_S = W$$

$$H-W \leq M-W \leq A \leq L \leq H \leq M \leq A+W \leq L+W$$

$$M-W \leq R \leq M < N$$

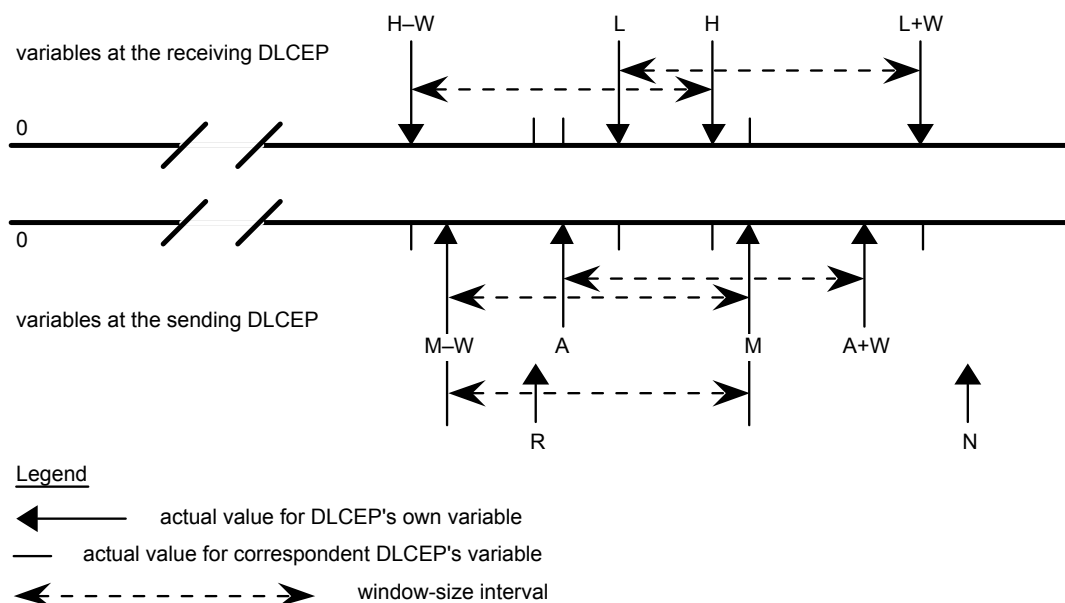


Figure 9 – Linear relationships of sending and receiving DLCEP sequence-number variables

6.7.4.2 $V_C(NP)$ negotiated DLCEP parameters

$V_C(NP)$ is used by a DLE to maintain the current DLCEP parameters as a single structured variable, both during and after the DLCEP-establishment process. These parameters are defined by the EC-parameters (see 9.1) of the ESTABLISH CONNECTION (EC) DLPDU.

Within this part of IEC 61158, references to specific parameters within the EC parameters are denoted by the syntax “V.F”, where V is a $V_C(NP)$ variable and F is a field of the EC parameters as labeled in 9.1.

6.7.4.3 $V_C(N)$ next sequence number to assign to a DLSDU

$V_C(N)$ is used by a DLE to maintain the ordinal number one greater than the number of DLSUDUs whose transmission on the DLCEP has been requested by the DLS-user. Its value at DLCEP establishment is one, and it increases monotonically each time the transmission of a new DLSDU is requested by the DLS-user. It does not increase when a DLS-user’s request for transmission from a buffer (which must be on a DLCEP whose sending data delivery features are UNORDERED or ORDERED) results in the retransmission of a previously transmitted DLSDU.

NOTE This last requirement may be realized by a two-phase incrementation process, in which writing to the buffer arms the incrementation process for all DLCEPs which send from the buffer, and incrementation for a given DLCEP occurs during DLPDU formation immediately prior to transmission (but not retransmission) of the first segment of the buffer on that DLCEP.

6.7.4.4 $V_C(R)$ maximum non-transmittable DLSDU sequence number

$V_C(R)$ is used by a DLE for a DISORDERED or CLASSICAL PUBLISHER DLCEP, and optionally for an ORDERED PEER or PUBLISHER DLCEP, to maintain the ordinal number of the highest-numbered DLSDU which is no longer available for DLE transmission or retransmission, or zero when there is no such DLSDU. Its value at DLCEP establishment is zero, and it is non-decreasing. It increments each time a DLSDU, for which transmission was requested, becomes unavailable to the DLE for retransmission, either because the DLSDU has been queued within the DLL for longer than its specified maximum confirm delay, or because the DLSDU’s sequence number differs from the highest sequence number transmitted by the sending window size $V_C(W_S)$, or due to DLC reset.

NOTE Incrementing $V_C(R)$ may cause a corresponding number of requests to move from the third to the second partition of the relevant $Q_A(UR)$.

6.7.4.5 $V_C(A)$ maximum acknowledged DLSDU sequence number

$V_C(A)$ is used by a DLE for a DISORDERED or CLASSICAL PEER DLCEP to maintain the ordinal number of the highest-numbered DLSDU transmitted from this DLCEP which has been acknowledged by the correspondent DLE. Its value at DLCEP establishment is zero, and it is non-decreasing. A newly-received CA, CD, DT, ED or RC DLPDU may cause it to be increased.

NOTE Incrementing $V_C(A)$ may cause a corresponding number of requests to move from the third to the second partition of the relevant $Q_A(UR)$.

6.7.4.6 $V_C(M)$ maximum transmitted DLSDU sequence number

$V_C(M)$ is used by a DLE for a sending DLCEP to maintain the ordinal number of the highest-numbered DLSDU which has at least one segment which is being, or has been, transmitted from this DLCEP. Its value at DLCEP establishment is zero, and it increases monotonically during the initial transmission of the first segment of each DLSDU.

6.7.4.7 $V_C(MS)$ maximum transmitted DLSDU segment number

$V_C(MS)$ is used by a DLE for an ORDERED, DISORDERED or CLASSICAL sending DLCEP to maintain the zero-based ordinal number of the highest-numbered segment of the $V_C(M)$ 'th DLSDU which is being, or has been, transmitted from this DLCEP. Its initial value at DLCEP establishment, before any DLSDU has been transmitted, is zero.

6.7.4.8 $V_{C,K}(SS)$ segments to send

$V_{C,K}(SS)$ is used by a DLE for a sending DLCEP to maintain the set of segments of the K 'th DLSDU which still need to be transmitted. For each new (N 'th) DLSDU presented by the DLS-user for transmission, a new list $V_{C,N}(SS)$ is created.

For $A < K \leq M$ and $R < K < N$, additions are made to the list upon receipt of a retransmission request for a segment of the K 'th DLSDU, or upon timeout of the corresponding timer $T_{C,K}(SS)$ (or upon timeout of the timer $T_C(SS)$ when it applies to the K th DLSDU, when the permission of 6.7.4.9.1 is employed).

NOTE The lowest-numbered segment of the lowest-numbered DLSDU remaining is usually the next to be transmitted; the DLE deletes this segment from the list upon transmission. For ORDERED or PUBLISHER sending DLCEPs, the first untransmitted segment of the first not-completely-transmitted DLSDU may be sent instead, to maintain an assured rate of delivery of new DLSDUs on periodic DLCs.

6.7.4.9 $T_{C,K}(SS)$ sent-segments monitor

$T_{C,K}(SS)$ is used by a DLE for a DISORDERED or CLASSICAL PEER DLCEP to monitor timely peer acknowledgment of the K 'th transmitted DLSDU, where $A < K \leq M$. It is started when the DLSDU has been transmitted in its entirety, and is cancelled when the DLSDU is acknowledged. Upon its timeout, the last segment of the K 'th DLSDU is again added to $V_{C,K}(SS)$, causing its retransmission and potentially triggering the receiving DLE to request retransmission of any lower-numbered segments which also have been missed.

6.7.4.9.1 $T_C(SS)$ simplified sent-segments monitor

It is permitted to replace the set of per-DLSDU timers $\{ T_{C,K}(SS) \}$ with a single per-DLCEP-timer, $T_C(SS)$. Such a replacement will lead to equivalent DL-service when the number of outstanding DLSDUs is limited by other constraints, such as a small negotiated the window size, or when the residual error rate of the communications path is very low. When such constraints do not apply, then this replacement may lead to inferior DL-service.

If such a replacement is made, then the following definition applies:

$T_C(SS)$ is used by a DLE for a DISORDERED or CLASSICAL PEER DLCEP to monitor timely peer acknowledgment of transmitted DLSDUs. It is started, but not restarted, when a DLSDU has been transmitted in its entirety, and is cancelled when a DLSDU is acknowledged. It is then restarted if there are still unacknowledged DLSDUs.

Upon its timeout, the last segment of the unacknowledged $DLSDU_K$ with the lowest sequence number is again added to the $V_{C,K}(SS)$ corresponding to that DLSDU, causing its retransmission and potentially triggering the receiving DLE to request retransmission of any lower-numbered segments which also have been missed.

6.7.4.10 $V_C(L)$ last-reported DLSDU sequence number

$V_C(L)$ is used by a DLE for an ORDERED, DISORDERED or CLASSICAL receiving DLCEP to maintain the ordinal number of the last (highest-numbered) DLSDU received at the DLCEP and reported to the remote DLCEP. Its value at DLCEP establishment is zero, and it increases monotonically with the reporting of each completely- received DLSDU to the DLS-user.

6.7.4.11 $V_C(H)$ highest-detected DLSDU sequence number

$V_C(H)$ is used by a DLE for a PEER or SUBSCRIBER DLCEP to maintain the ordinal number of the highest-numbered DLSDU received, or detected as missing, at the DLCEP. Its value at DLCEP establishment is zero, and it increases with the reception of the first-received segment of each sequentially-higher-numbered newly-received DLSDU, or with the detection of a sequentially-higher-numbered missing DLSDU.

6.7.4.12 $V_C(HS)$ highest-detected segment number of the highest-detected DLSDU sequence number

$V_C(HS)$ is used by a DLE for a PEER or SUBSCRIBER DLCEP to maintain the zero-based ordinal number of the highest-numbered segment of the $V_C(H)$ 'th DLSDU which has been received, or detected as missing, at the DLCEP. Its value at DLCEP establishment is zero:

- a) It is set to the zero-origin index of each newly-received segment, $N_R(ASN)$ (see 9.4.2.1C)6)), received in a DLPDU provided that
 - 1) the received DLSDU number is greater than the prior value of $V_C(H)$, or
 - 2) the received DLSDU number is the same as the prior value of $V_C(H)$, and the received segment number is greater than the prior value of $V_C(HS)$.
- b) It is set to zero when $V_C(H)$ is advanced upon detection of a missing DLSDU, and the number of segments in that DLSDU is unknown.

6.7.4.13 $V_{C,K}(MRS)$ missing received segments

$V_{C,K}(MRS)$ is used by a DLE for an ORDERED, DISORDERED or CLASSICAL PEER or SUBSCRIBER DLCEP to maintain the set of missing segments for the K 'th DLSDU, for $L < K \leq H$. Additions are made to this list, or one or more new lists created, upon receipt of any segment of a new DLSDU. Deletions are made as segments included on the list are received.

NOTE 1 When at least one segment of the K 'th DLSDU has been received, then the total number of segments in the DLSDU is known and the missing segments list is exact. Otherwise, it may be assumed that the missing DLSDU consists of only a single segment whose retransmission needs to be requested; upon receipt of any segment of the DLSDU, the list of segments still missing can be made exact.

NOTE 2 Subscribers of a common publisher DLCEP are permitted, but are not required, to eavesdrop on each other's transmissions to the publisher. Such eavesdropping may provide early detection of message loss. Thus one or more new lists may also be created upon receipt of a request for retransmission of a segment of the K 'th DLSDU, where $K > H$.

6.7.4.14 $V_{C,K}(RRS)$ retransmission-request required segments

$V_{C,K}(RRS)$ is used by a DLE for an ORDERED, DISORDERED or CLASSICAL, PEER or SUBSCRIBER DLCEP to maintain the set of missing segments for the K 'th DLSDU for which retransmission requests are required, for $L < K \leq H$. Additions are made to this list when a received DLPDU implies that one or more DLSDU segments or complete DLSDUs have been missed, and (for SUBSCRIBER DLCEPs) upon timeout of the timer $T_{C,K}(RRS)$ used by the DLE to monitor for a successful response to a previous retransmission request for one or more segments of the same DLSDU.

Deletions are made upon reception (including reception during transmission) of a DLPDU which either contains, or requests the retransmission of, a missing segment.

NOTE Such deletions upon reception of a request for retransmission for another subscriber on the same multi-peer DLC are optional (see note 2 in 6.7.4.13).

6.7.4.15 $T_{C,K}(RRS)$ retransmission request monitor

$T_{C,K}(RRS)$ is used by a DLE for a DISORDERED or CLASSICAL SUBSCRIBER DLCEP, and optionally for an ORDERED PEER or SUBSCRIBER DLCEP, to monitor for a successful response to a previous retransmission request for one or more segments of the K 'th DLSDU, where $L < K \leq H$. The timer is started when not running, but not restarted while running, when a retransmission request is sent for one of the segments of the incomplete DLSDU. It is cancelled when all segments of the DLSDU have been received. Upon its timeout, $V_{C,K}(RRS)$ is set equal to the then-current value of $V_{C,K}(MRS)$.

NOTE The timer may also be started, but not restarted, when a retransmission request from another subscriber of the same multi-peer DLC, for one of the segments of the incomplete DLSDU, is overheard (see Note 2 in 6.7.4.13).

6.7.4.16 $T_C(RAS)$ residual activity stimulus

$T_C(RAS)$ shall be used by a DLE at an ORDERED, DISORDERED or CLASSICAL, PUBLISHER or sending PEER DLCEP, when DLC establishment (see 10.2.1.1) requested residual activity on the DLC in this (sender-to-receiver) direction of data transfer, to ensure activity from that DLCEP during periods when all DLSDUs presented for transmission at that DLCEP since DLC establishment, or since the most recent DLC reset, have been transmitted

- a) at a sending DISORDERED or CLASSICAL PEER DLCEP, when all of those DLSDUs have been acknowledged (that is, $V_C(A) = V_C(M)$);
- b) at a PUBLISHER DLCEP, or sending ORDERED PEER DLCEP, when none of those DLSDUs need to be (wholly or partially) retransmitted.

This timer restarts at every transmission from the DLCEP. At timeout, it causes the current state of the DLCEP to be sent to the correspondent DLCEP(s).

6.7.4.17 $T_C(RAM)$ residual activity monitor

$T_C(RAM)$ is used by a DLE for an ORDERED, DISORDERED or CLASSICAL, SUBSCRIBER or receiving PEER DLCEP to detect inactivity on the DLC. It is only required when DLC establishment (see 10.2.1.1) requested residual activity on the DLC in this (receiver-from-sender) direction of data transfer. It runs continuously, and restarts upon reception of any DLPDU on the DLCEP. At timeout, it causes a DLCEP (and possibly DLC) reset, which may in turn lead to a DLCEP (and possibly DLC) disconnect.

6.7.4.18 $V_C(TNA)$ DL-time of last network access

$V_C(TNA)$ is used by the DLE to record the DL-time at which a DL-BUFFER-RECEIVED indication or DL-BUFFER-SENT indication was last generated at the associated DLCEP. It is used only when the DLCEP has been specified as a synchronizing DLCEP during the establishment of one or more other local DLCEPs.

6.7.4.19 V_B(TW) DL-time of last buffer write

V_B(TW) is used by the DLE to record the DL-time at which all of a DLSDU was last written to the associated buffer (by a DL-PUT request primitive, or by the completion of reception into the buffer at a DLCEP). It is used only when the DLCEP establishment specified a DL-timeliness class other than NONE.

6.7.4.20 V_B(TP) DL-time of production

V_B(TP) is used by the DLE to record the DL-time at which a DLS-user transferred the associated DLSDU to the DLE. This is the DL-time at which the sending DLS-user issued the DL-PUT request which placed the associated DLSDU in the sending DL-buffer. It is used only when the DLCEP establishment specified a DL-timeliness class other than NONE.

6.7.4.21 V_B(TS) Timeliness-status of buffer-writing

V_B(TS) is used by the DLE to record the timeliness-status associated with the DLSDU stored in a buffer. If the buffer is written as a result of a DL-PUT request, then the value of this variable shall be equal to the DLS-user data-timeliness. If the buffer is written as a result of reception of a DLPDU, then the value of this variable shall be updated as part of the reception procedure.

6.7.5 Variables and timers to support the Link Master class

The variables defined in 6.7.5 are required in all Link Master and Bridge DLEs.

6.7.5.1 V(DTA) delegation-address

V(DTA) is used by the LAS DLE to record the DL-address to which the token was last passed by a PASS TOKEN (PT), EXECUTE SEQUENCE (ES), or TRANSFER LAS (TL) DLPDU.

6.7.5.2 V(LL) local-link live-list

V(LL) is used by the LAS DLE to record the set of NODE DL-addresses (see 3.4.15 and 7.2.2.3) on the local link which appear to be in use by peer DLEs. It is used and updated by the LAS DLE during the LAS's address-probing and first-opportunity activities, as described in 12.2.

NOTE Some previously-existing national standards refer to such a set as a "live list", because it indicates the current set of interacting DLEs on the local link.

6.7.5.3 V(TCL) token-circulation list

V(TCL) is used by the LAS DLE to record the set of NODE DL-addresses (see 3.4.15 and 7.2.2.3) on the local link to which a PT DLPDU should be circulated to provide an opportunity for unscheduled transmissions, including requests to the LAS DLE for scheduling communications. This token circulation emulates the distributed token passing of prior national and international standards.

NOTE This variable is used to restrict the circulation of an unrestricted token to only the "master" (that is, self-sufficient) nodes of the local link. This functionality assists in migration from some previously-existing national standards.

V(TCL) is used and updated by the LAS DLE during the LAS's address-probing and first-opportunity activities, as described in 12.2. The DLEs represented in V(TCL) shall always be a subset of the DLEs represented in V(LL).

6.7.5.4 V(ENRL) expected-non-response list

V(ENRL) is used by the LAS DLE to record the set of NODE DL-addresses (see 3.4.15 and 7.2.2.3) on the local link of fractional-duty-cycle (FDC) DLEs which are expected to be non-responsive (that is, “asleep”). This is a subset of the set of FDC DLEs on the link, and changes dynamically as FDC DLEs inform the LAS DLE of either

- a) their attentiveness (“wakefulness”), or
- b) their intent to become non-responsive (to “go to sleep”).

Failure of a DLE listed in V(ENRL) to respond to a PT DLPDU addressed to the DLE, or an ES DLPDU addressed to a hierarchically-structured (see 7.2.2.1 format 1 or 7.2.2.2 format 1) DLSEP of the DLE,

- 1) shall not be treated as an error;
- 2) the transmission of the PT or ES DLPDU shall not be retried.

NOTE This last requirement is equivalent to using a value of zero (0) for V(MRC) when sending a PT DLPDU (see 8.15.3) to such a DLE.

6.7.5.5 V(MST) maximum-scheduled-traffic

V(MST) is used by the LAS DLE, during any dynamic schedule construction, to determine the maximum fraction of the theoretical interval V(TTRT) (see 6.7.5.11) which may be dedicated to explicitly scheduled DL-activity.

V(MST) is a configuration parameter of the local link which takes the form of a one-octet binary fraction. The range for this parameter is 0,00₁₆ to 0,BF₁₆; its default value is 0,40₁₆.

6.7.5.6 V(MSO) maximum-scheduling-overhead

V(MSO) specifies the maximum scheduling overhead permitted an LAS DLE by the existing link schedule. Its range is 0 to 3F₁₆; its default value is 3F₁₆, and its unit is the transmission duration of one octet. This overhead is included in the time allocated for each scheduled activity, and so is used only during schedule construction and determination of whether a DLE can serve as LAS for an existing schedule.

6.7.5.7 V(DMDT) default-minimum-token-delegation-time

V(DMDT) is used by the LAS DLE to determine the default minimum amount of local link capacity which the LAS shall allocate to a DLE in a single PT DLPDU sent to the DLE.

V(DMDT) is a configuration parameter of the local link. The range for this variable is 20₁₆ to 7FFF₁₆, its default value is 54₁₆ + V(PhLO), which permits the sending of one URGENT DLPDU, and its unit is the transmission duration of one octet.

6.7.5.8 V(DTHT) default-token-holding-time

V(DTHT) is used by the LAS DLE to specify the default initial amount of local link capacity which should be allocated to each DLE, in one cycle of “circulating the token”, by the LAS when the LAS sends one or more PT DLPDUs to the DLE.

V(DTHT) is a configuration parameter of the local link. The range for this variable is 114₁₆ to 65 000, its default value is 114₁₆ + V(PhLO), which permits the sending of one TIME-AVAILABLE DLPDU, and its unit is the transmission duration of one octet.

6.7.5.9 V(LTHT) link-maintenance-token-holding-time

V(LTHT) is used by the LAS DLE to specify the initial amount of local link capacity which should be allocated to LAS-related link maintenance activities in one cycle of “circulating the token”. These activities include

- probing a single unused NODE DL-address via a PN DLPDU for the appearance of a DLE not currently active on the link, and the consequent sending of a DT DLPDU containing a node-activation SPDU to any such DLE;
- measuring the round-trip communications delay with another DLE on the local link via an RQ DLPDU;
- conveying updated schedule information to other Link Master DLEs on the local link to provide for schedule continuation after transfer of the LAS role to such a DLE;
- coordinating a multi-link schedule with other LAS DLEs of the extended link.

V(LTHT) is a configuration parameter of the local link. The range for this variable is 124_{16} to 65 000, and its unit is the transmission duration of one octet. Its default value is

$$124_{16} + (3 \times V(\text{PhLO})) + (V(\text{IRRD}) \times V(\text{ST})), \quad \text{which permits}$$

the probing of one NODE DL-address and the sending of one TIME-AVAILABLE DT DLPDU.

6.7.5.10 V(MTHA) maximum-token-holding-time-array

V(MTHA) is used by the LAS to specify, separately for each DLE listed in V(TCL) (see 6.7.5.3), the initial amount of local link capacity which should be allocated to that DLE, in one cycle of “circulating the token”, by the LAS when the LAS sends one or more PT DLPDUs to that DLE. The range and units of this parameter are the same as V(DTHT) (see 6.7.5.8). Its default value is also V(DTHT).

6.7.5.11 V(TTRT) target-token-rotation-time

V(TTRT) is used by the LAS DLE to specify the desired upper bound on the time required for one cycle of “circulating the token” to all of the DLEs on the local link. One cycle of “circulating the token” is measured as the interval between successive occurrences of the LAS DLE sending a PT DLPDU, with a token-use subfield specifying RESTART (see 8.15.2), to the lowest-numbered NODE DL-address represented in V(LL).

V(TTRT) is a configuration parameter of the local link. The range for this variable is 1 to 60 000, its default value is 60 000, and its unit one ms.

6.7.5.12 V(ATRT) actual-token-rotation-time

V(ATRT) is used by the LAS DLE to determine the actual time used for each cycle of “circulating the token” to all of the DLEs on the local link, as measured by the interval between successive occurrences of the LAS DLE’s sending the PT DLPDU, with a token-use subfield specifying RESTART (see 8.15.2), to the lowest-numbered NODE DL-address represented in V(LL).

6.7.5.13 V(RTHA) remaining-token-holding-time-array

V(RTHA) is used by the LAS to specify, separately for each DLE listed in V(TCL) (see 6.7.5.3), the remaining amount of local link capacity which should be allocated to that DLE, in the current cycle of “circulating the token”, by the LAS when the LAS sends one or more PT DLPDUs to that DLE. The range and units of this array are the same as V(MTHA), and it is reinitialized from V(MTHA) at the beginning of each cycle of “circulating the token”.

6.7.5.14 V(NTHN) next-token-holding-node

V(NTHN) is used to record the NODE DL-address of the next DLE to which “the token should be circulated”, which is the next NODE DL-address to which a PT DLPDU will be sent. Its range is 10_{16} to FF_{16} (see 6.3.2.2), and it advances cyclically through the set of DLEs specified by V(TCL) (see 6.7.5.3).

6.7.5.15 V(FUN) first-unpolled-node

V(FUN) specifies the first NODE DL-address of a series of consecutive NODE DL-addresses which are to be omitted from the orderly probe of NODE DL-addresses for DLE's not specified by V(LL). Its range is 14_{16} to $F7_{16}$ (see 6.3.2.2).

6.7.5.16 V(NUN) number-of-consecutive-unpolled-nodes

V(NUN) specifies the number of consecutive NODE DL-addresses which are to be omitted from the orderly probe of NODE DL-addresses for DLE's not specified by V(LL). Its range is 00_{16} to $E4_{16}$, and its default value is 0.

6.7.5.17 P(TRD) token-recovery-delay

P(TRD) is used by the LAS DLE to record the DLE's token-recovery-delay (see 3.4.23), which is determined solely by the DLE itself. The default value for this variable is 14. Its range is $V(MRD)+3$ to 14, and its unit is one slot-time.

NOTE This unit is chosen so that all measurements of an inactive link are for multiples of one slot-time.

6.7.5.18 V(TDP) time-distribution-period

V(TDP) is used to determine the minimum frequency of time distribution on the local link. Its initial value shall be the minimum value required for the link's time-synchronization-class, V(TSC) as specified in 12.6.3. Its range is 5 ms to 55 s, and its unit is 1 ms.

6.7.5.19 V(MICD) maximum-inactivity-to-claim-LAS-delay

V(MICD) is used by the LAS DLE to record the link's maximum-inactivity-to-claim-LAS-delay (see 3.4.21), which is a configuration parameter of the local link. Its range is 1 to 4 095, and its unit is one-eighth of the transmission period of one octet.

6.7.5.20 V(LDDP) LAS-data-base-distribution-period

V(LDDP) is used by the LAS to determine the time between two successive distributions of the LAS' data-base by means of LAS-data-base-status SPDUs sent on the local link. The range of V(LDDP) is 100 ms to 55 s, and its unit is 1 ms. The default value of this variable is 5 s.

6.7.6 Variables and timers to support the Bridge class.

The variables and timers defined in 6.7.6 and in 12.4 are required in all Bridge DLEs.

6.7.6.1 V(ML) maximum-link

V(ML) may be used by a bridge DLE to limit the number of entries in the bridge's table of default forwarding information, which can have one entry for each link of the extended link. Its range is 1000_{16} to $FEFF_{16}$ (see 6.3.2.1), which is the maximum link-id defined for the extended network.

Additional variables and timers are defined in 12.4, mostly by reference to ISO/IEC 10038.

7 Type 1: General structure and encoding of PhIDUs and DLPDUs, and related elements of procedure

Within this clause, 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.

7.1 PhIDU structure and encoding

Each PhIDU consists of Ph-interface-control-information (PhICI) and in some cases one octet of Ph-interface-data (see 6.4). When the DLE transmits a DLPDU, it computes a frame check sequence for the DLPDU as specified in 5.1, concatenates the DLPDU and frame check sequence, and transmits the concatenated pair as a sequence of PhIDUs as follows:

- a) The DLE issues a single Ph-DATA request primitive with PhICI specifying START-OF-ACTIVITY, and awaits the consequent Ph-DATA confirm primitive.
- b) The DLE issues a sequence of Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the DLPDU as Ph-interface-data, from first to last octet of the DLPDU, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.
- c) The DLE issues a sequence of two Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the FCS as Ph-interface-data, from first to last octet of the FCS, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.
- d) The DLE issues a single Ph-DATA request primitive with PhICI specifying END-OF-DATA-AND-ACTIVITY, and awaits the consequent Ph-DATA confirm primitive.

The DLE forms a received DLPDU by concatenating the sequence of octets received as Ph-interface-control-information of consecutive Ph-DATA indications, computing a frame check sequence for those received octets as specified in 5.1, and checks the syndrome (residual) of the computed FCS for correctness as follows:

- e) The DLE receives a single Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY, and initializes its computation of an FCS for the received DLPDU.
- f) The DLE receives a sequence of Ph-DATA indication primitives with PhICI specifying DATA, each accompanied by one octet of the received DLPDU as Ph-interface-data, incrementally computes an FCS on the received octet, and concatenates all but the last two of those received octets to form the received DLPDU.
- g) The DLE receives a single Ph-DATA indication primitive with PhICI specifying either END-OF-DATA, END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY, and checks the syndrome of the computed FCS for correctness:
 - 1) If the PhICI specified END-OF-DATA or END-OF-DATA-AND-ACTIVITY, and the computed FCS syndrome was correct, then the DLE reports the reconstructed DLPDU and the two octets of received FCS as a correctly-received DLPDU suitable for further analysis.
 - 2) Otherwise the DLE increments its management statistics to reflect the erroneously-received DLPDU.

7.2 Common DLPDU structure, encoding and elements of procedure

Each DLPDU consists of a frame control field which specifies the type of DLPDU and conveys small size (fractional octet) parameters of the DLPDU; zero to three explicit address fields, each containing a DL-address, all of the same length; additional parameters of the DLPDU; and for most DLPDUs, a user data field conveying all or part of a DLSDU. To this is appended before transmission, and removed after reception, an FCS field (see 5.1) used to check the integrity of the received DLPDU.

7.2.1 Frame control (FC) field

The frame control (FC) field consists of one octet. It specifies the type of the DLPDU. For many types of DLPDU it also conveys a number of fractional-octet parameters, known as frame-control subfields, specific to the DLPDU type.

Some types of DLPDU require an immediate reply. Such DLPDUs may only be sent while holding a scheduler or delegated token.

When a token is delegated, then the priority specified in the token DLPDU is the minimum priority required of all DLPDUs sent during the following period of token use. DLPDUs of lower priority shall not be sent during that period.

7.2.1.1 Address size subfield

An address-size subfield is used to specify the number of octets in each address field of the DLPDU.

In DLPDUs which support multiple address sizes, this subfield occupies bit 3 of the frame control field. Its encoding is:

- 0: **SHORT** — the DLPDU's address fields are each two octets;
- 1: **LONG** — the DLPDU's address fields are each four octets, and their link designator subfields are non-zero if possible.

The DT DLPDU also has a special form with only implicit DL-addresses:

— **VERY-SHORT** — the DLPDU logically contains a single DL-address field, which is null (zero octets in length).

- i) If the single null DL-address field is a source address, then the source DL-address of the current DLPDU is implicitly the destination DL-address of the immediately prior CA, CD or ED DLPDU on the link (which must have been a DLCEP-address).
- ii) If the single null DL-address field is a destination address, then the destination DL-address of the current DLPDU is implicitly the source DL-address of the immediately prior CA, CD or ED DLPDU on the link (which itself may be an implicit DLSAP- or DLCEP-address).

NOTE This VERY SHORT form is indicated by other components of the DLPDU's frame control octet, and not by its address-size subfield.

7.2.1.2 Final token use subfield

A final-token-use designator is used to optimize the return of a delegated token at the end of the last transaction of an instance of delegated token usage. This subfield is present in most DLPDUs, including all DLPDUs which can be sent by a DLE which holds a delegated token or reply token. When present, this subfield occupies bit 2 of the frame control field. Its encoding and semantics depends on the type of DLPDU:

a) The encoding and semantics for CA, CD, CT, ED and RQ DLPDUs is:

- 0: **NOT-FINAL** — the delegated token is not returned to the LAS at the end of the current transaction;
- 1: **FINAL** — the delegated token is returned to the LAS at the end of the current transaction, after transmission of the DLPDU and its requested immediate reply DLPDU, whether detected or not, and no additional use of the token being returned is needed at this time.

- b) The encoding and semantics for DC, DT, EC, RC, RR, SR and TD DLPDUs is:
- 0: **NOT-FINAL** — the delegated token is not returned to the LAS at the end of the current transaction;
 - 1: **FINAL** — the delegated token is returned to the LAS at the end of the current transaction, after transmission of the DLPDU, and no additional use of the token being returned is needed at this time.
- c) The encoding and semantics for PT and ES DLPDUs is:
- 0: **RESTART** — this is the initial token delegation within the current cycle of “circulating the token” or of scheduled sequence execution, and so the indicated sequence and any repetitively scheduled transactions should be restarted;
 - 1: **CONTINUE** — this is a subsequent (that is, secondary) token delegation within the current cycle of “circulating the token” or of scheduled sequence execution, and so the previously initiated sequence of queued or scheduled transactions should be continued.

When a bridge receives for forwarding a DLPDU which contains a final-token-use subfield whose value is different than required by the bridge, then the bridge shall complement the received DLPDU’s final-token-use subfield before forwarding and shall make a compensating modification to the received DLPDU’s FCS field (see 5.1.3) to preserve the FCS integrity protection provided by the DLPDU’s originator.

NOTE Only CA, CD, DC, DT, EC, ED and RC DLPDUs (see 8.1 to 8.7) are forwarded through bridges.

7.2.1.3 Priority subfield

A Priority designator is used to specify the DLPDU’s or transaction’s priority, to limit the size of the user data field of the DLPDU, and to constrain the minimum (lowest) priority of any DLPDU sent as an immediate reply requested by, or token usage delegated by, this DLPDU. This subfield is present in all CA, CD, DT, ED, PT and RC DLPDUs which contain explicit DL-addresses. When present, this subfield occupies bits 1 and 0 of the frame control field. Its encoding is:

- 01: **URGENT** (high) priority;
- 10: **NORMAL** (medium) priority;
- 11: **TIME-AVAILABLE** (low) priority.

7.2.2 DL-address fields

The structure of DL-addresses is specified in 6.3, which also specifies standard pre-assigned addresses and address ranges.

7.2.2.1 LONG address field

A **LONG** address field usually consists of a fixed sequence of three parts as specified in 6.3. The parts are

- a) an explicit link designator component;
- b) an explicit node designator component;
- c) an explicit selector component.

Separately, the three parts reflect an hierarchical address structure. This hierarchy may be partially or totally flattened. These alternatives are shown in Figure 10.

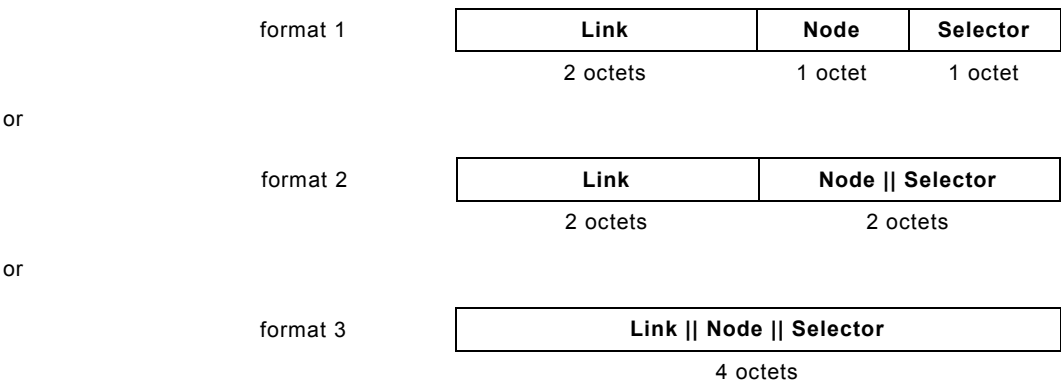


Figure 10 – DL-address alternative structures

When present, the Link field shall be delocalized on transmission and reception, as specified in 7.2.2.4. Similarly, when the Link field is present and specifies the local link, and when the Node field is present, then the Node field shall be delocalized on transmission.

7.2.2.2 SHORT address field

A **SHORT** address field usually consists of a fixed sequence of one implicit and two explicit parts, as specified in 6.3 and Figure 11. The parts are

- a) an implicit link designator component, specifying the local link, which is always present;
- b) an explicit node designator component;
- c) an explicit selector component.

Separately, the parts reflect an hierarchical address structure. This hierarchy may be partially flattened. The alternatives are shown in Figure 11.

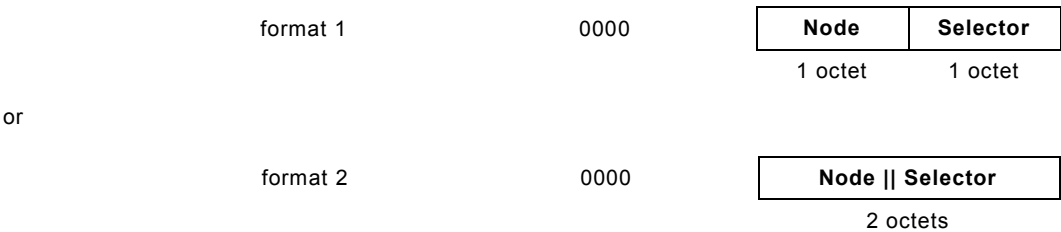


Figure 11 – SHORT DL-address field – alternative implicit structures

When present, the Node field shall always be delocalized on transmission, as specified in 7.2.2.4.

7.2.2.3 NODE DL-address field

A **NODE** DL-address is a one-octet version of a **SHORT** address field which designates some DLE's DL-support functions. It consists of a fixed sequence of two implicit and one explicit parts, as specified in 6.3 and Figure 12. The parts are

- a) an implicit link designator component, specifying the local link;
- b) an explicit node designator component, specifying the DLE;
- c) an implicit selector component of zero, specifying the DLE's DL-support functions.

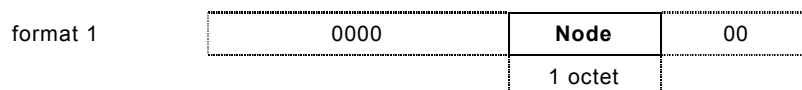


Figure 12 – NODE DL-address field – implicit structure

The Node field shall always be delocalized on transmission, as specified in 7.2.2.4.

7.2.2.4 Delocalization

The following transmission and reception processes are collectively referred to as delocalization (see 3.4.2), which is the DLE-internal process of converting synonymous DL-addresses to a canonical form for transmission, or for DL-address recognition during reception.

Two values of the Link field — the value zero and the value equal to V(TL) — are equivalent in designating the local link:

- a) On transmission, when the value of an explicit Link field would otherwise be zero, then the value of that Link field shall always be set to V(TL).

NOTE 1 The variable V(TL) can have the value zero, in which case this substitution effects no change.

NOTE 2 This equivalence makes it possible for a DLE to transmit on the local link without knowing the link's correct value for V(TL), or during periods when that value is in transition (in which case it is being administratively changed from zero to non-zero).

NOTE 3 This equivalence provision is an aid to DLS-users, and potentially in implementations of this DL-protocol; it permits references to the local link DL-address component to be represented uniformly by the value zero.

- b) On reception, with respect to the Link field, the value V(TL) shall be considered equivalent to the value zero, except that a DLPDU containing a DL-address with a Link field actually equal to zero shall not be forwarded by a bridge onto a different link.

Within a DLE, and when addressing the local link, two values of the Node field — the value zero and the value equal to V(TN) — are equivalent in designating the local DLE on the local link.

- c) On transmission, when the value of the Link field is equivalent to V(TL), and the value of a Node field would otherwise be zero, then the value of that Node field shall always be set to V(TN).

NOTE This equivalence provision is an aid to DLS-users, and potentially in implementations of this DL-protocol; it permits references to the local node DL-address component to be represented uniformly by the value zero.

7.2.3 Parameter field

Each DLPDU class may have a DLPDU-class-specific parameter field; these are all described in clause 9.

When it is desirable to distinguish between the values of a DLE variable, V(xx), or counter, C(xx), which is copied into a parameter field, and the current value of the same variable or counter, then the value within the parameter field is referred to as N(xx), because it no longer tracks changes in the value of the source variable or counter. The need for this separation of nomenclature is particularly apparent in the case of counters such as C(NT), which never stop counting, and in the various time-related DLPDUs (TD, RQ and RR), which may contain multiple fields based on various samplings of C(NT).

7.2.4 User data field

RC, CA, DT and ED DLPDUs, associated with the connectionless and connection-oriented data transfer services, contain a user data field which is used to convey either a partial or complete DLSU from one DLS-user to another. The size of this user data field is constrained to be no larger than that permitted by the priority of the conveying DLPDU:

- 01: **URGENT** (high) priority: ≤ 64 octets;
- 10: **NORMAL** (medium) priority: ≤ 128 octets;
- 11: **TIME-AVAILABLE** (low) priority: ≤ 256 octets.

PR DLPDUs contain a user data field which is used to convey a probe-response SPDU (see 11.3.2.1) from the token-holding DLE to the current LAS DLE; its size is constrained to be no larger than permitted at URGENT priority — 64 octets.

TL DLPDUs contain a user data field which is used to convey a LAS-data-base-status SPDU (see 11.3.2.3) from the LAS DLE to the addressed DLE; its size is constrained to be no larger than permitted at URGENT priority — 64 octets.

DC and EC DLPDUs contain a user data field. Their implicit priority is TIME-AVAILABLE; however, the size of their data fields is constrained to be no larger than permitted at NORMAL priority — 128 octets.

7.2.5 Elements of procedure for minimum-inter-DLPDU delay

The DLE which holds the dominant token shall start transmission only after providing a delay of at least minimum-inter-DLPDU delay, $V(MID)$, octet durations, where the delay is measured as specified in 6.7.1.12.

7.2.6 Elements of procedure for dropping of token by dominant token holder

If the DLE which holds the dominant token has just completed either

- a) the transmission of a DLPDU, as indicated by receipt of a Ph-DATA confirm primitive corresponding to the most-recent Ph-DATA request primitive, which shall have specified END-OF-DATA-AND-ACTIVITY, or
- b) the reception of a DLPDU, as indicated by receipt of a Ph-Data indication primitive specifying either END-OF-DATA-AND-ACTIVITY or END-OF-DATA or END-OF-ACTIVITY

but has not yet started its next transmission, either

- c) because it is waiting for the required interval of minimum-inter-DLPDU delay, or
- d) because it is not ready to transmit due to delay in preparing the next transmission;

then if that DLE is able to receive during this interval and if a Ph-Data indication (see 6.4.4) reporting DATA is received, then the DLE shall drop the token.

7.2.7 Common elements of procedure for monitoring link activity

A number of DLPDU types (CA, CD, ED, CT, RQ, PT, ES, TL) sent by a token holder request an immediate response. Common procedures apply for those DLEs which

- a) need to monitor for that immediate response;
- b) need to determine
 - when to stop monitoring for that response;
 - whether or not that response occurred.

7.2.7.1 Monitoring for an immediate response by the initiating DLE

After sending a CA, CD, ED, CT or RQ, DLPDU which requests an immediate response,

- a) where the sending DLE is not also the responding DLE, then the sending DLE shall
 - 1) monitor the local link for a period of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations, waiting for a reply;
 - 2) then take appropriate action based on the result of that monitoring;
- b) where the sending DLE is also the responding DLE, it shall take action as if the monitoring had occurred and had successfully detected the requested immediate response DLPDU.

The monitoring procedure is:

- 1) If a Ph-Data indication (see 6.4.4) reporting DATA is received, then the DLE shall
 - i) stop further monitoring;
 - ii) then wait for receipt of the Ph-Data indication reporting END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY;
- 2) If 1) does not apply, and the monitoring period expires, then
 - i) if the bus is not active at that moment (that is, the last-received Ph-Data indication reported END-OF-ACTIVITY), then the DLE shall stop further monitoring.
 - ii) if i) does not apply, implying that the link is active at that moment (that is, the last-received Ph-Data indication reported START-OF-ACTIVITY), then
 - α) For implementations that were not demonstrable on or before 31 December 1995, the DLE shall monitor the local link for a period of one additional slot-time, $V(ST)$ octet-durations, waiting for a Ph-Data indication:
 - A) If a Ph-Data indication reporting DATA is received, then the DLE shall proceed as in 1).
 - B) If a Ph-Data indication reporting END-OF-ACTIVITY is received, and A) does not apply, then the DLE shall stop further monitoring.
 - C) If neither A) nor B) applies
 - D) and the monitoring period expires before a Ph-Data indication is received, then the DLE shall wait for receipt of the Ph-Data indication reporting END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY.
 - β) Implementations that were demonstrable on or before 31 December 1995 alternatively may just proceed as in α).C) when neither α).A) nor α).B) applies.

NOTE After this document achieved initial ACDV status, implementors were encouraged to develop chips to assist in evaluating this complex protocol. As a consequence, it was found desirable to improve the noise rejection characteristics of the token passing process, and the text 2)ii) α), which would require changes in those existing chips, was the result. β) grandfathers the noise rejection approach of the first ACDV for those early implementations, and only for those early implementations.

At the end of monitoring, the sending DLE shall act based on the result of that monitoring:

- 3) If 2)i) or 2)ii) α)B) applied, then the sending DLE shall
 - report the failure to detect a DLPDU to local DL-management;
 - if the final-token-use subfield of the originating CA, CD, ED, CT or RQ DLPDU specified NOT-FINAL, then the DLE shall start the next transmission within immediate-response-recovery-delay plus one slot-times, $(V(IRRDR) + 1) \times V(ST)$ octet-durations, of the end of transmission of that CA, CD, ED, CT or RQ DLPDU.

- 4) If 1) or 2)ii)α)A) or 2)ii)α)C) applied and link activity did not result in a DLPDU, then the sending DLE shall
 - report the failure to detect a valid reply to local DL-management;
 - if the final-token-use subfield of the originating CA, CD, ED, CT or RQ DLPDU specified NOT-FINAL, then the DLE shall start the next transmission within maximum-reply-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the beginning of the current period of link non-activity;
- 5) If
 - the link activity resulted in a DLPDU;
 - if the received DLPDU was not a permissible reply DLPDU,

then any held token shall be dropped and local DL-management shall be notified of the event.

NOTE These DL-management reports may take the form of incrementing a DL-management error counter.

- 6) If
 - the link activity resulted in a DLPDU;
 - the received DLPDU was a permissible reply DLPDU;
 - the final-token-use subfield of the originating CA, CD, ED, CT or RQ DLPDU specified NOT-FINAL,

then the DLE shall start the next transmission within maximum-reply-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the beginning of the current period of link non-activity.

7.2.7.2 Monitoring for an immediate response by the LAS DLE

If the final-token-use subfield of the received CA, CD, ED or RQ DLPDU has the value FINAL, then

- a) if the LAS DLE is not the addressed immediate responder DLE, then the LAS DLE shall apply the monitoring procedures of 7.2.7.1 1) and 7.2.7.1 2);
- b) if the LAS DLE is also the responding DLE, then it shall take action as if the monitoring had occurred and had successfully detected the immediate response DLPDU.

After completing the monitoring a), or after sending the requested immediate response b), the LAS DLE shall

- 1) assume that the current use of the delegated token has terminated and that the scheduler token is again dominant on the local link; and
- 2) treat that termination as if the token had been returned by an RT (see 8.17) DLPDU;
- 3) resume active operation as the LAS.

7.2.7.3 Monitoring for an immediate response as a subscriber to a DLC

A DLE which

- receives a CA, CD or ED DLPDU requesting an immediate response;
- is an intended recipient of that requested immediate response DLPDU;
- is not itself the CA, CD or ED DLPDU's addressed responder,

shall initiate a timer with a duration of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations:

- a) If a Ph-Data indication (see 6.4.4) reporting DATA is received, then the DLE shall stop further monitoring and wait for receipt of the Ph-Data indication reporting END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY and proceed as in e).
- b) If the monitoring period expires, and the bus is not active at the end of this monitoring period, then the DLE shall stop further monitoring and proceed as in d).
- c) If the monitoring period expires, and the bus is active at the end of this monitoring period, then the DLE shall wait for receipt of the Ph-Data indication reporting END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY, and then proceed as in d) or e) as appropriate.
- d) If the monitoring did not result in a DLPDU as in b) or possibly in c), then the DLE shall invalidate V(RA).
- e) If the link activity resulted in a DLPDU as in a) or possibly in c), and if the received DLPDU was a DT DLPDU whose source DL-address is implicitly or explicitly the destination DLCEP-address specified by the CA, CD or ED DLPDU, then the recorded address V(RA) shall be assumed to be the implied source address of that CA, CD or ED DLPDU. Otherwise the recorded address V(RA) shall be invalidated.

7.2.8 Two-way alternate (half-duplex) operation

There is no requirement for a sending DLE to receive PH-DATA indications while it is transmitting a DLPDU; but when such indications are not receivable (that is, when the DLE is operating in a two-way alternate or half-duplex mode), then a sending DLE shall treat each transmitted DLPDU as if it had been received concurrently by that DLE.

8 Type 1: DLPDU-specific structure, encoding and elements of procedure

NOTE 1 The elements of procedure in this clause correspond generally to the lowest DLL sublevel, as specified in 6.1.1.

This clause defines the structure, contents and encoding of each DLPDU type and format, and specifies elements of procedure for that DLPDU type and format. (Table 10 summarizes their structure.)

Within each subclause, the DLPDU's structure, contents and encoding are described first. Then those aspects relating to the sending and receiving DLS-users and their DLEs are addressed, followed by those aspects, if any, that are unique to the store-and-forwarding functions of bridges (relay DLEs).

Throughout this clause, whenever a conditional action is specified and the specified enabling condition does not occur, then the corresponding action also does not occur. The net effect is that events which do not meet any of the enabling conditions specified in a service procedure will have no consequential actions with respect to that specific service procedure.

A DLE which is not ONLINE to the local link, but which is attempting to change its DL-state to ONLINE, may send a DLPDU only as a response to a PN DLPDU addressed to the DLE's DL-support functions, as specified in 8.13.4, 8.14.3 and 12.2.1. During this interval the DL-protocol specifies the response of the DLE to received PN, PT and DT DLPDUs addressed to the DLE's DL-support functions. The DLE's behavior after it changes its state to ONLINE shall not be dependent in any way on the receipt before it went ONLINE of other than the above-specified (PN, PT, DT) DLPDUs.

NOTE 2 Receipt of a DLPDU without observable side-effects is a purely local matter, and is thus outside the scope of standardization.

A DLE which is ONLINE to the local link may send a DLPDU only when it holds the dominant token on the local link. The set of DLPDUs which may be sent while holding the dominant token is dependent on the class of the dominant token, as summarized in Table 11:

- a) If the dominant token is a scheduler token, then the permitted DLPDU classes are CA, CD, DC, DT, EC, ED, ES, PN, PT, RC, RQ, TD, TL, WK and IDLE.
- b) If the dominant token is a delegated token, then the permitted DLPDU classes are CA, CD, CT, DC, DT, EC, ED, RC, RI, RQ, RT, WK and IDLE.
- c) If the dominant token is a reply token, then the permitted DLPDU classes are
 - 1) DT and SR when replying to a CA, CD or ED DLPDU;
 - 2) TD when replying to a CT DLPDU;
 - 3) PR when replying to a PN DLPDU;
 - 4) RR when replying to an RQ DLPDU.
- d) Tokens are created only by the following DLPDUs:
 - 1) A scheduler token is created only by a CL DLPDU;
 - 2) A delegated token is created only by PT and ES DLPDUs;
 - 3) A reply token is created only by CA, CD, CT, ED, PN, RQ and TL DLPDUs.

Table 10 – Summary structure of DLPDUs

DLPDU class	Frame control	DL-addresses			Parameters	User data
		Destination	Source	2nd source		
EC 1	1111 LF00	[HL.]N.S	[HL.]N.S	[HL.]N.S	EC-p	o-DLSDU
EC 2	1110 LF00		[HL.]N.S	[HL.]N.S	EC-p	o-DLSDU
DC 1	0111 LF00	[HL.]N.S	[HL.]N.S		DC-p	o-DLSDU
DC 2	0110 LF00		[HL.]N.S		DC-p	o-DLSDU
RC 1	0111 LFPP	[HL.]N.S	[HL.]N.S		RC-p	o-DLSDU
RC 2	0110 LFPP		[HL.]N.S		RC-p	o-DLSDU
CA 1	1110 LFPP	[HL.]N.S	[HL.]N.S		SD-p	o-pDLSDU
CA 2	1010 LFPP	[HL.]N.S	—		SD-p	o-pDLSDU
CD 1	1111 LFPP	[HL.]N.S	[HL.]N.S		o-SD-p	—
CD 2	1011 LFPP	[HL.]N.S	—		o-SD-p	—
ED 1	1100 LFPP	[HL.]N.S	[HL.]N.S		SD-p	pDLSDU
ED 2	1000 LFPP	[HL.]N.S	—		SD-p	pDLSDU
DT 1	1101 LFPP	[HL.]N.S	[HL.]N.S		SD-p	o-pDLSDU
DT 2	1001 LFPP	[HL.]N.S	—		SD-p	o-pDLSDU
DT 3	0101 LFPP		[HL.]N.S		SD-p	o-pDLSDU
DT 4	1001 0F00	[PSA]	—		SD-p	o-pDLSDU
DT 5	0101 0F00		[PDA]		SD-p	o-pDLSDU
SR	0001 0F11	[PSA]	N		o-SR-p	—
CT	0001 0F00	—	—		—	—
TD	0001 0F01	—	N		TD-p	—
RQ	1100 0F00	N.0	N.0		RQ-p	—
RR	1101 0F00	N.0	N.0		RR-p	—
PN	0010 0110	N	—		PN-p	—
PR	0010 0111	—	—		—	SPDU
PT	0011 0FPP	N	—		DD-p	—
ES	1000 LF00	[HL.]N.S	—		DD-p	—
RT	0011 0100	—	[DTH]		—	—
RI	0010 0000	—	[DTH]		DD-p	—
CL	0000 0001	—	N		—	—
TL	0000 0110	N	—		—	SPDU
WK	0000 0000	N	—		—	—
IDLE	0001 0F10	—	—		—	o-DLSDU

Key

L indicates the length of the associated DL-addresses (0 = SHORT, 1 = LONG)

F indicates final use of a token, or that a sequence should be finished rather than restarted

PP specifies the priority of the DLPDU and any passed token

shading indicates a logically non-existent field

— indicates a logically existent field whose contents are required to be null

[HL.]N.S is a four-octet LONG DL-address (HLNS) when L = 1
or a two-octet SHORT DL-address (NS) with HL = 00 implied when L = 0

N is a one-octet NODE DL-address

N.0 is the two-octet SHORT DL-address form of a one-octet NODE DL-address

[PDA] is the implied DL-address equal to the explicit destination DL-address of the immediately prior DLPDU on the link, which must have been a CA, CD or ED DLPDU. This is a logically existent field whose actual contents are required to be null

[PSA] is the implied DL-address equal to the implied or explicit source DL-address of the immediately prior DLPDU on the link. This is a logically existent field whose actual contents are required to be null

[DTH] is the implied DL-address equal to the explicit destination DL-address of the most recent PT or ES DLPDU on the link — the DLPDU which delegated the token returned by the RT or RI DLPDU. This is a logically existent field whose actual contents are required to be null

o- indicates optional field contents

xx-p indicates xx-class DLPDU parameters

DLSDU is a DL Service Data Unit

pDLSDU is a complete or partial DLSDU

SPDU is a Support Protocol Data Unit

Table 11 – DLPDU restrictions based on dominant token

DLPDU class	Type of token created	Can be sent while using Scheduler token	Can be sent while using Delegated token	Can be sent in Reply to
EC	none	Y	Y	—
DC	none	Y	Y	—
RC	none	Y	Y	—
CA	Reply	Y	Y	—
CD	Reply	Y	Y	—
ED	Reply	Y	Y	—
DT	none	Y	Y	CA, CD, ED
SR	none	N	N	CA, CD, ED, TL
CT	Reply	N	Y	—
TD	none	Y	N	CT
RQ	Reply	Y	Y	—
RR	none	N	N	RQ
PN	Reply	Y	N	—
PR	none	N	N	PN
PT	Delegated	Y	N	—
ES	Delegated	Y	N	—
RT	none	N	Y	—
RI	none	N	Y	—
CL	none	N	N	—
TL	Scheduler	Y	N	—
WK	none	Y	Y	—
IDLE	none	Y	Y	—

8.1 Establish Connection (EC) DLPDU

An ESTABLISH CONNECTION (EC) DLPDU is used to establish a peer DLC between two DLS-users, or a multi-peer DLC between a publishing DLS-user and subscribing DLS-users.

8.1.1 Structure of the EC DLPDU

Table 12 – Structure of EC DLPDUs

format	Frame control	Destination address	Source address	2nd source address	Parameters	User data
1L	1111 1F00	HL.N.S	HL.N.S	HL.N.S	EC-p	o-DLSDU
1S	1111 0F00	N.S	N.S	N.S	EC-p	o-DLSDU
2L	1110 1F00		HL.N.S	HL.N.S	EC-p	o-DLSDU
2S	1110 0F00		N.S	N.S	EC-p	o-DLSDU

8.1.1.1 Frame control field

The frame control field shall specify

- the DLPDU's function;

- b) the DLPDU's implicit priority, which is TIME-AVAILABLE;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.1.1.2 Address field

The address field shall consist of either

- a) an explicit destination DL-address and two explicit source DL-addresses, in that order, for formats 1L and 1S, or
- b) only two explicit source DL-addresses, for formats 2L and 2S.

For formats 1L and 2L, all addresses shall be LONG; for formats 1S and 2S, all addresses shall be SHORT.

8.1.1.3 Parameters field

The establish-connection parameters (EC-parameters) field specifies the proposed or selected QoS attributes of the DLC, including the version of the DL protocol in use. This field shall be structured and encoded as described in 9.1.

8.1.1.4 User data field

The user data field shall consist of a single optional DLSDU whose maximum size is limited to 128 octets.

8.1.2 Content of the EC DLPDU

The frame control field shall be encoded as specified in Table 12.

For formats 1L and 1S,

- a) the first DL-address shall be a DL(SAP)-address or a DLCEP-address;
- b) the second DL-address shall be a DLCEP-address or a DLSAP-address;
- c) the third DL-address shall be a DLSAP-address of the DLSAP associated with that second DL-address.

For formats 2L and 2S, the first DL-address shall be a publisher DLCEP-address, and the second DL-address shall be a DLSAP-address of the DLSAP associated with that first DL-address.

The EC-parameters shall specify the proposed or selected QoS attributes of the DLC, including the version of the DL-protocol in use. The contents of this field shall be as described in 9.1.

8.1.3 Sending the EC DLPDU

An EC DLPDU may be sent on the link when the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, C(RD), permits completion of the EC DLPDU's transmission prior to expiration of the token.

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) before transmission.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU is needed at that time, then the DLE may set the final-token-use subfield of the EC DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

8.1.4 Receiving the EC DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received EC DLPDU shall be treated as follows by the receiving DLE:

8.1.4.1 Actions required of all DLEs

If the first DL-address specified by the DLPDU designates an active DL(SAP)-address or an active DLCEP of the DLE, then the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.3) for further processing.

8.1.4.2 Additional actions required of a Link-Master class DLE

None.

8.1.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.1.4.2 also apply to a Bridge class DLE.
- b) If the first DL-address specified in the DLPDU is one which the bridge should forward, and the bridge was able to buffer the DLPDU without error, then the received DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).
- c) The bridge shall attempt to update its routing table entry for the source DLCEP-address and source DLSAP-address specified in the DLPDU to reflect the bridge port from which the DLPDU was received.
- d) If the DLPDU's Establish-Connection parameters indicate that the DLPDU's addressee(s) will be the subscriber(s) of a multi-peer DLC, then the bridge shall attempt to update its routing table entry for the source DLCEP-address specified in the DLPDU by adding the bridge port(s) to which the DLPDU is being forwarded to the set of sink DLCEP ports associated with that source DLCEP-address.

NOTE This last procedure is only meaningful for formats 1L and 1S of the EC DLPDU.

8.1.4.4 Additional actions required of the current LAS DLE

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
- b) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.2 Disconnect Connection (DC) DLPDU

A DISCONNECT CONNECTION (DC) DLPDU is used to disconnect an existing or proposed DLC.

8.2.1 Structure of the DC DLPDU

Table 13 – Structure of DC DLPDUs

format	Frame control	Destination address	Source address	Parameters	User data
1L	0111 1F00	HL.N.S	HL.N.S	DC-p	o-DLSDU
1S	0111 0F00	N.S	N.S	DC-p	o-DLSDU
2L	0110 1F00		HL.N.S	DC-p	o-DLSDU
2S	0110 0F00		N.S	DC-p	o-DLSDU

8.2.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is TIME-AVAILABLE;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.2.1.2 Address field

The address field shall consist of either

- a) an explicit destination DL-address and an explicit source DL-address, in that order, for formats 1L and 1S, or
- b) only an explicit source DL-address, for format 2L and 2S.

For formats 1L and 2L, all addresses shall be LONG; for formats 1S and 2S, all addresses shall be SHORT.

8.2.1.3 Parameters field

The disconnect-connection parameters (DC-parameters) field shall specify the version of the DL protocol in use, the desired action and the reason for that action. This field shall be structured and encoded as described in 9.2.

8.2.1.4 User data field

The user data field shall consist of a single optional DLSDU whose maximum size is limited to 128 octets.

8.2.2 Content of the DC DLPDU

The frame control field shall be encoded as specified in Table 13.

For formats 1L and 1S, either

- a) both DL-addresses shall be peer DLCEP-addresses, or
- b) the first DL-address shall be a DL(SAP)-address, and the second DL-address shall be a DLCEP-address, or
- c) the first DL-address shall be a DLCEP-address, and the second DL-address shall be a DLSAP-address.

For formats 2L and 2S, the sole DL-address shall be a publisher DLCEP-address.

The DC-parameters shall specify the version of the DL-protocol in use, the desired action and reason, and other information. The contents of this field shall be encoded as described in 9.2.

8.2.3 Sending the DC DLPDU

A DC DLPDU may be sent on the link when the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, $C(RD)$, permits completion of the DLPDU's transmission prior to expiration of the token.

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) before transmission.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU is needed at that time, then the DLE may set the final-token-use subfield of the DC DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

8.2.4 Receiving the DC DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received DC DLPDU shall be treated as follows by the receiving DLE:

8.2.4.1 Actions required of all DLEs

If the first DL-address specified by the DLPDU designates an active DLSAP-address or an active DLCEP of the receiving DLE, then the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.1.7) for further processing.

8.2.4.2 Additional actions required of a Link-Master class DLE

None.

8.2.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.2.4.2 also apply to a Bridge class DLE.
- b) If the first DL-address specified in the DLPDU is one which the bridge should forward, and the bridge was able to buffer the DLPDU without error, then the received DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).
- c) If the DLPDU's Disconnect-Connection parameters indicate that a reply is not requested, then the bridge may attempt to remove the routing table entries for any explicit DLCEP addresses specified in the DLPDU.

8.2.4.4 Additional actions required of the current LAS DLE

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
- b) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.3 Reset Connection (RC) DLPDU

A RESET CONNECTION (RC) DLPDU is used to reset an existing or proposed DLC.

8.3.1 Structure of the RC DLPDU

Table 14 – Structure of RC DLPDUs

format	Frame control	Destination address	Source address	Parameters	User data
1L	0111 1FPP	HL.N.S	HL.N.S	RC-p	o-DLSDU
1S	0111 0FPP	N.S	N.S	RC-p	o-DLSDU
2L	0110 1FPP		HL.N.S	RC-p	o-DLSDU
2S	0110 0FPP		N.S	RC-p	o-DLSDU

8.3.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's priority;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.3.1.2 Address field

The address field shall consist of either

- a) an explicit destination DL-address and an explicit source DL-address, in that order, for formats 1L and 1S, or
- b) only an explicit source DL-address, for format 2L and 2S.

For formats 1L and 2L, all addresses shall be LONG; for formats 1S and 2S, all addresses shall be SHORT.

8.3.1.3 Parameters field

The reset-connection parameters (RC-parameters) field shall specify the version of the DL protocol in use, the desired action and reason, and other information. This field shall be structured and encoded as described in 9.3.

8.3.1.4 User data field

The user data field shall consist of a single optional DLSDU whose maximum size is limited to the maximum DLS-user data size permitted for a DLPDU of the priority specified in 8.3.1.1b).

8.3.2 Content of the RC DLPDU

The frame control field shall be encoded as specified in Table 14.

For formats 1L and 1S, both DL-addresses shall be peer DLCEP-addresses. For formats 2L and 2S, the sole DL-address shall be a publisher DLCEP-address.

The RC-parameters shall specify the version of the DL protocol in use, the desired action and reason, and other information. The contents of this field shall be encoded as described in 9.3.

8.3.3 Sending the RC DLPDU

An RC DLPDU may be sent on the link when the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, $C(RD)$, permits completion of the DLPDU's transmission prior to expiration of the token.

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) before transmission.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU is needed at that time, then the DLE may set the final-token-use subfield of the RC DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

8.3.4 Receiving the RC DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received RC DLPDU shall be treated as follows by the receiving DLE:

8.3.4.1 Actions required of all DLEs

If the first DL-address specified by the DLPDU designates an active DLCEP of the receiving DLE, then the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing.

8.3.4.2 Additional actions required of a Link-Master class DLE

None.

8.3.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.3.4.2 also apply to a Bridge class DLE.
- b) If the receiving DLE is a bridge, and the first DL-address specified in the DLPDU is one which the bridge should forward, and the bridge was able to buffer the DLPDU without error, then the received DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).

8.3.4.4 Additional actions required of the current LAS DLE

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;

assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.4 Compel Acknowledgement (CA) DLPDU

A COMPEL ACKNOWLEDGMENT (CA) DLPDU is used

- a) to transfer (or retransfer) a limited amount of transparent user data from one requesting DLS-user to another DLS-user;
- b) to request that the transfer be acknowledged as soon as possible,

without requiring that the transaction (request and acknowledgment) occur within the context of a DLC. It is also used for similar purposes within the context of a DLC, and to assist in the synchronization of DLCEPs and of their DLS-users.

A CA DLPDU creates and passes a reply token to the addressed receiving DLE, which upon reception becomes the dominant token on its link. A CA DLPDU requires an immediate reply of either a DATA (DT) DLPDU with no DLS-user data or a STATUS RESPONSE (SR) DLPDU. If no DLPDU is received in reply, then the transaction is repeated a maximum of V(MRC) times.

8.4.1 Structure of the CA DLPDUs

Table 15 – Structure of CA DLPDUs

format	Frame control	Destination address	Source address	Parameters	User data
1L	1110 1FPP	HL.N.S	HL.N.S	SD-p	o-pDLSDU
1S	1110 0FPP	N.S	N.S	SD-p	o-pDLSDU
2L	1010 1FPP	HL.N.S	—	SD-p	o-pDLSDU
2S	1010 0FPP	N.S	—	SD-p	o-pDLSDU

8.4.1.1 Frame control field

The frame control field shall specify

- the DLPDU's function, including that an immediate reply is being requested;
- the transaction's and DLPDU's priority;
- the length, number and type of DLPDU addresses;
- whether or not transmission of the current DLPDU, and either its expected immediate reply or the appropriate link-idle timeout, terminates use of a delegated token.

NOTE This field necessarily has the value NOT-FINAL when an immediate retry of the current transaction is possible. Only a transaction which is guaranteed to not need an immediate retry may have the value FINAL specified in its associated DLPDU.

8.4.1.2 Address field

The address field shall consist of either

- an explicit destination DL-address and an explicit source DL-address, in that order, for formats 1L and 1S, or
- only an explicit destination DL-address, for formats 2L and 2S.

For formats 1L and 2L, all addresses shall be LONG; for formats 1S and 2S, all addresses shall be SHORT.

8.4.1.3 Parameters field

The status-data-parameters (SD-parameters) field specifies information appropriate to the associated destination DL-address:

- If that DL-address is a DLSAP-address, then the SD-parameters specify
 - a transaction-id used by the originating DLE to correlate a delayed returned reply with the originating request;
 - a DLSDU-priority used to convey the actual priority of the accompanying DLSDU to the responding DLE, or that there is no accompanying DLSDU.

This field shall be structured and encoded as described in 9.4.1.

- If that DL-address is a DLCEP-address, then the SD-parameters specify state information for the addressed DLCEP. This field shall be structured and encoded as described in 9.4.2.

NOTE The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.

8.4.1.4 User data field

The user data field size and content are limited by the associated destination DL-address:

- a) If that DL-address is a DLSAP-address, then the user data field shall consist of DLS-user data whose maximum size is limited to the smaller of the maximum DLS-user data sizes permitted for a DLPDU of the priorities specified in 8.4.1.1b) and 8.4.1.3a)2), and shall not be null.
- b) If that DL-address is a DLCEP-address, then the user data field shall consist of DLS-user data whose maximum size is limited to the smaller of
 - 1) the maximum DLS-user data size permitted for a DLPDU of the priority specified in 8.4.1.1b);
 - 2) the maximum DLSDU size negotiated on the DLC for data transmission to that DLCEP, and may be null.

8.4.2 Content of the CA DLPDU

The frame control field shall be encoded as specified in Table 15.

Either the DL-addresses shall be

- a) two explicit DLSAP-addresses, or
- b) one explicit DLCEP-address, and a second explicit or implicit DLCEP-address, or
- c) one explicit DLCEP-address, followed by one explicit or implicit DLSAP-address.

8.4.2.1 Content of the CA DLPDU when specifying a destination DLSAP-address

When the first DL-address is a DLSAP-address as in 8.4.2a), then

- a) if the DLPDU format is format 1L or 1S, then
 - 1) this DLPDU is being used to implement the unitdata transfer service with remote DLE confirmation;
 - 2) the DL(SAP)-role for the destination DLSAP-address shall be BASIC;
 - 3) the second address shall be present, shall be a DLSAP-address, and the DL(SAP)-role for that DLSAP-address shall be BASIC;
 - 4) the SD-parameters field shall specify a DLSDU-priority and a transaction-id used by the originating DLE to correlate a delayed returned reply with the originating request, where
 - i) the contents of this field shall be as described in 9.4;
 - ii) the DLSDU-priority shall be the priority of the accompanying user data and shall be the same as the DLPDU-priority specified in 8.4.1.1b);
 - 5) the user data shall be a single DLSDU whose size is limited to the maximum size for the priority specified in 8.4.1.4a), and shall not be null;
- b) no other DLPDU format may be used.

8.4.2.2 Content of the CA DLPDU when specifying a destination DLCEP-address

When the first address is a DLCEP-address, as in 8.4.2b) and 8.4.2c), then

- a) this DLPDU can convey a single or partial DLSDU
 - from one peer DLCEP to its corresponding peer DLCEP, or
 - from a subscriber DLCEP to its corresponding publisher DLCEP,

shall request state information from the addressed DLCEP, and shall not permit DLS-user data to be included in the reply DLPDU;
- b) the second address, if present,
 - shall be the peer DLCEP-address of the same DLC as the destination peer DLCEP-address, or
 - shall be a subscriber DLCEP's calling-DLSAP-address of the same DLC as the destination publisher DLCEP-address;
- c) the SD-parameters field shall specify state information for the addressed DLCEP, and the contents of this field shall be as described in 9.4;

NOTE 1 The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.

- d) the user data shall specify those octets of a DLSDU consistent with the negotiated DLSDU size and the segmentation information specified in the accompanying SD-parameters, and may be null.

NOTE 2 A CA DLPDU with null user data can be used by a DLE with a subscriber DLCEP to solicit current DLC state information from the corresponding publisher DLCEP.

NOTE 3 Formats 1L and 1S are used for peer-to-peer (see 8.4.2b)) and subscriber-to-publisher (see 8.4.2c)) communications when the DLPDU-authentication attribute is SOURCE or MAXIMAL. Formats 2L and 2S are used for peer-to-peer and subscriber-to-publisher communications when the DLPDU-authentication attribute is ORDINARY.

It is a protocol error if the above conditions are not met.

8.4.3 Sending the CA DLPDU

A CA DLPDU may be selected for transmission on the link when

- a) the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link;
- b) the remaining allocated duration of token usage, C(RD), permits completion of V(MRC)+1 implied transactions prior to expiration of the token, where each transaction consists of sending the CA DLPDU that requires an immediate reply, and awaiting a worst-case SR DLPDU or worst-case permitted DT reply DLPDU not containing DLS-user data;
- c) if the CA DLPDU will be addressed to a DLSAP-address, then the outstanding-transaction-array, V(OTA) (see 6.7.1.15), searched circularly from the last-transaction-index, V(LTI) (see 6.7.1.16), has an unassigned entry whose index is not V(LTI).

Once selected, if the CA DLPDU will be addressed to a DLSAP-address, then

- V(LTI) shall be set to the index of that unassigned entry in V(OTA);
- that entry in V(OTA) shall be assigned to the selected CA DLPDU and shall record information which permits an expected response DT DLPDU to be correlated with the specific invocation of the unitdata-transfer service which gave rise to the CA DLPDU.

Once selected, transmission of the CA DLPDU shall be retried until either

- 1) a permissible immediate reply DLPDU is received, or
- 2) an impermissible DLPDU is received when an immediate reply was expected, or

- 3) the original transmission and the permitted maximum number of transmission retries, $V(MRC)$ (see 6.7.1.5), have all failed to elicit one of the permissible reply DLPDUs.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU and awaiting its immediate reply is needed at that time, then the DLE may set the final-token-use subfield of the CA DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

NOTE The FINAL transaction is necessarily the $(V(MRC)+1)$ 'th in a series; otherwise, if a reply is not received, then another CA DLPDU would have to be sent before the current cycle of token use was completed, making the current transaction NOT-FINAL.

Each explicit DL-address in the CA DLPDU shall be delocalized (see 7.2.2) before transmission.

After sending a CA DLPDU, the sending DLE shall monitor the local link for a reply as specified in 7.2.7.1. The permissible reply DLPDU is either

- a DT DLPDU whose destination DL-address is implicitly or explicitly the originating DLSAP-address or DLCEP specified by the CA DLPDU, or
- a DT DLPDU without a destination DL-address, or
- an SR DLPDU.

If $V(LTI)$ was assigned to the transaction, as a result of the search described in c), and if a permissible reply DLPDU was not received, then the $V(LTI)$ 'th entry shall be deassigned.

8.4.4 Receiving the CA DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received CA DLPDU shall be treated as follows by the receiving DLE:

8.4.4.1 Actions required of all DLEs

NOTE The next alternative attempts to detect the reception of a duplicated CA DLPDU resulting from an immediate retry by the current token-holding DLE, which itself was probably caused by an error detected during receipt of the earlier reply DT DLPDU. In such a case the response DT DLPDU is required to be identical to the first in those fields and subfields which convey DLS-user data and related information.

a) If

- 1) the destination DL-address specified by the DLPDU designates an active DLSAP-address of the receiving DLE, or an active DLCEP-address of a DLC for which the receiving DLE is peer or publisher;
- 2) the immediately-prior DLPDU was transmitted as an immediate-response to a received CA DLPDU whose destination DL-address was the same DLSAP-address or DLCEP-address;
- 3) if a source DL-address is present in the just-received CA DLPDU, then it was also present and identical in that prior received CA DLPDU;
- 4) starting with the SD-parameters field, the first three octets of the just-received CA DLPDU, or the remainder of the DLPDU if fewer than three octets, are identical to the corresponding octets of that prior received CA DLPDU;
- 5) a period of link inactivity of $(\text{immediate-response-recovery-delay} + 1) \times \text{slot-times}$, $(V(IRR) + 1) \times V(ST)$ octet-durations, has not occurred since receipt of that prior received CA DLPDU,

then the receiving DLE

- i) shall discard the received DLPDU and not forward it to the DLE's upper-level functions for further processing;

ii) shall retransmit the prior-transmitted immediate-reply DT DLPDU, within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the CA DLPDU, unchanged from the prior transmission except for those subfields of

— the final-token-use subfield of the frame control octet which conveys information to the listening LAS;

— those subfields of any present DLCEP SD-parameters which acknowledge receipt of, or request retransmission of, received DLSDUs.

NOTE This requirement is meant to ensure that the DL-priority, content and identity of any DLS-user data conveyed in the immediate reply DT DLPDU is identical to that in the immediately-prior DT DLPDU sent from this same DL-address, including the originator's transaction-id if one is present in the received and transmitted SD-parameters.

- b) If a) does not apply, and the destination DL-address specified by the DLPDU designates a DLSAP-address of the receiving DLE, then the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.3) for further processing.

NOTE Group DL-addresses are not covered here; such CA DLPDUs are erroneous and are unrecognized upon receipt.

The receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the CA DLPDU. The reply DLPDU shall be a DT DLPDU with any explicit destination DL-address in the DT DLPDU having the same value as the source DL-address of the received CA DLPDU, and length and format as follows:

- DT (see 8.7.1 format 2L) in response to CA (see 8.4.1 format 1L), or
- DT (see 8.7.1 format 4) in response to CA (see 8.4.1 format 1S),

and shall include SD-parameters specifying

- the originator's transaction-id as received in the stimulating CA DLPDU, as specified in 8.4.2.1a)4);
- reception status for the DLSDU conveyed by that CA DLPDU.

The reply DT DLPDU shall not contain any DLS-user data.

- c) If a) does not apply, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE
- is a peer or publisher, and the DLL priority of the DLCEP is not equal to the priority specified in the received DLPDU, or
 - is a peer, and the length and number of DL-addresses is not as expected or the DLPDU specifies an explicit source address which is not equal to the remote peer DLCEP's DLCEP-address,

then

1) the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;

2) the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the CA DLPDU. The reply DLPDU shall be a DT DLPDU in the format negotiated for the DLC for the selected direction of transmission, and shall contain SD-parameters appropriate to the sending DLCEP, but shall not contain DLS-user data.

- d) If neither a) nor c) applies, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE is a peer or publisher, then

- 1) the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the CA DLPDU. The reply DLPDU shall be a DT DLPDU in the format negotiated for the DLC for the selected direction of transmission, shall contain SD-parameters appropriate to the sending DLCEP, but shall not contain DLS-user data;
 - 3) the requesting CA DLPDU may contain information about the state of the requesting DLCEP. The reply DT DLPDU is permitted, but is not required, to reflect that state information in its reply; immediate processing of that state information before sending the immediate reply shall be permitted but shall not be required.
- e) If a) does not apply, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE is a subscriber, then
- 1) the non-DLS-user data portion of the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall record the destination DL-address from the received CA DLPDU in V(RA) for subsequent association with the expected immediate reply DT DLPDU, which should be the next DLPDU received;
 - 3) the receiving DLE shall monitor the local link for a reply and then act based on the result of that monitoring, all as specified in 7.2.7.3 .

8.4.4.2 Additional actions required of a Link-Master class DLE

None.

8.4.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.1.4.2 also apply to a Bridge class DLE.
- b) If all of the following conditions hold
 - A) the destination DL-address specified in the DLPDU is one which the bridge should forward but to which the bridge DLE itself would not otherwise generate an immediate reply DLPDU;
 - B) the immediately-prior DLPDU was transmitted as an immediate-response to a received CA DLPDU with the same destination DL-address;
 - C) if a source DL-address is present in the just-received CA DLPDU, then it was also present and identical in that prior received CA DLPDU;
 - D) the basic-DLC-parameters portion of the SD-parameters of the just-received CA DLPDU is identical to that of that prior received CA DLPDU;
 - E) a period of link inactivity of immediate-response-recovery-delay plus one slot-time, $(V(IRRDR)+1) \times V(ST)$ octet-durations, has not occurred since receipt of that prior received CA DLPDU,

then the bridge

- 1) shall discard the received DLPDU and not forward it to the bridge's other functions for further processing;
- 2) shall initiate retransmission of the prior-transmitted immediate-reply SR DLPDU, unchanged from the prior transmission.

NOTE This requirement is meant to ensure that any status conveyed in the immediate reply SR DLPDU is identical to that in the immediately prior SR DLPDU.

Otherwise,

i) If the destination DL-address specified in the DLPDU is one which the bridge should forward but which the bridge DLE itself would not otherwise receive, then the bridge shall form and send an SR DLPDU

— within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the CA DLPDU;

— with status indicating whether or not the bridge was able to buffer the received DLPDU.

ii) If the destination DL-address specified in the DLPDU is one which the bridge should forward, and the bridge was able to receive and buffer the DLPDU without error, then the received DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).

iii) If the DLPDU contains an explicit source DL-address, then the bridge shall attempt to update its routing table entry for the source DL-address specified in the DLPDU to reflect the bridge port from which the DLPDU was received.

c) It is a protocol error for a bridge DLE which will forward the received CA DLPDU to not send an SR reply DLPDU when a reply is required.

NOTE At most one bridge DLE on the local link should be forwarding the received CA DLPDU.

8.4.4.4 Additional actions required of the current LAS DLE

The LAS DLE shall act as specified in 7.2.7.2.

8.5 Compel Data (CD) DLPDU

A COMPEL DATA (CD) DLPDU is used to request the transfer (or retransfer) of a limited amount of transparent user data from another DLS-user to the requesting DLS-user without requiring that the transaction (request and acknowledgment) occur within the context of a DLC. It is also used for similar purposes within the context of a DLC, and to assist in the synchronization of DLCEPs and of their DLS-users. When the CD DLPDU is addressed to a publishing DLCEP, then the DLS-user data whose transfer is requested will be distributed to all of the subscribers of the DLC.

A CD DLPDU creates and passes a reply token to the addressed receiving DLE, which upon reception becomes the dominant token on the link. A CD DLPDU requires an immediate reply of either a DATA (DT) DLPDU or a STATUS RESPONSE (SR) DLPDU. If no DLPDU is received in reply, then the transaction is repeated a maximum of $V(\text{MRC})$ times.

8.5.1 Structure of the CD DLPDUs

Table 16 – Structure of CD DLPDUs

format	Frame control	Destination address	Source address	Parameters
1L	1111 1FPP	HL.N.S	HL.N.S	o-SD-p
1S	1111 0FPP	N.S	N.S	o-SD-p
2L	1011 1FPP	HL.N.S	—	o-SD-p
2S	1011 0FPP	N.S	—	o-SD-p

8.5.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function, including that an immediate reply is being requested;
- b) the transaction's priority and DLPDU's implied priority;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU, and either its expected immediate reply or the appropriate link-idle timeout, terminates use of a delegated token.

NOTE This field necessarily has the value NOT-FINAL when an immediate retry of the current transaction is possible. Only a transaction which is guaranteed not to need an immediate retry may have the value FINAL specified in its associated DLPDU.

8.5.1.2 Address field

The address field shall consist of either

- a) an explicit destination DL-address and an explicit source DL-address, in that order, for formats 1L and 1S, or
- b) only an explicit destination DL-address, for formats 2L and 2S.

For formats 1L and 2L, all addresses shall be LONG; for formats 1S and 2S, all addresses shall be SHORT.

8.5.1.3 Parameters field

The status-data-parameters (SD-parameters) field specifies information appropriate to the associated destination DL-address:

- a) If that DL-address is a DLSAP-address bound in a responder DL(SAP)-role, then the SD-parameters specify a transaction-id used by the originating DLE to correlate a delayed returned reply with the originating request. This field shall be structured and encoded as described in 9.4.1.
- b) If that DL-address is a DLCEP-address, then the SD-parameters specify state information for the addressed DLCEP. This field may be null. When non-null, this field shall be structured and encoded as described in 9.4.2.

NOTE The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.

When the CD DLPDU is sent by the LAS DLE, which occurs while the dominant token is a scheduler token, then this field shall be null.

8.5.1.4 User data field

The user data field shall be null.

8.5.2 Content of the CD DLPDU

The frame control field shall be encoded as specified in Table 16.

Either the DL-addresses shall be

- a) two explicit DLSAP-addresses, or
- b) one explicit DLCEP-address, and a second explicit or implicit or not-present DLCEP-address, or
- c) one explicit DLCEP-address, followed by one explicit or implicit DLSAP-address.

8.5.2.1 Content of the CD DLPDU when specifying a destination DLSAP-address

When the first DL-address is a DLSAP-address as in 8.5.2a), then

- a) if the DLPDU format is format 1L or 1S, then
 - 1) this DLPDU is being used to implement the unitdata exchange service;
 - 2) the DL(SAP)-role for the destination DLSAP-address shall be CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER;
 - 3) the second address shall be present, shall be a DLSAP-address, and the DL(SAP)-role for that DLSAP-address shall be INITIATOR;
 - 4) the SD-parameters field shall specify a transaction-id used by the originating DLE to correlate a delayed returned reply with the originating request, where the contents of this field shall be as described in 9.4;
 - 5) the user data shall be null.
- b) no other DLPDU format may be used.

8.5.2.2 Content of the CD DLPDU when specifying a destination DLCEP-address

When the first address is a DLCEP-address, as in 8.5.2b) and 8.5.2c), then

- a) this DLPDU shall request state information from the addressed DLCEP, and shall request that DLS-user data be included in the reply DLPDU;
- b) the second address, if present,
 - shall be the peer DLCEP-address of the same DLC as the destination peer DLCEP-address, or
 - shall be a subscriber DLCEP's calling-DLSAP-address of the same DLC as the destination publisher DLCEP-address;
- c) the SD-parameters field, if present, shall specify state information for the addressed DLCEP, and the contents of this field shall be as described in 9.4;

NOTE 1 The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.

- d) the user data shall be null.

NOTE 2 If the user data field is not null, then an ED DLPDU (see 8.6) should be used instead.

NOTE 3 Formats 1L and 1S are used for peer-to-peer (see 8.5.2b)) and subscriber-to-publisher (see 8.5.2c)) communications when the DLPDU-authentication attribute is SOURCE or MAXIMAL. Formats 2L and 2S are used for peer-to-peer and subscriber-to-publisher communications when the DLPDU-authentication attribute is ORDINARY.

8.5.3 Sending the CD DLPDU

A CD DLPDU may be selected for transmission on the link when

- a) the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link;
- b) the remaining allocated duration of token usage, C(RD), permits completion of V(MRC)+1 implied transactions prior to expiration of the token, where each transaction consists of sending the CD DLPDU that requires an immediate reply, and awaiting a worst-case SR DLPDU or worst-case permitted DT reply DLPDU containing DLS-user data;
- c) if the CD DLPDU will be addressed to a DLSAP-address, then the outstanding-transaction-array, V(OTA) (see 6.7.1.15), searched circularly from the last-transaction-index, V(LTI) (see 6.7.1.16), has an unassigned entry whose index is not V(LTI).

Once selected, if the CD DLPDU will be addressed to a DLSAP-address, then

- V(LTI) shall be set to the index of that unassigned entry in V(OTA);

- that entry in V(OTA) shall be assigned to the selected CD DLPDU and shall record information which permits an expected response DT DLPDU to be correlated with the specific invocation of the unitdata-exchange service which gave rise to the CD DLPDU.

Once selected, transmission of the CD DLPDU shall be retried until either

- 1) a permissible immediate reply DLPDU is received, or
- 2) an impermissible DLPDU is received when an immediate reply was expected, or
- 3) the original transmission and the permitted maximum number of transmission retries, V(MRC) (see 6.7.1.5), have all failed to elicit one of the permissible reply DLPDUs.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU and awaiting its immediate reply is needed at that time, then the DLE may set the final-token-use subfield of the CD DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

NOTE The FINAL transaction is necessarily the (V(MRC)+1)'th in a series; otherwise, if a reply is not received, then another CD DLPDU would have to be sent before the current cycle of token use was completed, making the current transaction NOT-FINAL.

Each explicit DL-address in the CD DLPDU shall be delocalized (see 7.2.2) before transmission.

After sending a CD DLPDU, the sending DLE shall monitor the local link for a reply as specified in 7.2.7.1. The permissible reply DLPDU is either

- a DT DLPDU whose destination DL-address is implicitly or explicitly the originating DLSAP-address or DLCEP specified by the CD DLPDU, or
- a DT DLPDU without a destination DL-address, or
- an SR DLPDU.

If V(LTI) was assigned to the transaction, as a result of the search described in c), and if a permissible reply DLPDU was not received, then the V(LTI)'th entry shall be deassigned.

8.5.4 Receiving the CD DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received CD DLPDU shall be treated as follows by the receiving DLE:

8.5.4.1 Actions required of all DLEs

NOTE The next alternative attempts to detect the reception of a duplicated CD DLPDU resulting from an immediate retry by the current token-holding DLE, which itself was probably caused by an error detected during receipt of the earlier reply DT DLPDU. In such a case the response DT DLPDU is required to be identical to the first in those fields and subfields which convey DLS-user data and related information.

a) If

- 1) the destination DL-address specified by the DLPDU designates an active DLSAP-address of the receiving DLE, or an active DLCEP-address of a DLC for which the receiving DLE is peer or publisher;
- 2) the immediately-prior DLPDU was transmitted as an immediate-response to a received CD DLPDU whose destination DL-address was the same DLSAP-address or DLCEP-address;
- 3) if a source DL-address is present in the just-received CD DLPDU, then it was also present and identical in that prior received CD DLPDU;
- 4) starting with the SD-parameters field, the first three octets of the just-received CD DLPDU, or the remainder of the DLPDU if fewer than three octets, are identical to the corresponding octets of that prior received CD DLPDU;

- 5) a period of link inactivity of (immediate-response-recovery-delay + 1) slot-times, $(V(\text{IRRD}) + 1) \times V(\text{ST})$ octet-durations, has not occurred since receipt of that prior received CD DLPDU,

then the receiving DLE

i) shall discard the received DLPDU and not forward it to the DLE's upper-level functions for further processing;

ii) shall retransmit the prior-transmitted immediate-reply DT DLPDU, within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the CD DLPDU, unchanged from the prior transmission except for those subfields of

— the final-token-use subfield of the frame control octet which conveys information to the listening LAS;

— those subfields of any present DLCEP SD-parameters which acknowledge receipt of, or request retransmission of, received DLSDUs.

NOTE This requirement is meant to ensure that the DL-priority, content and identity of any DLS-user data conveyed in the immediate reply DT DLPDU is identical to that in the immediately-prior DT DLPDU sent from this same DL-address, including the originator's transaction-id if one is present in the received and transmitted SD-parameters.

- b) If a) does not apply, and the destination DL-address specified by the DLPDU designates a DLSAP-address of the receiving DLE, then the processing of the received DLPDU shall be based upon the DL(SAP)-role specified for that DLSAP-address.

NOTE Group DL-addresses are not covered here; such CD DLPDUs are erroneous and are unrecognized upon receipt.

The receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the CD DLPDU:

- 1) If the DLE can prepare it in time, then the reply DLPDU shall be a DT DLPDU with any explicit destination DL-address in the DT DLPDU having the same value as the source DL-address of the received CD DLPDU, and length and format as follows:

— DT (see 8.7.1 format 2L) in response to CD (see 8.5.1 format 1L);

— DT (see 8.7.1 format 2S) in response to CD (see 8.5.1 format 1S) if the DT DLPDU contains DLS-user data, or

— DT (see 8.7.1 format 4) in response to CD (see 8.5.1 format 1S) if the DT DLPDU does not contain DLS-user data,

and shall include SD-parameters specifying

— the originator's transaction-id as received in the stimulating CD DLPDU, as specified in 8.5.2.1a)4);

— status for the DLSDU requested by that CD DLPDU.

The reply DT DLPDU may include a DLSDU which was already buffered at that responding DLE at the time of CD reception.

NOTE This restriction prohibits the reply DLPDU from reflecting any higher-layer (than DLL) processing of the received DLSDU. This restriction is necessary to enable migration from prior national standards.

If the DL(SAP)-role of the destination DLSAP-address specified

- i) BASIC or INITIATOR, then the DLE shall reject the received CD DLPDU based on this DL(SAP)-role, and an appropriate error status without an accompanying DLSDU shall be included in the reply DT DLPDU;
 - ii) UNCONSTRAINED RESPONDER, then the SD-parameter status shall indicate the highest DLL priority sending buffer with a non-null DLSDU which was available at the addressed DLSAP at the time of reception of the CD DLPDU;
 - A) if that DLL priority is greater than or equal to the priority specified in 8.5.1.1b), then that DLSDU shall be included in the reply DT DLPDU and its buffer set to empty if so configured, and the DLL priority of that reply DT DLPDU shall be the DLL priority of the conveyed DLSDU; or
 - B) if that DLL priority is less than the priority specified in 8.5.1.1b), then no DLSDU shall be included in the reply DT DLPDU, and the DLL priority of that reply DT DLPDU shall be the DLL priority specified in 8.5.1.1b); or
 - C) if there is no such sending buffer with a non-null DLSDU, then the reply DT DLPDU shall specify an appropriate error status, the DLL priority of that reply DT DLPDU shall be the DLL priority specified in 8.5.1.1b), and that reply DT DLPDU shall not contain DLS-user data.
 - iii) CONSTRAINED RESPONDER, then
 - A) if the source DLSAP-address of the CD DLPDU is equal to the remote DLSAP address which was specified in the prior DL-BIND request primitive for the receiving DLSAP-address (or its DL-management equivalent), both after delocalization, then the procedure specified in ii) shall be followed;
 - B) otherwise, the received DLSDU shall be discarded, and an appropriate error status without an accompanying DLSDU shall be included in the reply DT DLPDU.
- 2) If the DLE cannot prepare the required reply DT DLPDU in time, then the DLE shall send a DT DLPDU with any explicit destination DL-address in the DT DLPDU having the same value as the source DL-address of the received CD DLPDU, and length and format as follows:
- DT (see 8.7.1 format 2L) in response to CD (see 8.5.1 format 1L), or
 - DT (see 8.7.1 format 4) in response to CD (see 8.5.1 format 1S),
- and shall include SD-parameters
- with a transaction-id identical to the transaction-id from the received CD DLPDU, as specified in 8.5.2.1a)4);
 - with a status (DR — “delayed reply”) indicating that the DLE requires additional time to prepare the required response;
 - with a null user data field,

and the DLE

- shall prepare that DT DLPDU as specified in 1) as soon as possible;
 - shall include an explicit destination address in the reply DT DLPDU;
 - shall append that reply DT DLPDU to the DLE's Q(US) to be transmitted at the first opportunity.
- c) If a) does not apply, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE
- is a peer or publisher, and the DLL priority of the DLCEP is not equal to the priority specified in the received DLPDU, or
 - is a peer, and the length and number of DL-addresses is not as expected or the DLPDU specifies an explicit source address which is not equal to the remote peer DLCEP's DLCEP-address,
- then
- 1) the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the CD DLPDU. The reply DLPDU shall be a DT DLPDU in the format negotiated for the DLC for the selected direction of transmission, and shall contain SD-parameters appropriate to the sending DLCEP, but shall not contain DLS-user data.
- d) If neither a) nor c) applies, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE is a peer or publisher, then
- 1) the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the CD DLPDU. The reply DLPDU shall be a DT DLPDU in the format negotiated for the DLC for the selected direction of transmission, shall contain SD-parameters appropriate to the sending DLCEP, and shall contain DLS-user data if any was available and waiting for transmission or retransmission from the DLCEP;
- NOTE Migration from prior national standards requires that this data always be included when available.
- 3) The requesting CD DLPDU may contain information about the state of the requesting DLCEP. The reply DT DLPDU is permitted, but is not required, to reflect that state information in its reply; immediate processing of that state information before sending the immediate reply shall be permitted but shall not be required.
- e) If a) does not apply, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE is a subscriber, then
- 1) the non-DLS-user data portion of the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall record the destination DL-address from the received CD DLPDU in V(RA) for subsequent association with the expected immediate reply DT DLPDU, which should be the next DLPDU received;
 - 3) the receiving DLE shall monitor the local link for a reply and then act based on the result of that monitoring, all as specified in 7.2.7.3 .

8.5.4.2 Additional actions required of a Link-Master class DLE

None.

8.5.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.5.4.2 also apply to a Bridge class DLE.
- b) If all of the following conditions hold
 - A) the destination DL-address specified in the DLPDU is one which the bridge should forward but to which the bridge DLE itself would not otherwise generate an immediate reply DLPDU;
 - B) the immediately prior DLPDU was transmitted as an immediate-response to a received CD DLPDU with the same destination DL-address;
 - C) if a source DL-address is present in the just-received CD DLPDU, then it was also present and identical in that prior received CD DLPDU;
 - D) the basic-DLC-parameters portion of the SD-parameters of the just-received CD DLPDU is identical to that of that prior received CD DLPDU;
 - E) a period of link inactivity of immediate-response-recovery-delay plus one slot-times, $(V(IRR\!D)+1) \times V(ST)$ octet-durations, has not occurred since receipt of that prior received CD DLPDU,

then the bridge

- 1) shall discard the received DLPDU and not forward it to the bridge's other functions for further processing;
- 2) shall initiate retransmission of the prior-transmitted immediate-reply SR DLPDU, unchanged from the prior transmission.

NOTE This requirement is meant to ensure that any status conveyed in the immediate reply SR DLPDU is identical to that in the immediately prior SR DLPDU.

Otherwise

- i) If the destination DL-address specified in the DLPDU is one which the bridge should forward but which the bridge DLE itself would not otherwise receive, then the bridge shall form and send an SR DLPDU,
 - within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the CD DLPDU;
 - with status indicating whether or not the bridge was able to buffer the received DLPDU.
- ii) If the destination DL-address specified in the DLPDU is one which the bridge should forward, and the bridge was able to receive and buffer the DLPDU without error, then the received DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).
- iii) If the DLPDU contains an explicit source DL-address, then the bridge shall attempt to update its routing table entry for the source DL-address specified in the DLPDU to reflect the bridge port from which the DLPDU was received.
- c) It is a protocol error for a bridge DLE which will forward the received CD DLPDU to not send an SR reply DLPDU when a reply is required.

NOTE At most one bridge DLE on the local link should be forwarding the received CD DLPDU.

8.5.4.4 Additional actions required of the current LAS DLE

The LAS DLE shall act as specified in 7.2.7.2.

8.6 Exchange Data (ED) DLPDU

An EXCHANGE DATA (ED) DLPDU is used

- a) to transfer (or retransfer) a limited amount of transparent user data from the requesting DLS-user to another DLS-user;
- b) to request the transfer (or retransfer) of a limited amount of transparent user data from that other DLS-user to the requesting DLS-user,

without requiring that the transaction (request and acknowledgment) occur within the context of a DLC. It is also used for similar purposes within the context of a DLC, and to assist in the synchronization of DLCEPs and of their DLS-users. When the ED DLPDU is addressed to a publishing DLCEP, then the DLS-user data whose transfer is requested in b) will be distributed to all of the subscribers of the DLC.

An ED DLPDU creates and passes a reply token to the addressed receiving DLE, which upon reception becomes the dominant token on the link. An ED DLPDU requires an immediate reply of either a DATA (DT) DLPDU or a STATUS RESPONSE (SR) DLPDU. If no DLPDU is received in reply, then the transaction is repeated a maximum of V(MRC) times.

8.6.1 Structure of the ED DLPDUs

Table 17 – Structure of ED DLPDUs

format	Frame control	Destination address	Source address	Parameters	User data
1L	1100 1FPP	HL.N.S	HL.N.S	SD-p	pDLSDU
1S	1100 0FPP	N.S	N.S	SD-p	pDLSDU
2L	1000 1FPP	HL.N.S	—	SD-p	pDLSDU
2S	1000 0FPP	N.S	—	SD-p	pDLSDU

8.6.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function, including that an immediate reply is being requested;
- b) the transaction's priority, and the DLPDU's implied priority unless explicitly overridden by 8.6.1.3a)2);
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU, and either its expected immediate reply or the appropriate link-idle timeout, terminates use of a delegated token.

NOTE This field necessarily has the value NOT-FINAL when an immediate retry of the current transaction is possible. Only a transaction which is guaranteed to not need an immediate retry may have the value FINAL specified in its associated DLPDU.

8.6.1.2 Address field

The address field shall consist of either

- a) an explicit destination DL-address and an explicit source DL-address, in that order, for formats 1L and 1S, or
- b) only an explicit destination DL-address, for formats 2L and 2S.

For formats 1L and 2L, all addresses shall be LONG; for formats 1S and 2S, all addresses shall be SHORT.

8.6.1.3 Parameters field

The status-data-parameters (SD-parameters) field specifies information appropriate to the associated destination DL-address:

- a) If that DL-address is a DLSAP-address bound in a responder DL(SAP)-role, then the SD-parameters specify
 - 1) a transaction-id used by the originating DLE to correlate a delayed returned reply with the originating request;
 - 2) a DLSDU-priority used to convey the actual priority of the accompanying DLSDU to the responding DLE.

This field shall be structured and encoded as described in 9.4.1.

- b) If that DL-address is a DLCEP-address, then the SD-parameters specify state information for the addressed DLCEP. This field shall be structured and encoded as described in 9.4.2.

NOTE The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.

8.6.1.4 User data field

The user data field size and content are limited by the associated destination DL-address:

- a) If that DL-address is a DLSAP-address, then the user data field shall consist of DLS-user data whose maximum size is limited to the smaller of the maximum DLS-user data sizes permitted for a DLPDU of the priority specified in 8.6.1.1b) and 8.6.1.3a)2), and shall not be null.
- b) If that DL-address is a DLCEP-address, then the negotiated DLC attributes for the intended direction of transmission shall permit DLS-user data to be carried in an ED DLPDU (see 9.1c)6) and 9.1d)6)), and the user data field shall consist of DLS-user data whose maximum size is limited to the smaller of
 - 1) the maximum DLS-user data size permitted for a DLPDU of the priority specified in 8.6.1.1b);
 - 2) the maximum DLSDU size negotiated on the DLC for data transmission to that DLCEP, and shall not be null.

8.6.2 Content of the ED DLPDU

The frame control field shall be encoded as specified in Table 17.

Either the DL-addresses shall be

- a) two explicit DLSAP-addresses, or
- b) one explicit DLCEP-address, and a second explicit or implicit DLCEP-address, or
- c) one explicit DLCEP-address, followed by one explicit or implicit DLSAP-address.

8.6.2.1 Content of the ED DLPDU when specifying a destination DLSAP-address

When the first DL-address is a DLSAP-address as in 8.6.2a), then

- a) if the DLPDU format is format 1L or 1S, then
 - 1) this DLPDU is being used to implement the unitdata exchange service;

- 2) the DL(SAP)-role for the destination DLSAP-address shall be CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER;
- 3) the second address shall be present, shall be a DLSAP-address, and the DL(SAP)-role for that DLSAP-address shall be INITIATOR;
- 4) the SD-parameters field shall specify a DLSDU-priority and a transaction-id used by the originating DLE to correlate a delayed returned reply with the originating request, where the contents of this field shall be as described in 9.4, and the DLSDU-priority shall be the priority of the accompanying user data and shall be the same as specified in 8.6.1.3a)2) and shall be as least as high as the DLPDU priority specified in 8.6.1.1b);
- 5) the user data shall be a single DLSDU whose size is limited to the maximum size for the priority specified in 8.6.1.4a), and shall not be null.

b) no other DLPDU format may be used.

8.6.2.2 Content of the ED DLPDU when specifying a destination DLCEP-address

When the first address is a DLCEP-address, as in 8.6.2b) and 8.6.2c), then

- a) this DLPDU can convey a single or partial DLSDU
 - from one peer DLCEP to its corresponding peer DLCEP, or
 - from a subscriber DLCEP to its corresponding publisher DLCEP,

shall request state information from the addressed DLCEP, and shall request that DLS-user data be included in the reply DLPDU, and
 - b) the second address, if present,
 - shall be the peer DLCEP-address of the same DLC as the destination peer DLCEP-address, or
 - shall be a subscriber DLCEP's calling-DLSAP-address of the same DLC as the destination publisher DLCEP-address;
 - c) the SD-parameters field shall specify state information for the addressed DLCEP, and the contents of this field shall be as described in 9.4;
- NOTE 1 The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.
- d) the user data shall specify those octets of a DLSDU consistent with the negotiated DLSDU size and the segmentation information specified in the accompanying SD-parameters, and shall not be null.

NOTE 2 If the user data field is null, then a CD DLPDU (see 8.5) should be used instead.

NOTE 3 Formats 1L and 1S are used for peer-to-peer (see 8.6.2b) and subscriber-to-publisher (see 8.6.2c)) communications when the DLPDU-authentication attribute is SOURCE or MAXIMAL. Formats 2L and 2S are used for peer-to-peer and subscriber-to-publisher communications when the DLPDU-authentication attribute is ORDINARY.

8.6.3 Sending the ED DLPDU

An ED DLPDU may be selected for transmission on the link when

- a) the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link;
- b) the remaining allocated duration of token usage, C(RD), permits completion of V(MRC)+1 implied transactions prior to expiration of the token, where each transaction consists of sending the ED DLPDU that requires an immediate reply, and awaiting a worst-case SR DLPDU or worst-case permitted DT reply DLPDU not containing DLS-user data;
- c) if the ED DLPDU will be addressed to a DLSAP-address, then the outstanding-transaction-array, V(OTA) (see 6.7.1.15), searched circularly from the last-transaction-index, V(LTI) (see 6.7.1.16), has an unassigned entry whose index is not V(LTI).

Once selected, if the ED DLPDU will be addressed to a DLSAP-address, then

- V(LTI) shall be set to the index of that unassigned entry in V(OTA);
- that entry in V(OTA) shall be assigned to the selected ED DLPDU and shall record information which permits an expected response DT DLPDU to be correlated with the specific invocation of the unitdata-exchange service which gave rise to the ED DLPDU.

Once selected, transmission of the ED DLPDU shall be retried until either

- 1) a permissible immediate reply DLPDU is received, or
- 2) an impermissible DLPDU is received when an immediate reply was expected, or
- 3) the original transmission and the permitted maximum number of transmission retries, V(MRC) (see 6.7.1.5), have all failed to elicit one of the permissible reply DLPDUs.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU and awaiting its immediate reply is needed at that time, then the DLE may set the final-token-use subfield of the ED DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

NOTE The FINAL transaction is necessarily the (V(MRC)+1)'th in a series; otherwise, if a reply is not received, then another ED DLPDU would have to be sent before the current cycle of token use was completed, making the current transaction NOT-FINAL.

Each explicit DL-address in the ED DLPDU shall be delocalized (see 7.2.2) before transmission.

After sending an ED DLPDU, the sending DLE shall monitor the local link for a reply as specified in 7.2.7.1. The permissible reply DLPDU is either

- a DT DLPDU whose destination DL-address is implicitly or explicitly the originating DLSAP-address or DLCEP specified by the ED DLPDU, or
- a DT DLPDU without a destination DL-address, or
- an SR DLPDU.

If V(LTI) was assigned to the transaction, as a result of the search described in c), and if a permissible reply DLPDU was not received, then the V(LTI)'th entry shall be deassigned.

8.6.4 Receiving the ED DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received ED DLPDU shall be treated as follows by the receiving DLE.

8.6.4.1 Actions required of all DLEs

NOTE The next alternative attempts to detect the reception of a duplicated ED DLPDU resulting from an immediate retry by the current token-holding DLE, which itself was probably caused by an error detected during receipt of the earlier reply DT DLPDU. In such a case the response DT DLPDU is required to be identical to the first in those fields and subfields which convey DLS-user data and related information.

a) If

- 1) the destination DL-address specified by the DLPDU designates an active DLSAP-address of the receiving DLE, or an active DLCEP-address of a DLC for which the receiving DLE is peer or publisher;
- 2) the immediately-prior DLPDU was transmitted as an immediate-response to a received ED DLPDU whose destination DL-address was the same DLSAP-address or DLCEP-address;
- 3) if a source DL-address is present in the just-received ED DLPDU, then it was also present and identical in that prior received ED DLPDU;

- 4) starting with the SD-parameters field, the first three octets of the just-received ED DLPDU, or the remainder of the DLPDU if fewer than three octets, are identical to the corresponding octets of that prior received ED DLPDU;
- 5) a period of link inactivity of (immediate-response-recovery-delay + 1) slot-times, $(V(IRR) + 1) \times V(ST)$ octet-durations, has not occurred since receipt of that prior received ED DLPDU,

then the receiving DLE

- i) shall discard the received DLPDU and not forward it to the DLE's upper-level functions for further processing;
- ii) shall retransmit the prior-transmitted immediate-reply DT DLPDU, unchanged from the prior transmission except for
 - the final-token-use subfield of the frame control octet which conveys information to the listening LAS;
 - those subfields of any present DLCEP SD-parameters which acknowledge receipt of, or request retransmission of, received DLSDUs.

NOTE This requirement is meant to ensure that the DL-priority, content and identity of any DLS-user data conveyed in the immediate reply DT DLPDU is identical to that in the immediately-prior DT DLPDU sent from this same DL-address, including the originator's transaction-id if one is present in the received and transmitted SD-parameters.

- b) If a) does not apply, and the destination DL-address specified by the DLPDU designates a DLSAP-address of the receiving DLE, then the processing of the received DLPDU shall be based upon the DL(SAP)-role specified for that DLSAP-address.

NOTE 1 Group DL-addresses are not covered here; such ED DLPDUs are erroneous and are unrecognized upon receipt.

The receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the ED DLPDU:

- 1) If the DLE can prepare it in time, then the reply DLPDU shall be a DT DLPDU
 - with any explicit destination DL-address in the DT DLPDU having the same value as the source DL-address of the received ED DLPDU, and length and format as follows:
 - DT (see 8.7.1 format 2L) in response to ED (see 8.6.1 format 1L), or
 - DT (see 8.7.1 format 2S) in response to ED (see 8.6.1 format 1S) if the DT DLPDU contains DLS-user data, or
 - DT (see 8.7.1 format 4) in response to ED (see 8.6.1 format 1S) if the DT DLPDU does not contain DLS-user data,

and shall include SD-parameters specifying

- the originator's transaction-id as received in the stimulating ED DLPDU, as specified in 8.6.2.1a)4);
- reception error status for that ED-conveyed DLSDU if the received DLSDU could not be buffered or queued; otherwise status for the DLSDU requested by that ED DLPDU.

The received DLSDU shall be forwarded to the DLE's upper-level functions (see 10.3) for further processing.

If the received DLSDU was buffered or queued successfully, then the reply DLPDU may include a DLSDU which was already buffered at that responding DLE at the time of ED reception.

NOTE 2 This restriction prohibits the reply DLPDU from reflecting any higher-layer (than DLL) processing of the received DLSDU. This restriction is necessary to enable migration from prior national standards.

If the DL(SAP)-role of the destination DLSAP-address specified

i) BASIC or INITIATOR, then the DLE shall reject the received ED DLPDU based on this DL(SAP)-role, the received DLSDU shall be discarded, and an appropriate error status without an accompanying DLSDU shall be included in the reply DT DLPDU;

ii) UNCONSTRAINED RESPONDER, then

A) if no receiving buffer or queue is explicitly bound to the destination DLSAP-address at the DLL priority of the received DLSDU, or if a receiving queue is explicitly bound but is full, then the received DLSDU shall be discarded, and an appropriate error status without an accompanying DLSDU shall be included in the reply DT DLPDU; or

B) otherwise the received DLSDU shall be put into that buffer or appended to that not-full queue and the SD-parameter status shall indicate the highest DLL priority sending buffer with a non-null DLSDU which was available at the addressed DLSAP at the time of reception of the ED DLPDU, and

I) if that DLL priority is greater than or equal to the priority specified in 8.6.1.1b), then that DLSDU shall be included in the reply DT DLPDU and its buffer set to empty if so configured, and the DLL priority of that reply DT DLPDU shall be the DLL priority of the conveyed DLSDU;

II) if that DLL priority is less than the priority specified in 8.6.1.1b), then no DLSDU shall be included in the reply DT DLPDU, and the DLL priority of that reply DT DLPDU shall be the DLL priority specified in 8.6.1.1b); or

III) if there is no such sending buffer with a non-null DLSDU, then the reply DT DLPDU shall specify an appropriate error status, the DLL priority of that reply DT DLPDU shall be the DLL priority specified in 8.6.1.1b), and that reply DT DLPDU shall not contain DLS-user data.

iii) CONSTRAINED RESPONDER, then

A) if the source DLSAP-address of the ED DLPDU is equal to the remote DLSAP address which was specified in the prior DL-BIND request primitive for the receiving DLSAP-address (or its DL-management equivalent), both after delocalization, then the procedure specified in ii) shall be followed; or

B) otherwise, the received DLSDU shall be discarded, and an appropriate error status without an accompanying DLSDU shall be included in the reply DT DLPDU.

- 2) If the DLE cannot prepare the required reply DT DLPDU in time, then the DLE shall send a DT DLPDU with any explicit destination DL-address in the DT DLPDU having the same value as the source DL-address of the received ED DLPDU, and length and format as follows:

- DT (see 8.7.1 format 2L) in response to ED (see 8.6.1 format 1L), or
- DT (see 8.7.1 format 4) in response to ED (see 8.6.1 format 1S),

and shall include SD-parameters

- with a transaction-id identical to the transaction-id from the received ED DLPDU, as specified in 8.6.2.1a)4);
- with a status (DR — “delayed reply”) indicating that the DLE requires additional time to prepare the required response;
- with a null user data field,

and the DLE

- shall prepare that required DT DLPDU as soon as possible, as just described;
- shall include an explicit destination address in the reply DLPDU;
- shall append that reply DLPDU to the DLE’s Q(US) to be transmitted at the first opportunity.

- c) If a) does not apply, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE
- is a peer or publisher, and the DLL priority of the DLCEP is not equal to the priority specified in the received DLPDU; or
 - is a peer, and the length and number of DL-addresses is not as expected or the DLPDU specifies an explicit source address which is not equal to the remote peer DLCEP’s DLCEP-address,

then

- 1) the received DLPDU shall be forwarded to the DLE’s upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the CD DLPDU. The reply DLPDU shall be a DT DLPDU in the format negotiated for the DLC for the selected direction of transmission, and shall contain SD-parameters appropriate to the sending DLCEP, but shall not contain DLS-user data.
- d) If neither a) nor b) applies, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE is a peer or publisher, then
- 1) the received DLPDU shall be forwarded to the DLE’s upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the ED DLPDU. The reply DLPDU shall be a DT DLPDU in the format negotiated for the DLC for the selected direction of transmission, shall contain SD-parameters appropriate to the sending DLCEP, and shall contain DLS-user data, if any

- was available and waiting for transmission or retransmission from the DLCEP;

— can be included in the DT DLPDU while still meeting the required maximum reply delay;

- 3) the requesting ED DLPDU may contain information about the state of the requesting DLCEP. The reply DT DLPDU is permitted, but is not required, to reflect that state information in its reply; immediate processing of that state information before sending the immediate reply shall be permitted but shall not be required.

The received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing.

- e) If a) does not apply, and the destination DL-address specified by the DLPDU designates an active DLCEP-address of a DLC for which the receiving DLE is a subscriber, then
 - 1) the non-DLS-user data portion of the received DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2.2.9) for further processing;
 - 2) the receiving DLE shall record the destination DL-address from the received ED DLPDU in V(RA) for subsequent association with the expected immediate reply DT DLPDU, which should be the next DLPDU received;
 - 3) the receiving DLE shall monitor the local link for a reply and then act based on the result of that monitoring, all as specified in 7.2.7.3 .

8.6.4.2 Additional actions required of a Link-Master class DLE

None.

8.6.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.6.4.2 also apply to a Bridge class DLE.
- b) If all of the following conditions hold
 - A) the destination DL-address specified in the DLPDU is one which the bridge should forward but to which the bridge DLE itself would not otherwise generate an immediate reply DLPDU;
 - B) the immediately-prior DLPDU was transmitted as an immediate-response to a received ED DLPDU with the same destination DL-address;
 - C) if a source DL-address is present in the just-received ED DLPDU, then it was also present and identical in that prior received ED DLPDU;
 - D) the basic-DLC-parameters portion of the SD-parameters of the just-received ED DLPDU is not null, and is identical to that of that prior received ED DLPDU;
 - E) a period of link inactivity of immediate-response-recovery-delay plus one slot-time, $(V(IRR\!D)+1) \times V(ST)$ octet-durations, has not occurred since receipt of that prior received ED DLPDU,

then the bridge

- 1) shall discard the received DLPDU and not forward it to the bridge's other functions for further processing;
- 2) shall initiate retransmission of the prior-transmitted immediate-reply SR DLPDU, unchanged from the prior transmission.

NOTE This requirement is meant to ensure that any status conveyed in the immediate reply SR DLPDU is identical to that in the immediately-prior SR DLPDU.

Otherwise,

- i) if the destination DL-address specified in the DLPDU is one which the bridge should forward but which the bridge DLE itself would not otherwise receive, then the bridge shall form and send an SR DLPDU
 - within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the ED DLPDU;
 - with status indicating whether or not the bridge was able to buffer the received DLPDU.
 - ii) if the destination DL-address specified in the DLPDU is one which the bridge should forward, and the bridge was able to receive and buffer the DLPDU without error, then the received DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).
 - iii) if the DLPDU contains an explicit source DL-address, then the bridge shall attempt to update its routing table entry for the source DL-address specified in the DLPDU to reflect the bridge port from which the DLPDU was received.
- c) It is a protocol error for a bridge DLE which will forward the received ED DLPDU not to send an SR reply DLPDU when a reply is required.

NOTE At most one bridge DLE on the local link should be forwarding the received ED DLPDU.

8.6.4.4 Additional actions required of the current LAS DLE

The LAS DLE shall act as specified in 7.2.7.2.

8.7 Data (DT) DLPDU

A DATA (DT) DLPDU is used to transfer a limited amount of transparent user data from one DLS-user to one or more other DLS-users; to acknowledge the transfer of such data; and to assist in the synchronization of DLCEPs and of DLS-users.

It is also used by a responding DLE when replying to a received CA, CD or ED DLPDU, when the time permitted by $V(ST)$ and $V(MRD)$ (see 6.7.1.1 and 6.7.1.2) is inadequate for that DLE to generate the required response to the received DLPDU.

It is also used by a DLE to send an SPDU to one or more other DLEs.

8.7.1 Structure of the DT DLPDUs

Table 18 – Structure of DT DLPDUs

format	Frame control	Destination address	Source address	Parameters	User data
1L	1101 1FPP	HL.N.S	HL.N.S	SD-p	o-pDLSDU
1S	1101 0FPP	N.S	N.S	SD-p	o-pDLSDU
2L	1001 1FPP	HL.N.S	—	SD-p	o-pDLSDU
2S	1001 0FPP	N.S	—	SD-p	o-pDLSDU
3L	0101 1FPP		HL.N.S	SD-p	o-pDLSDU
3S	0101 0FPP		N.S	SD-p	o-pDLSDU
4	1001 0F00	[PSA]	—	SD-p	o-pDLSDU
5	0101 0F00		[PDA]	SD-p	o-pDLSDU
Key: [PDA] is the destination DL-address from the immediately-prior CA, CD or ED DLPDU [PSA] is the implied source DL-address from the immediately-prior CA, CD or ED DLPDU					

8.7.1.1 Frame control field

The frame control field shall specify

- the DLPDU's function;
- the DLPDU's priority;
- the length, number and type of DLPDU addresses;
- whether or not transmission of the current DLPDU terminates use of a delegated token.

8.7.1.2 Address field

The address field shall consist of either

- an explicit destination DL-address and an explicit source DL-address, in that order, for formats 1L and 1S,
- only an explicit destination DL-address, for formats 2L and 2S,
- only an explicit source DL-address, for formats 3L and 3S,
- only an implied destination DL-address, for format 4, or
- only an implied source DL-address, for format 5.

For formats 1L, 2L and 3L, all addresses shall be LONG; for formats 1S, 2S and 3S, all addresses shall be SHORT; for formats 4 and 5 all addresses shall be VERY-SHORT (see 7.2.1.1) (that is, implicit).

8.7.1.3 Parameters field

When the destination DL-address explicitly specifies a group DL-address, the status-data-parameters (SD-parameters) field is null (formats 1L and 1S), as described in 9.4.1.

When the destination DL-address explicitly or implicitly specifies a DLSAP-address, the SD-parameters field specifies a transaction-id and status (formats 2L, 2S, 4), or is null (formats 1L and 1S), as described in 9.4.2.

When the first DL-address explicitly or implicitly specifies a DLCEP-address, the SD-parameters field specifies state information for the DLCEP addressed by the destination DL-address (formats 1L, 1S, 2L, 2S, 4), or source DL-address (formats 3L, 3S, 5), as described in 9.4. The size and structure of this field is dependent on the QoS attributes associated with the addressed DLCEP, and is determined during DLCEP establishment.

8.7.1.4 User data field

The user data field shall consist of a single or partial optional DLSDU whose maximum size is limited to the smaller of

- a) the maximum DLS-user data size permitted for a DLPDU of the priority specified in 8.7.1.1b);
- b) when the DLPDU's explicit or implied destination (or source) address is a DLCEP-address, the maximum DLSDU size negotiated on the DLC for this direction of data transmission to (or from) that DLCEP.

8.7.2 Content of the DT DLPDU

The frame control field shall be encoded as specified in Table 18.

Either the DL-addresses shall be

- a) the first a group DL-address and the second a DLSAP-address,
- b) all DLSAP-addresses,
- c) all DLCEP-addresses, or
- d) the first a DLCEP-address and the second a DLSAP-address.

8.7.2.1 Content of the DT DLPDU when specifying a destination DL(SAP)-address

When the first address is a group DL-address as in 8.7.2a), then

- a) If the DLPDU format is format 1L or 1S, then
 - 1) the DLPDU is being used to implement the unitdata transfer service;
 - 2) the DL(SAP)-role for the destination DLSAP-address shall be GROUP;
 - 3) the DL(SAP)-role for the source DLSAP-address shall be BASIC;
 - 4) the SD-parameters field shall be null;
 - 5) the user data shall be a single DLSDU whose size is limited to the maximum size for the priority specified in 8.7.1.1b), and shall not be null.
- b) No other DLPDU format may be used.

When the addresses are DLSAP-addresses as in 8.7.2b), then

- c) If the DLPDU format is format 1L or 1S, then
 - 1) this DLPDU is being used to implement the unitdata transfer service;
 - 2) the DL(SAP)-role for the destination DLSAP-address shall be BASIC;
 - 3) the DL(SAP)-role for the source DLSAP-address shall be BASIC;
 - 4) the SD-parameters field shall be null;
 - 5) the user data shall be a single DLSDU whose size is limited to the maximum size for the priority specified in 8.7.1.1b), and shall not be null.
- d) If the DLPDU format is format 2L, 2S or 4, then
 - 1) this DLPDU conveys an acknowledgment or reply as part of the unitdata transfer (with remote-DLE-confirmation) service, or the unitdata exchange service;
 - 2) the DL(SAP)-role for the destination DLSAP-address shall be BASIC or INITIATOR;

- 3) the DL(SAP)-role for the implicit source DLSAP-address is BASIC or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER;
- 4) the SD-parameters field shall specify a transaction-id and reply status for the addressed DLSAP, and the contents of this field shall be as described in 9.4;
- 5) the user data either shall be a single DLSDU whose size is limited to the maximum size for the priority specified in 8.7.1.1b), or shall be null, as follows:
 - formats 2L may be null or non-null;
 - format 2s shall always be non-null;
 - format 4 shall always be null.

e) No other DLPDU format may be used.

8.7.2.2 Content of the DT DLPDU when specifying a destination or source DLCEP-address

When the first address is a DLCEP-address, as in 8.7.2c) and 8.7.2d), then

- a) this DLPDU can convey a single or partial DLSDU
 - from one peer DLCEP to its corresponding peer DLCEP,
 - from a subscriber DLCEP to its corresponding publisher DLCEP, or
 - from a publisher DLCEP to its corresponding subscriber DLCEPs

and

- b) the second address, if present,
 - shall be the peer DLCEP-address of the same DLC as the destination DLCEP-address, or
 - shall be a subscriber DLCEP's calling-DLSAP-address;
- c) the SD-parameters field shall specify state information for the addressed DLCEP, and the contents of this field shall be as described in 9.4.2;

NOTE The size and structure of this field is dependent on the QoS attributes associated with the DLCEP addressed by the destination DL-address specified in this DLPDU, and is determined during DLCEP establishment.

- d) the user data shall specify those octets of a DLSDU consistent with the negotiated DLSDU size and the segmentation information specified in the accompanying SD-parameters, and may be null.

NOTE 1 Formats 1L, 2L, 1s and 2s are used for peer-to-peer and subscriber-to-publisher communications; 1L is used when the DLPDU-authentication attribute is SOURCE or MAXIMAL, 1s is used when the DLPDU-authentication attribute is SOURCE, 2L and 2s are used when the DLPDU-authentication attribute is ORDINARY.

Format 3L is used for publisher-to-subscriber communications when the DLPDU-authentication attribute is MAXIMAL. Formats 3L and 3s are used for publisher-to-subscriber communications when the DLPDU-authentication attribute is ORDINARY or SOURCE.

The specific format to be used (of formats 1L to 3s) is determined as part of DLCEP establishment.

NOTE 2 Formats 4 and 5 can be used instead of formats 2s and 3s, respectively, only when the sending DLE holds a reply token and when the DLPDU-authentication attribute is ORDINARY.

8.7.3 Sending the DT DLPDU

A DT DLPDU may be selected for transmission on the link when the sending DLE

- a) has just received a reply token in a CA, CD or ED DLPDU, permitting a single transmission of a DT or SR DLPDU; or
- b) holds a scheduler token or delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, C(RD), permits completion of the DT DLPDU's transmission prior to expiration of the token.

Each explicit DL-address in the DLPDU shall be delocalized (see 7.2.2) before transmission.

8.7.3.1 Transmission when the reply token is dominant

- a) A DT DLPDU may be sent on the link when the sending DLE has received a CD or ED DLPDU addressed

- to one of its active DLSAP-addresses, or
- to one of its active DLCEP-addresses for which it has a peer or publisher DLCEP,

and the sending DLE is replying as specified by 8.5.4.1 or 8.6.4.1, by forming as an immediate reply a DT DLPDU which may include a DLSDU which was already buffered or queued at that responding DLE at the time of the CD or ED DLPDU's reception.

NOTE This restriction prohibits the reply DLPDU from reflecting any higher-layer processing of the received DLSDU. This restriction is necessary to enable migration from prior national standards.

- b) A DT DLPDU may be sent on the link when the sending DLE has received a CA DLPDU addressed

- to one of its active DLSAP-addresses, or
- to one of its active DLCEP-addresses for which it has a peer or publisher DLCEP,

and the sending DLE is replying as specified by 8.4.4.1, by forming as an immediate reply a DT DLPDU which does not include a DLSDU.

When an immediate reply to a CA, CD or ED DLPDU is required, as specified in a) or b), then the replying DLE shall send a reply DT DLPDU within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the requesting CA, CD or ED DLPDU.

If the CA, CD or ED DLPDU which required the reply was addressed to a DLCEP, then that CA, CD or ED DLPDU may have contained SD-parameters which conveyed information about the state of the sending DLCEP to the receiving DLCEP. The receiving DLE is permitted, but is not required, to reflect that state information in its reply DT DLPDU; immediate processing of that state information before sending the immediate reply shall be permitted but shall not be required.

The final-token-use subfield of the reply DT DLPDU shall have the same value as that in the requesting CA, CD or ED DLPDU.

Each explicit DL-address in the reply DT DLPDU shall be delocalized (see 7.2.2) before transmission.

It is a protocol error for the addressed DLE not to send a DT reply DLPDU when a reply is required.

NOTE At most one DLE on the local link should be sending a reply to the received CA, CD or ED DLPDU. That reply may be either a DT or SR DLPDU.

8.7.3.2 Transmission when the delegated token is dominant

A DT DLPDU may be sent on the link when the sending DLE holds a delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, $C(RD)$, permits completion of the DLPDU's transmission prior to expiration of the token.

If no additional use of that token after sending this DLPDU is needed at that time, then the DLE may set the final-token-use subfield of the DT DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

Each explicit DL-address in the DLPDU shall be delocalized (see 7.2.2) before transmission.

8.7.3.3 Transmission when the scheduler token is dominant

The LAS DLE may send a DT DLPDU on the link when both the remaining allocated duration of token usage before the next scheduled activity, and the remaining allocated duration of scheduler token usage for link maintenance, which is $V(LTHT)$ octet-durations minus the amount of link capacity used for link maintenance during the current cycle of “circulating the token”, permits completion of the DLPDU’s transmission prior to expiration of the token.

Each explicit DL-address in the DLPDU shall be delocalized (see 7.2.2) before transmission.

8.7.4 Receiving the DT DLPDU

Each DL-address in the DLPDU shall be delocalized (see 7.2.2) upon reception.

A received DT DLPDU shall be treated as follows by the receiving DLE:

8.7.4.1 Actions required of all DLEs

8.7.4.1.1 Actions required when the reply token was not dominant at start-of-reception

- a) If the received DT DLPDU has format 1L or 1S, and its destination DL-address designates a DL(SAP)-address of the receiving DLE, then the received DLPDU shall be forwarded to the DLE’s upper-level functions (see 10.3.1.3) for further processing.
- b) If the received DT DLPDU has format 2L or 2S, and its destination DL-address designates a DLSAP-address of the receiving DLE, and the transaction-id reported by the SD-parameters indexes an assigned transaction in the outstanding-transaction-array, $V(OTA)$ (see 6.7.1.15), then
 - the DLE shall consider the prior transmission to have been error-free;
 - the information in the indexed entry in $V(OTA)$ shall be associated with the received DT DLPDU;
 - the received DT DLPDU and that associated information shall be forwarded to the DLE’s upper-level functions (see 10.3) for further processing;
 - the indexed entry in $V(OTA)$ shall be unassigned.
- c) If the received DT DLPDU
 - 1) has format 1L, 1S, 2L or 2S, and its destination DL-address designates a DLCEP-address designating a peer or publisher DLCEP of the receiving DLE; or
 - 2) has format 3L or 3S, and its source DL-address designates a DLCEP-address designating a subscriber DLCEP of the receiving DLE,

then the received DLPDU shall be forwarded to the DLE’s upper-level functions (see 10.2) for further processing.
- d) If the received DT DLPDU has format 1L or 1S, and its destination DL-address designates NODE.0 DL-address, $V(TN).0$, of the receiving DLE’s DL-support functions, then the received DLPDU shall be forwarded to the DLE’s upper-level functions (see 12.3) for further processing.
- e) If none of a), b), c) or d) applies, then the DT DLPDU shall be reported to local DL-management as an unexpected response, and shall be discarded.

NOTE This report may take the form of incrementing a DL-management error counter.

8.7.4.1.2 Actions required when the reply token was dominant at start-of-reception and the receiving DLE sent the CA, CD or ED DLPDU which created the reply token

- a) If the received DT DLPDU
 - has format 2L or 2S, and its destination DL-address designates a DLSAP-address of the receiving DLE; or

- has format 4, and the source DL-address from the immediately-prior CA, CD or ED DLPDU was a DLSAP-address,

then

- 1) If the status reported by the SD-parameters is “DR — delayed reply” (that is, has a value of F₁₆), then the DLE

- shall consider the prior transmission to have been error-free;
- shall discard the DLPDU.

NOTE The DLE should expect to receive another DT DLPDU at a later time, with the same destination DL-address and format, as a delayed reply.

- 2) If 1) does not apply, then

- the DLE shall consider the prior transmission to have been error-free;
- the information in the V(LTI)'th entry in V(OTA) (see 6.7.1.16 and 6.7.1.15) shall be associated with the received DT DLPDU;
- the received DT DLPDU and that associated information shall be forwarded to the DLE's upper-level functions (see 10.3) for further processing;
- the V(LTI)'th entry in V(OTA) shall be deassigned.

- b) If the received DT DLPDU has format 1L, 1S, 2L or 2S, and its destination DL-address designates a DLCEP-address of a peer DLCEP of the receiving DLE, then
 - the DLE shall consider the prior transmission to have been error-free;
 - the received DT DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing.
- c) If the received DT DLPDU has format 4, and the explicit or implicit source DL-address from the immediately-prior CA, CD or ED DLPDU was a DLCEP-address of a peer DLCEP of the receiving DLE, then
 - the DLE shall consider the prior transmission to have been error-free;
 - the received DT DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing, with its implied destination DLCEP-address assumed to be the explicit or implicit source DL-address from that immediately-prior CA, CD or ED DLPDU.
- d) If the received DT DLPDU has format 3L or 3S, and its source DL-address designates the publisher's DLCEP-address of a subscriber DLCEP of the receiving DLE, and this source DL-address is equal to the destination DL-address from the immediately-prior CA, CD or ED DLPDU, then
 - the DLE shall consider the prior transmission to have been error-free;
 - the received DT DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing.
- e) If the received DT DLPDU has format 5, and the explicit destination DL-address from the immediately-prior CA, CD or ED DLPDU was a DLCEP-address of a subscriber DLCEP of the receiving DLE, then
 - the DLE shall consider the prior transmission to have been error-free;
 - the received DT DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing, with its implied source DLCEP-address assumed to be the explicit destination DL-address from that immediately-prior CA, CD or ED DLPDU.
- f) If none of a) to e) applies, then the DT DLPDU shall be reported to local DL-management as an unexpected response, and shall be discarded.

NOTE This report may take the form of incrementing a DL-management error counter.

8.7.4.1.3 Actions required when the reply token was dominant at start-of-reception and the receiving DLE did not send the CA, CD or ED DLPDU which created the reply token

- a) If the received DT DLPDU has format 3L or 3S, and its source DL-address designates a publisher's DLCEP-address of a subscriber DLCEP of the receiving DLE, and this source DL-address is equal to the destination DL-address, V(RA), from the immediately-prior CA, CD or ED DLPDU, then
 - the DLE shall consider the prior transmission to have been error-free;
 - the received DT DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing.
- b) If the received DT DLPDU has format 5, and the explicit destination DL-address from the immediately-prior CA, CD or ED DLPDU was the publisher's DLCEP-address of a subscriber DLCEP of the receiving DLE, then
 - the DLE shall consider the prior transmission to have been error-free;
 - the received DT DLPDU shall be forwarded to the DLE's upper-level functions (see 10.2) for further processing, with its implied source DLCEP-address assumed to be the explicit destination DL-address from that immediately-prior CA, CD or ED DLPDU.

8.7.4.2 Additional actions required of a Link-Master class DLE

None.

8.7.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.7.4.2 also apply to a Bridge class DLE.
- b) If the first DL-address specified in the DLPDU is an explicit DL-address to which the bridge should forward the DLSDU and the bridge was able to buffer the DLPDU without error, then the DLPDU shall be forwarded with modification of the frame-control field in the forwarded DLPDU as appropriate (see 5.1.3).
- c) If the DLPDU contains an explicit source DL-address, then the bridge shall attempt to update its routing table entry for the source DL-address specified in the DLPDU to reflect the bridge port from which the DLPDU was received.
- d) Otherwise, if neither b) nor c) applies, then the DLE shall not forward the DLPDU.

8.7.4.3.1 Actions required when the reply token was dominant at start-of-reception and the receiving bridge DLE forwarded, but did not originate, the CA, CD or ED DLPDU which created the reply token

- a) The DLE shall consider the prior transmission to have been error-free.
- b) If
 - the received DT DLPDU has format 4;
 - the destination DL-address from the immediately-prior CA, CD or ED DLPDU was a DLSAP-address;
 - the status reported by the SD-parameters is other than "DR — delayed reply" (that is, has a value other than F₁₆);
 - the user data field is null,

then the DLE shall form a DT DLPDU

- 1) with format 2L when the previous CA, CD or ED DLPDU was format 1L, or format 2S when the previous CA, CD or ED DLPDU was format 1S;
- 2) with the explicit destination DL-address equal to the source DL-address in that immediately prior CA, CD or ED DLPDU;
- 3) with an SD-parameters field equal to the SD-parameters field of the received DT DLPDU;

4) with a null user data field;

and shall forward the just-formed DT DLPDU as if it had just been originated by the upper-level functions of the bridge DLE.

NOTE This means that the bridge DLE originates the FCS for the newly-formed DT DLPDU as in 5.1.1, rather than just modifying a previously received FCS during normal forwarding as in 5.1.3.

8.7.4.4 Additional actions required of the current LAS DLE

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
- b) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.8 Status Response (SR) DLPDU

A STATUS RESPONSE (SR) DLPDU is sent only while holding a reply token; it is used

- a) to indicate the receipt of the immediately prior CA, CD or ED DLPDU by the bridge which would normally forward that DLPDU toward the addressed DLE, to indicate to the sending DLE that no error occurred, or that the indicated error occurred;
- b) to reject an attempted transfer of the LAS role from the current LAS DLE to another link-master DLE.

8.8.1 Structure of the SR DLPDU

Table 19 – Structure of SR DLPDUs

Frame control	Destination address	Source NODE-address	Parameters
0001 0F11	[PSA]	N	o-SR-p
Key: [PSA] is the implied DL-address equal to the implied or explicit source DL-address of the immediately prior DLPDU on the link			

8.8.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is that of the immediately-prior DLPDU on the local link (to which the SR DLPDU is an immediate response);
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.8.1.2 Address field

The address field shall consist of an explicit source NODE DL-address.

8.8.1.3 Parameters field

The delayed-reply-parameters (SR-parameters) field specifies an optional transaction id and status. When non-null, this field shall be structured and encoded as described in 9.5.

8.8.1.4 User data field

The user data field shall be null.

8.8.2 Content of the SR DLPDU

The frame control field shall be encoded as specified in Table 19:

- When sent as an immediate reply to a TL DLPDU, the final-token-use subfield of the SR DLPDU shall specify FINAL.
- When sent as an immediate reply to a CA, CD or ED DLPDU, the final-token-use subfield of the SR DLPDU shall have the same value as that in the requesting CA, CD or ED DLPDU.

The address field shall be the replying DLE's NODE DL-address, whose value is the sender's node-id, V(TN).

NOTE The primary purposes for the inclusion of the NODE DL-address is to identify the DLE which has assumed the reply token to any observing link analyzer.

When sent as an immediate reply to a CA, CD or ED DLPDU,

- a) the SR-parameters shall be null when the DLE is a bridge DLE which has been able to accept the transaction-initiating DLPDU for forwarding;
- b) otherwise, when a) does not apply, the SR-parameters shall be encoded as described in 9.5.

When sent as an immediate reply to a TL DLPDU, the SR-parameters shall be encoded as described in 9.5.

8.8.3 Sending the SR DLPDU

- a) An SR DLPDU may be sent on the link when the sending DLE is a bridge DLE which has just received a reply token in a CA, CD or ED DLPDU, permitting that DLE to transmit a single DT or SR DLPDU, and the DLE is replying as specified by 8.4.4.3, 8.5.4.3 or 8.6.4.3.

It is a protocol error for a bridge DLE which will forward the received CA, CD or ED DLPDU to not send an SR reply DLPDU when a reply is required.

NOTE At most one bridge DLE on the local link should be forwarding the received CA, CD or ED DLPDU.

- b) An SR DLPDU may be sent on the link when the sending DLE is a link master or bridge DLE which has just received a reply token in a TL DLPDU, and the receiving DLE needs to reject the transfer of the LAS role (see 8.20.4.2).

The final-token-use subfield of the reply SR DLPDU shall have the same value as that in the requesting CA, CD or ED DLPDU, or shall have the value FINAL for a reply to a requesting TL DLPDU.

When an immediate reply to a CA, CD, ED or TL DLPDU is required, then the DLE shall reply within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, of receipt of the requesting CA, CD, ED or TL DLPDU.

8.8.4 Receiving the SR DLPDU

A received SR DLPDU shall be treated as follows by the receiving DLE:

8.8.4.1 Actions required of all DLEs

- a) A received SR DLPDU, received as a reply to an immediately prior CA, CD or ED DLPDU which was originated by the receiving DLE and which was addressed to a DLSAP-address, shall cause the receiving DLE

- 1) to consider the prior transmission to have been error-free;
- 2) to forward that received DLPDU to the DLE's upper-level functions (see 10.3 for further processing).
- b) A received SR DLPDU, received as a reply to an immediately prior TL DLPDU which was originated by the receiving (LAS) DLE shall cause the receiving DLE
 - 1) to consider the prior transmission to have been error-free;
 - 2) as specified in 8.20.3
 - to re-assume the scheduler token;
 - to inform local DL-management of the event;
 - to resume active operation as the LAS, and commence transmission on the link.

8.8.4.2 Additional actions required of a Link-Master class DLE

None.

8.8.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.8.4.2 also apply to a Bridge class DLE.
- b) A received SR DLPDU
 - 1) which reports a status other than "BOK — bridge OK" (that is, has a value other than E₁₆);
 - 2) which was received as a reply to an immediately prior CA, CD or ED DLPDU which was forwarded (and therefore not originated) by the receiving DLE and which was addressed to a DLSAP-address

shall cause the receiving DLE to form a DT DLPDU

- i) with format 2L when the previous CA, CD or ED DLPDU was format 1L, or format 2s when the previous CA, CD or ED DLPDU was format 1s;
- ii) with the explicit destination DL-address equal to the source DL-address in that immediately prior CA, CD or ED DLPDU;
- iii) with an SD-parameters field equal to the SR-parameters field of the received SR DLPDU;
- iv) with a null user data field;

and shall forward the just-formed DT DLPDU as if it had just been originated by the upper-level functions of the bridge DLE.

NOTE This means that the bridge DLE originates the FCS for the newly-formed DT DLPDU as in 5.1.1, rather than just modifying a previously-received FCS during normal forwarding as in 5.1.3.

- c) Otherwise, if b) does not apply, then the DLE shall not forward the DLPDU.

8.8.4.4 Additional actions required of the current LAS DLE

- a) If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall
 - 1) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
 - 2) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

- b) A received SR DLPDU, received as a reply to an immediately prior TL DLPDU which was originated by the LAS DLE, shall cause the LAS DLE
- 1) to consider the prior transmission to have been error-free;
 - 2) to terminate the attempted transfer of the LAS role;
 - 3) to inform DL-management of the event.

8.9 Compel Time (CT) DLPDU

A COMPEL TIME (CT) DLPDU is used by the current holder of a delegated token to request the LAS DLE to transmit as an immediate response a TIME DISTRIBUTION (TD) DLPDU, thus enabling all other DLEs on the local link

- to update their sense of DL-time;
- to synchronize the rates of advance of their senses of time.

8.9.1 Structure of the CT DLPDU

Table 20 – Structure of CT DLPDUs

Frame control
0001 0F00

8.9.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is URGENT;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU, and either its expected immediate reply or the appropriate link-idle timeout, terminates use of a delegated token.

8.9.1.2 Address field

The address field shall be null.

8.9.1.3 Parameters field

The parameters field shall be null.

8.9.1.4 User data field

The user data field shall be null.

8.9.2 Content of the CT DLPDU

The frame control field shall be encoded as specified in Table 20.

8.9.3 Sending the CT DLPDU

A DLE's need for the current DL-time, or to synchronize with the LAS DLE and that DLE's rate of advance of the current DL-time, may result in transmission of a CT DLPDU. A CT DLPDU requests that the LAS DLE send a TD DLPDU in immediate reply.

A CT DLPDU may be sent on the link when the sending DLE holds a delegated token whose remaining allocated duration of token usage, C(RD), permits completion of the implied transaction — sending the CT DLPDU, which requires an immediate reply, and awaiting the TD reply DLPDU — prior to return of the token.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU and awaiting its immediate reply is needed at that time, then the DLE may set the final-token-use subfield of the CT DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

After sending a CT DLPDU, the sending DLE shall monitor the local link for a reply as specified in 7.2.7.1. The permissible reply DLPDU is a TD DLPDU.

8.9.4 Receiving the CT DLPDU

A received CT DLPDU shall be treated as follows by the receiving DLE:

8.9.4.1 Actions required of all DLEs

None.

8.9.4.2 Additional actions required of a Link-Master class DLE

None.

8.9.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.9.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.9.4.4 Additional actions required of the current LAS DLE

The DLE shall reply immediately with a TD DLPDU.

If the final-token-use subfield of the received CT DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token has terminated and that the scheduler token is again dominant on the local link;
- b) treat that termination as if the token had been returned by an RT (see 8.17) DLPDU;
- c) resume active operation as the LAS.

8.10 Time Distribution (TD) DLPDU

A TIME DISTRIBUTION (TD) DLPDU is transmitted by the LAS DLE to enable the DLEs on the local link to coordinate and to synchronize the rates of advance of their senses of DL-time.

8.10.1 Structure of the TD DLPDU

Table 21 – Structure of TD DLPDUs

Frame control	Source NODE-address	Parameters
0001 0F01	N	TD-p

8.10.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is URGENT;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.10.1.2 Address field

The address field shall consist of an explicit source NODE DL-address.

8.10.1.3 Parameters field

The parameter field shall be structured and encoded as specified in 9.6 .

8.10.1.4 User data field

The user data field shall be null.

8.10.2 Content of the TD DLPDU

The frame control field shall be encoded as specified in Table 21.

The address field shall specify the node designator of the LM DLE which is co-resident with the LAS DLE.

The TD-parameters field shall be encoded as specified in 9.6 , and shall reflect the current values of the DLE's time-related variables and node-time counter at the moment of DLPDU formation, which shall be less than 1 s before DLPDU transmission.

The user data field shall be null.

8.10.3 Sending the TD DLPDU

A TD DLPDU may be sent on the link when the sending DLE (the LAS)

- a) has just received a reply token in a CT DLPDU, permitting a single transmission of a TD DLPDU; or
- b) holds a scheduler token which is the dominant token on the local link, and is required to send a TD DLPDU as specified in 10.4.1.6, and the remaining allocated duration of token usage before the next scheduled activity permits completion of the DLPDU's transmission prior to expiration of the token.

If the TD DLPDU is sent as a reply to a received CT DLPDU, then the final-token-use subfield of the reply TD DLPDU shall have the same value as that in the requesting CT DLPDU.

If the time-synchronization class of the sending DLE is not NONE, then after transmitting the TD DLPDU, the LAS DLE shall schedule another transmission of a TD DLPDU as specified in 10.4.1.2.

Reception of a CT DLPDU by the LAS DLE shall cause the LAS DLE to reply immediately with a TD DLPDU, sent to all other DLEs on the local link.

8.10.4 Receiving the TD DLPDU

A received TD DLPDU shall be treated as follows by the receiving DLE.

8.10.4.1 Actions required of all DLEs

- a) Each DLE that supports the variables, timers and counters defined in 6.7.1.19 to 6.7.1.26, other than the sending DLE, shall append to the received DLPDU the low-order 24 bits of the local node-time $C(NT)$, known thereafter as $N_R(NT)$, at which the TD DLPDU reception completed (that is, receipt of the END-OF-DATA or END-OF-DATA-AND-ACTIVITY PhIDU).
- b) Each DLE other than the sending DLE shall process the received DLPDU as specified in 10.4.1.3.

8.10.4.2 Additional actions required of a Link-Master class DLE

None.

8.10.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.10.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU. However, if the TD DLPDU was received at the port which is the bridge DLE's root port (that is, the port toward the root of the bridge spanning tree), then the DLE shall forward the DLPDU reception event to the bridge operation level (see 6.1.2).

8.10.4.4 Additional actions required of the current LAS DLE

If the DLE did not just send the DLPDU, then the DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
- b) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.11 Round-Trip-Delay Query (RQ) DLPDU

A ROUND-TRIP-DELAY QUERY (RQ) DLPDU is sent from one DLE to another on the local link to initiate the measurement and computation of the round-trip delay intrinsic to their inter-communication. Its receipt results in the return of a complementary ROUND-TRIP-DELAY REPLY (RR) DLPDU completing the measurement.

8.11.1 Structure of the RQ DLPDU

Table 22 – Structure of RQ DLPDUs

Frame control	Destination address	Source address	Parameters
1100 0F00	N.0	N.0	RQ-p

8.11.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is NORMAL;

- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU, and either its expected immediate reply DLPDU or the appropriate link-idle timeout, terminates use of a delegated token.

8.11.1.2 Address field

The address field shall consist of two explicit SHORT DL-addresses, destination and source, in that order.

8.11.1.3 Parameters field

The parameters field shall be structured and encoded as specified in 9.7.

8.11.1.4 User data field

The user data field shall be null.

8.11.2 Content of the RQ DLPDU

The frame control field shall be encoded as specified in Table 22.

The address field shall specify two NODE DL-addresses expressed in SHORT DL-address format, where the destination DL-address is

- the SHORT flat DL-address for the local LAS, 0400_{16} (see 6.3.3.2); or
- has the value formed by concatenating the intended receiver's node-id, $V_R(TN)$, and an octet of zero, thus specifying that DLE's DL-support functions;

and the source DL-address has the value formed by concatenating the sender's node-id, $V_S(TN)$, and an octet of zero, thus specifying the sending DLE's DL-support functions.

The RQ-parameters field shall be encoded as specified in 9.7, and shall reflect the value of the DLE's node-time counter, if maintained by the DLE, at a moment of DLPDU formation which is a constant time offset from the moment when the END-OF-DATA-AND-ACTIVITY PhIDU will be transmitted for the DLPDU. Otherwise, the DLE shall use any value for this field.

The user data field shall be null.

8.11.3 Sending the RQ DLPDU

An RQ DLPDU may be sent on the link when the sending DLE

- a) holds a delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, $C(RD)$, permits completion of the implied transaction — sending the RQ DLPDU, which requires an immediate reply, and awaiting the RR reply DLPDU — prior to expiration of the token; or
- b) holds a scheduler token which is the dominant token on the local link, and when both
 - the remaining allocated duration of token usage before the next scheduled activity;
 - the remaining allocated duration of scheduler token usage for link maintenance, which is $V(LTHT)$ octet-durations minus the amount of link capacity used for link maintenance during the current cycle of "circulating the token",
 permit completion of the implied transaction — sending the RQ DLPDU, which requires an immediate reply, and awaiting the RR reply DLPDU — prior to expiration of the token.

The RQ DLPDU shall be sent at NORMAL priority.

If the DLE holds a delegated token, and no additional use of that token after sending this DLPDU and awaiting its immediate reply RR DLPDU is needed at that time, then the DLE may set the final-token-use subfield of the RQ DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

After sending an RQ DLPDU, the sending DLE shall monitor the local link for a reply as specified in 7.2.7.1. The permissible reply DLPDU is an RR DLPDU.

8.11.4 Receiving the RQ DLPDU

A received RQ DLPDU shall be treated as follows by the receiving DLE:

8.11.4.1 Actions required of all DLEs

If the destination NODE DL-address specified by the DLPDU designates the receiving DLE, then

- a) when the receiving DLE's time-synchronization class (see 12.6.3) is NONE and the DLE does not maintain even an estimated C(NT), then the DLE shall append to the received DLPDU the 24-bit value zero; or
- b) else if a) does not apply, then the DLE shall
 - 1) append to the received DLPDU the low-order 24 bits of the local node-time, C(NT), at which the RQ DLPDU reception completed (that is, receipt of the END-OF-DATA or END-OF-DATA-AND-ACTIVITY PhIDU);
 - 2) adjust that 24-bit value to remove any systemic difference in the DLE-internal time delay between the DLE's transmit and receive paths, caused by known implementation considerations within the receiving real end system.

NOTE 1 This adjustment will probably vary inversely with the DLE's actual instantaneous rate of data transmission.

NOTE 2 The objective of this adjustment is to ensure that V(MD), as computed in 10.4.1.5, is as close as possible to double the sum of the one-way transmission, propagation, and reception delays incurred in transmission of a TD DLPDU between the addressed source node and the local DLE.

In either case the receiving DLE shall initiate a reply within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of receipt of the RQ DLPDU. The reply DLPDU shall be an RR DLPDU with a destination DL-address equal to the source DL-address of the received RQ DLPDU, and with a source DL-address specifying the replying DLE's DL-support functions.

8.11.4.2 Additional actions required of a Link-Master class DLE

None.

8.11.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.11.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.11.4.4 Additional actions required of the current LAS DLE

The LAS DLE shall act as specified in 7.2.7.2.

8.12 Round-Trip-Delay Reply (RR) DLPDU

A ROUND-TRIP-DELAY REPLY (RR) DLPDU is sent from one DLE to another on the local link to permit completion of the measurement and computation of the round-trip delay intrinsic to their inter-communication. It is only sent as an immediate reply to a received RQ DLPDU.

8.12.1 Structure of the RR DLPDU

Table 23 – Structure of RR DLPDUs

Frame control	Destination address	Source address	Parameters
1101 0F00	N.0	N.0	RR-p

8.12.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is NORMAL;
- c) the length, number and type of DLPDU addresses;
- d) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.12.1.2 Address field

The address field shall consist of two explicit SHORT DL-addresses, destination and source, in that order.

8.12.1.3 Parameters field

The parameters field shall be structured and encoded as specified in 9.8 .

8.12.1.4 User data field

The user data field shall be null.

8.12.2 Content of the RR DLPDU

The frame control field shall be encoded as specified in Table 23.

The address field shall specify two NODE DL-addresses expressed in SHORT DL-address format, where the destination DL-address is equal to the source DL-address received in the immediately prior RQ DLPDU, and the source DL-address has the value formed by concatenating the sender's node-id, $V_S(TN)$, and an octet of zero, thus specifying the sending DLE's DL-support functions.

The RR-parameters field shall be encoded as specified in 9.8 , and shall reflect the value of the DLE's node-time counter, if maintained by the DLE, at a moment of DLPDU formation which is a constant time offset from the moment when the END-OF-DATA-AND-ACTIVITY PhIDU will be transmitted for the DLPDU. Otherwise the DLE shall use the value zero for this field.

The user data field shall be null.

8.12.3 Sending the RR DLPDU

An RR DLPDU may be sent on the link when the sending DLE has just received a reply token in an RQ DLPDU, permitting a single transmission of an RR DLPDU. The final-token-use subfield of the reply RR DLPDU shall have the same value as that in the requesting RQ DLPDU.

8.12.4 Receiving the RR DLPDU

A received RR DLPDU shall be treated as follows by the receiving DLE.

8.12.4.1 Actions required of all DLEs

If the destination NODE DL-address specified by the DLPDU designates the receiving DLE, and if the immediately prior DLPDU was an RQ DLPDU sent by the receiving DLE, then the DLE shall consider the prior transmission to have been error-free, and if the DLE maintains C(NT), then the DLE shall

- a) append to the received DLPDU the low-order 24 bits of the local node-time, C(NT), at which the RR DLPDU reception completed (that is, receipt of the END-OF-DATA or END-OF-DATA-AND-ACTIVITY PhIDU);
- b) adjust that 24-bit value to remove any systemic difference in the DLE-internal time delay between the DLE's transmit and receive paths, caused by known implementation considerations within the receiving real end system;

NOTE 1 This adjustment probably will vary inversely with the DLE's actual instantaneous rate of data transmission.

NOTE 2 The objective of this adjustment is to ensure that V(MD), as computed in 10.4.1.5, is as close as possible to double the sum of the one-way transmission, propagation, and reception delays incurred in transmission of a TD DLPDU between the addressed destination node and the local DLE.

- c) and the DLE shall process the received DLPDU as specified in 10.4.1.5.

8.12.4.2 Additional actions required of a Link-Master class DLE

None.

8.12.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.12.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.12.4.4 Additional actions required of the current LAS DLE

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
- b) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.13 Probe Node DL-address (PN) DLPDU

A PROBE NODE DL-ADDRESS (PN) DLPDU is used by the LAS DLE to probe for the existence of previously unrecognized DLEs on the local link. A PN DLPDU creates and passes a reply token to any DLE(s) which may have their V(TN) (see 6.7.1.8) equal to the specified (configured but inactive) NODE DL-address. The only permissible reply is a PR DLPDU (see 8.14), which must be an immediate reply.

8.13.1 Structure of the PN DLPDU

Table 24 – Structure of PN DLPDUs

Frame control	Destination NODE-address	Parameters
0010 0110	N	PN-p

8.13.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the length, number and type of DLPDU addresses.

8.13.1.2 Address field

The address field shall consist of an explicit destination NODE DL-address.

8.13.1.3 Parameters field

The parameters field shall be structured and encoded as specified in 9.9.

8.13.1.4 User data field

The user data field shall be null.

8.13.2 Content of the PN DLPDU

The frame control field shall be encoded as specified in Table 24.

The address field shall specify the NODE DL-address which is being probed.

The Probe NODE DL-ADDRESS parameters (PN-parameters) field shall convey, as specified in 9.9, the then-current values for the local link of those DLE and PhE parameters necessary for a receiving DLE to configure itself and its associated PhE so that they can reply to this or a subsequent PN DLPDU.

The user data fields shall be null.

8.13.3 Sending the PN DLPDU

The LAS on its own, at the beginning of each cycle of link maintenance-related transmissions, and possibly during what otherwise would be unoccupied periods of link capacity, probes configured but apparently-unused NODE DL-addresses by sending PN DLPDUs to those DL-addresses and awaiting a reply. Should such a reply occur, it is an indication that a new DLE has joined (or rejoined) the link, and the LAS then adds that DLE's NODE DL-address

- to its live list, V(LL), to be polled with PT DLPDUs;
- to its token-circulation-list, V(TCL), if so requested in the responding PR DLPDU.

PN DLPDUs are not otherwise sent to NODE DL-addresses.

The LAS DLE shall send a PN DLPDU when probing NODE DL-addresses which are not associated with any known active DLEs on the local link. A PN DLPDU may be sent on the link by the LAS DLE when

- a) the LAS DLE is using the link capacity allocated for link maintenance as specified in clause 11; or
- b) the scheduler token is the dominant token on the local link; and the remaining allocated duration of token usage before the next scheduled activity permits completion of a PN - PR transaction as specified in clause 11).

Before sending the PN DLPDU, the LAS DLE shall determine the next NODE DL-address to be probed by choosing, on alternate instances of sending the PN DLPDU, from the following two sets of NODE DL-addresses, as defined in 6.3.2.2, and cyclically in increasing numeric order within each set:

- 1) $\{ 10_{16}..F7_{16} \} - V(LL) - \{ V(FUN)..(V(FUN) + V(NUN) - 1) \}$,
which is the set of configured bridge-class, link-master-class and basic-class DLE addresses, minus the set of active DLE addresses, and minus the set of excluded unused DLE addresses
- 2) $\{ F8_{16}..FF_{16} \} - V(LL)$,
which is the set of visitor DLE addresses, plus the set of DLE addresses for non-visitor DLEs which do not know their proper DLE addresses, minus the set of active DLE addresses.

After sending a PN DLPDU, the LAS DLE shall monitor local link activity for a period of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations, waiting for a reply. After this interval, as soon as the link is inactive, the LAS DLE shall continue with its other activities. Even when a response has been received, the LAS shall wait until the end of the immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations, after the PN transmission, and then wait for the link to become inactive, before continuing with its next transaction, to provide for a potential second response from another DLE which has the same NODE DL-address.

NOTE This requirement imposes timer considerations which differ subtly from all other uses of slot-time-based timers in the DL-protocol. For this timer, the timing action does not stop when ACTIVITY or DATA is reported on the link, but rather continues even after such reports.

If the LAS receives a DLPDU, which is not a PR DLPDU, during this monitoring interval, then the LAS DLE shall drop its scheduler token and inform local DL-management of the event.

8.13.3.1 Additional considerations

From the LAS DLE's perspective, the maximum overhead link capacity required by any PN DLPDU and an upper bound on the maximum permitted responses may be pre-computed as

$$\begin{aligned}
 & 2 \times \text{framing-overhead} \\
 & + 1 \times \text{PN-size} \\
 & + 1 \times V(IRRDR) \times V(ST) \\
 & + 1 \times \text{maximum-PR-size} \\
 & + 1 \times V(MID)
 \end{aligned}$$

8.13.4 Receiving the PN DLPDU

Receipt of a PN DLPDU by a newly attached or newly initialized DLE enables the DLE to learn those PhL and DLL configuration parameters which are essential for transmission on the local link.

Receipt of a PN DLPDU by the addressed DLE enables that DLE

- to indicate its presence and desired activity on the local link;
- to begin the process of being included in the link's other activities.

A received PN DLPDU shall be treated as follows by the receiving DLE:

8.13.4.1 Actions required of all DLEs

The DLE shall use the received PN DLPDU in the DLE-initialization procedures of 12.1.2.1 and 12.1.3. If those procedures require the receiving DLE to reply to the received PN DLPDU, then the DLE

- a) shall assume a reply token;
- b) shall initiate transmission of a PR DLPDU within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations (as measured at the PN-receiving DLE);

- c) shall include a probe-response SPDU, as described in clause 11, in the user data field of that PR DLPDU, and shall ensure that any randomly chosen fields of that SPDU are chosen from approximately-uniform distributions in a manner that is statistically independent of similar choices made by other DLEs.

8.13.4.2 Additional actions required of a Link-Master class DLE

None.

8.13.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.13.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.13.4.4 Additional actions required of the current LAS DLE

If the DLE did not just send the DLPDU, then the DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

8.14 Probe Response (PR) DLPDU

A PROBE RESPONSE (PR) DLPDU is sent to the LAS in response to the immediately-prior PROBE NODE DL-ADDRESS (PN) DLPDU, to convey probe-response information to the LAS.

8.14.1 Structure of the PR DLPDU

Table 25 – Structure of PR DLPDUs

Frame control	User data
0010 0111	SPDU

8.14.1.1 Frame control field

The frame control field shall specify the DLPDU's function.

8.14.1.2 Address field

The address field shall be null.

8.14.1.3 Parameters field

The parameters field shall be null.

8.14.1.4 User data field

The user data field shall consist of a higher-level probe-response SPDU.

NOTE This small limit on the size of the probe-response SPDU facilitates probing of NODE DL-addresses during intervals within the link schedule which would otherwise be unusable.

8.14.2 Content of the PR DLPDU

The frame control field shall be encoded as specified in Table 25; the implicit value for the final-token-use subfield shall be FINAL.

The address field shall be null. The generic LAS NODE DL-address of 04 is implicitly addressed as the DLPDU's destination, and the destination address from the immediately prior PN DLPDU sent on the local link is implicitly the PR DLPDU's source address.

The parameters field shall be null.

The user data field shall convey a higher-level probe-response SPDU whose maximum size is limited as specified in 8.14.1.4.

8.14.3 Sending the PR DLPDU

A PR DLPDU shall result from receipt of a PN DLPDU. The PR DLPDU shall be sent as specified in clause 11 and 8.13.4.

8.14.4 Receiving the PR DLPDU

A received PR DLPDU shall be treated as follows by the receiving DLE:

8.14.4.1 Actions required of all DLEs

None.

8.14.4.2 Additional actions required of a Link-Master class DLE

None.

8.14.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.14.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.14.4.4 Additional actions required of the current LAS DLE

If the DLE had sent a PN DLPDU as the previous DLPDU on the link, then the DLE

- a) shall report any higher-level SPDU conveyed by the received DLPDU, as if it had been received as DLS-user data contained in a DT DLPDU addressed to the LAS's NODE DL-address and sent from the NODE DL-address which was contained in the last-sent PN DLPDU (see 8.13.3), and shall process the SPDU as specified in clause 11;
- b) shall assume that the scheduler token is again dominant on the local link, resume active operation as the LAS, and initiate transmission
 - 1) within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the end of reception of the PR DLPDU, as measured at the receiving DLE,
 - 2) but not before the end of the observation interval following the PN DLPDU, as specified in 8.13.3.

If the DLE had sent a PN DLPDU as a previous DLPDU on the link, but another PR DLPDU was received after sending the PN DLPDU, and before reception of the current PR DLPDU, then the DLE shall ignore the current PR DLPDU and any contained SPDU.

If the DLE had not sent a PN DLPDU as the immediately prior non-PR DLPDU on the link, then the DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

8.15 Pass Token (PT) DLPDU

A PASS TOKEN (PT) DLPDU is used to pass a delegated token from the DLE functioning as an LAS to a DLE on the local link. By doing this repeatedly, the LAS DLE provides a delegated token which "circulates" successively, usually in NODE DL-address order, to all active DLEs on the local link which are included in the link's token-circulation list, $V(TCL)$ (see 6.7.5.3).

NOTE This usage also supports migration from previously existing national standards.

This DLPDU provides the receiving DLE with the right to initiate DL-transactions for a period of time specified in the delegating DLPDU.

8.15.1 Structure of the PT DLPDU

Table 26 – Structure of PT DLPDUs

Frame control	Destination NODE-address	Parameters
0011 0FPP	N	DD-p

8.15.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the minimum required priority of token use;
- c) the phase (initial or continuation) of the token delegation;
- d) the length, number and type of DLPDU addresses.

8.15.1.2 Address field

The address field shall consist of an explicit destination NODE DL-address.

8.15.1.3 Parameters field

The parameters field shall be structured and encoded as specified in 9.10.

8.15.1.4 User data field

The user data field shall be null.

8.15.2 Content of the PT DLPDU

The frame control field shall be encoded as specified in Table 26. The value of the priority subfield determines the minimum permitted priority of DLPDU which may be transmitted by the receiving DLE. The value of the final-use subfield shall be interpreted generally as specified in 7.2.1.2c), and specifically as:

RESTART — this is the initial token delegation within the current cycle of “circulating the token”, and so any repetitively scheduled transactions should be restarted;

CONTINUE — this is a subsequent (that is, secondary) token delegation within the current cycle of “circulating the token”, and so the previously initiated sequence of queued transactions should be continued.

The address field shall specify the NODE DL-address to which the token is being delegated.

The Delegation-Duration-parameter (DD-parameter) field shall convey, as specified in 9.10, the duration for which the token is being delegated to service Q(US), measured in octet-durations. Its permitted values are 0 to 65 000, measured in units of the transmission-duration of one octet (that is, in octet-durations).

NOTE The lower bound of zero is required to permit probing of node DL-addresses for DLEs which are included in the local link's live-list, V(LL), but not in the local link's token circulation list, V(TCL). The minimum practical value for use of this duration is about 16.

A duration of zero shall be used only when the addressed DLE's NODE DL-address is in the live list, V(LL), but not in the list of DLEs, V(TCL), to be polled with PT DLPDUs. In this case the addressed DLE indicates its continued presence by responding with an RT or RI DLPDU.

The user data field shall be null.

8.15.3 Sending the PT DLPDU

The LAS DLE sends a PT DLPDU to "circulate the token", usually in NODE DL-address order, through all of the active DLEs on the local link.

8.15.3.1 Determination of the PT DLPDU fields and related "token-rotation" parameters

- a) A PT DLPDU may be sent on the link
 - when the sending DLE (the LAS) holds a scheduler token which is the dominant token on the local link;
 - when the number of octet-durations of link capacity remaining until the next scheduled activity permits sending the PT DLPDU, plus the minimal required use by the receiving DLE, plus return of the token in an RI DLPDU or recovery of the token if required;
 - when the sending DLE does not need to send some other DLPDU.
- b) The overhead in delegating the token, which is not included in the delegated link capacity specified in the DD-parameters of the PT DLPDU, but which is factored into the determination of whether to send the PT DLPDU, is computed as the sum of
 - 1) the link capacity required to send the PT DLPDU;
 - 2) $V(IRR) \times V(ST)$ octet-durations for token acceptance and commencement of initial use;
 - 3) the link capacity required for the delegated-token-holder to send an RI DLPDU terminating token usage;
 - 4) the larger of
 - i) the minimum inter-DLPDU delay, $V(MID)$ octet-durations (see 6.7.1.12), which follows the RI DLPDU,
 - ii) any implementation-dependent period, measured in units of link capacity, required to recover the token upon expiration of the link capacity actually delegated.
- c) The maximum number of octet-durations of link capacity which can be delegated is computed as the smaller of
 - 1) 65 000 octet-durations,
 - 2) the number of octet-durations of link capacity remaining until the next scheduled activity minus the overhead in delegating the token specified in b), with the result reduced to account for possible differences in the two DLEs' data rates (that is, multiplied by the minimum possible ratio between the LAS DLE's and receiving DLE's Ph-data rates, as reported in 6.4.1.1).
- d) The minimum number of octet-durations of link capacity which a DLE requires shall be computed as follows:
 - 1) zero, when the DLE's NODE DL-address is not a member of the set V(TCL), indicating that the DLE does not require use of the token;
 - 2) $V(DMDT)$ (see 6.7.5.7), when 1) does not apply, and when
 - the PT DLPDU's final-use subfield specifies RESTART, or
 - the last token sent by a PT DLPDU to the same DL-address was not returned

by an RI DLPDU; or

- 3) the value from the DD-parameters of the RI DLPDU which returned the last token sent by a PT DLPDU to the same DL-address, when 1) and 2) do not apply.
- e) The maximum number of octet-durations of link capacity which may be delegated to a DLE shall be computed as follows:
- 1) zero, when the DLE's NODE DL-address is not a member of the set V(TCL), indicating that the DLE does not require use of the token;
 - 2) the member of the array V(MTHA) corresponding to the DLE's NODE DL-address, when 1) does not apply, and when the PT DLPDU's final-use subfield specifies RESTART;
 - 3) the member of the array V(RTHA) corresponding to the DLE's NODE DL-address, when 1) and 2) do not apply.
- NOTE The link capacity actually used by token-returning RI and RT DLPDUs is not included in this computation, since that link capacity is allocated to the overhead of the token-passing process.
- f) The token shall be delegated in ascending order of NODE DL-addresses to those DLE NODE DL-addresses represented in V(LL), beginning with the lowest-numbered such NODE DL-address. When a DLE has just assumed the scheduler token (by sending a CL DLPDU or receiving a TL DLPDU) and become the LAS, it shall commence token-passing with this lowest-numbered NODE DL-address, and shall restart token-passing with the lowest-numbered DLE after all other DLE's represented in the set V(LL) have terminated their token usage
- by specifying a final use of FINAL, or
 - by being unable to use the token without exceeding the maximum number of octet-durations of link capacity determined in e).

A DLE can, but need not, be temporarily bypassed in this ascending order while its requirements, as computed in d), exceed the link capacity momentarily available for token delegation, as computed in c). Any such bypassed DLE shall be delegated its remaining allocated link capacity before a PT DLPDU whose final-use field specifies RESTART is again delegated to that DLE.

NOTE This bypassing is an implementation choice, since its existence merely improves the momentary efficiency of the link and does not affect interoperability.

It is a protocol error to specify any DL-address other than a valid NODE DL-address, or to explicitly address a DLE which is known to be not active on the local link.

- g) The actual period of token delegation shall be computed to be the lesser of the values computed in c) and e), and the delegation shall occur only if that period is at least as great as the value computed in d).
- h) The initial PT DLPDU sent to each DLE on the link during one cycle of "circulating the token" shall specify RESTART in its final-use field; all subsequent PT DLPDUs sent to the same DLE during the same cycle, if any, shall specify CONTINUE in that final-use field. When RESTART is specified, the DLE also shall initialize the relevant member of V(RTHA) to the value of the corresponding member of V(MTHA).
- j) The actual period of token rotation, V(ATRT), shall be measured by the LAS DLE as the interval between successive occurrences of the LAS DLE's sending the PT DLPDU, with a token-use subfield specifying RESTART (see 8.15.2), to the lowest-numbered NODE DL-address represented in V(LL).
- k) The DLL priority specified in the PT DLPDU shall be determined before each cycle of "circulating the token" as follows:
 - 1) After the LAS DLE has just assumed the scheduler token (by sending a CL DLPDU or receiving a TL DLPDU), the priority shall be NORMAL.
 - 2) If 1) does not apply, and the just-computed V(ATRT) was greater than V(TTTRT), then
 - DL-management shall be notified of the event;

— the token priority shall be increased to the next higher level of urgency, if possible.

- 3) If 1) and 2) do not apply, so that the just-computed $V(ATRT)$ was less than or equal to $V(TTRT)$, then the token priority shall be decreased to the next lower level of urgency, if possible.

NOTE The inclusion of hysteresis or “learning” in this priority adjustment is an area of future study.

Once determined, the same DLL priority shall be included in all PT DLPDUs transmitted during that cycle of “circulating the token”.

8.15.3.2 Sending the PT DLPDU and monitoring the DLE to which the token is delegated

When sending the PT DLPDU, the LAS DLE shall record the destination address from the DLPDU as the delegation-address, $V(DTA)$ (see 6.7.5.1), for potential DL-management use in case the token is lost and not returned, and for potential association with any received RI or RT DLPDU.

After sending a PT DLPDU, the LAS DLE shall monitor the local link for a period of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations, waiting for a response:

- a) If a Ph-Data indication (see 6.4.4) reporting DATA is received, then the DLE shall proceed as in c).
- b) If a) does not apply, and the monitoring period of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations expires, then
 - 1) if the bus is not active at that moment (that is, the last-received Ph-Data indication reported END-OF-ACTIVITY), then the DLE shall proceed as in d);
 - 2) if 1) does not apply, implying that the link is still active at that moment (that is, the last-received Ph-Data indication reported START-OF-ACTIVITY), then

A) for implementations that were not demonstrable on or before 31 December 1995, the DLE shall monitor the local link for a period of one additional slot-time, $V(ST)$ octet-durations, waiting for a Ph-Data indication:

- i) if a Ph-Data indication reporting DATA is received, then the DLE shall proceed as in c).

- ii) if a Ph-Data indication reporting END-OF-ACTIVITY is received, and i) does not apply, then the DLE shall stop further monitoring and proceed as in d).

NOTE Without provision ii), ill-timed low-level noise can cause the LAS DLE to not retry the token pass, resulting in omitting a DLE from the current cycle of circulating the token. Provision ii) enables the LAS DLE to distinguish probable noise events from actual transmissions.

- iii) if neither i) nor ii) applies, and the monitoring period expires before a Ph-Data indication is received, then the DLE shall proceed as in c).

B) Implementations that were demonstrable on or before 31 December 1995 alternatively may just proceed as in c).

- c) Until the delegated token is returned, the LAS DLE shall continually monitor local link activity, and if it observes
 - 1) a period of token-recovery-delay slot-times, $P(TRD) \times V(ST)$ octet-durations, of continuous non-activity; or
 - 2) for implementations that were not demonstrable on or before 31 December 1995, a period of token-recovery-delay slot-times, $P(TRD) \times V(ST)$ octet-durations, during which any continuous period of activity (that is, between a Ph-Data indication reporting

START-OF-ACTIVITY and the next following Ph-Data indication reporting END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) both

- i) did not result in a Ph-Data indication (see 6.4.4) reporting DATA, and
- ii) had a duration of less than one to two $V(ST)$ octet-durations

NOTE The vagueness in the extent of permissible activity duration permits some economy of implementation of the measurement.

then it shall

- assume that the scheduler token is dominant on the local link;
 - inform local DL-management of the event;
 - commence transmission on the link.
- d) If the monitoring period of immediate-response-recovery-delay slot-times, $V(IRR) \times V(ST)$ octet-durations expires and a) does not apply; or, for implementations that were not demonstrable on or before 31 December 1995, the additional monitoring period of one slot-time, $V(ST)$ octet-durations, expires and c) does not apply; then the LAS DLE shall retry sending the PT DLPDU and monitoring the local link, up to the number of times specified as the maximum-retry-count, $V(MRC)$ (see 6.7.1.5), of the local link, each time using as the value for its duration parameter the smaller of
- 1) the value originally sent; or
 - 2) the number of octet-durations of link capacity remaining until the next scheduled activity minus the overhead in delegating the token specified in 8.15.3.1b), with the result reduced to account for possible differences in the two DLEs' data rates as specified in 8.15.3.1c)2),

provided that the retry and attempted re-delegation shall occur only if that period is at least as great as the value computed in 8.15.3.1d).

This retry shall commence within token-recovery-delay slot-times, $P(TRD) \times V(ST)$ octet-durations, of the beginning of the current period of link non-activity.

If all attempted retries are unsuccessful, then the LAS DLE shall

- inform local DL-management of the event;
- NOTE DLEs with node DL-addresses in the set $\{ F816..FF16 \}$ are expected to terminate operation by dropping out of the token circulation process. Thus, DL-management should not treat such occurrences as evidence of DLE or local-link malfunction.
- start the next transmission within token-recovery-delay slot-times, $P(TRD) \times V(ST)$ octet-durations, of the beginning of the current period of link non-activity.

After three such event reports for any specific NODE DL-address within three cycles of "circulating the token", DL-management shall remove that NODE DL-address from the LAS's local-link token-circulation-list, $V(TCL)$, and local-link live-list, $V(LL)$.

8.15.4 Receiving the PT DLPDU

Receipt of a PT DLPDU by the addressed DLE enables the DLE to emulate a "circulated token" as found in previous token-passing bus standards, such as ISO/IEC 8802-4 and IEC 60955, by enabling transmission of DLPDUs at the specified and higher priority either

- until all pending transmissions have occurred, or
- until the portion of the local-link's transmission capacity which was allocated by the DD-parameters of the received PT DLPDU requires return of the delegated token.

A received PT DLPDU shall be treated by the receiving DLE as follows:

8.15.4.1 Actions required of all DLEs

If the destination DL-address specified by the DLPDU designates the NODE DL-address of the receiving DLE, then if the receiving DLE, due to its construction, does not use the delegated token, then the receiving DLE shall initiate transmission of an RT DLPDU within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the end of reception of the PT DLPDU, as measured at the receiving DLE; else the receiving DLE shall

- a) copy the value of the duration parameter from the DLPDU into its local remaining-duration down-counter, C(RD);
- b) assume the delegated token;
- c) initiate transmission within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the end of reception of the PT DLPDU, as measured at the receiving DLE;
- d) repeatedly employ the following selection criteria until the delegated token is returned to the LAS, each time initiating transmission within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the end of the prior transmission (unless that was a CA, CD, CT, ED or RQ transmission), as measured at the PT-receiving DLE.

The counter C(RD) shall decrement and be managed as specified in 6.7.1.11.

Transactions of the priority specified in the delegating PT DLPDU, or of higher priority

- 1) shall be initiated successively;
- 2) beginning as follows:
 - starting from the first transaction and including all REPETITIVE sequences if the token-use subfield of the received PT DLPDU specified RESTART, or
 - continuing from the previously initiated transaction and thus excluding all REPETITIVE sequences which have been executed since the last PT DLPDU specifying RESTART was received if the token-use subfield of the received PT DLPDU specified CONTINUE;
- 3) subject to the constraint that the DLE shall limit its token use to a sequence of transactions which is guaranteed to complete within the duration specified in the DD-parameter of the received PT DLPDU.

8.15.4.1.1 Selection of the next transaction to be executed

If the DLE has a need to send a CT DLPDU, the DLE shall transmit a CT DLPDU and await its immediate reply. Otherwise, the receiving DLE repeatedly shall select the next member, if any, of the DLE's unscheduled-service queue, Q(US), whose priority is not less than the priority of the received PT, and shall apply all of the remainder of this subclause, as appropriate:

- a) If that next member is a reference to a DL-address user-request queue, Q_A(UR), then
 - 1) if the member is a reference which signals the need to compel a DLSDU transmission from the remote correspondent of a local peer or subscriber DLCEP (see 6.7.1.17a)1)), then the DLE shall
 - i) transmit a CD DLPDU to the remote DLC endpoint, with SD-parameters conveying the state of the local DLCEP to the remote DLCEP, as specified in 10.2.2.1 and 10.2.2.4;

ii) await an immediate reply;

NOTE The maximum DLPDU size of that immediate reply can be determined solely on the basis of the negotiated DLCEP parameters.

iii) delete the referencing member from the DLE's unscheduled-service queue,

Q(US).

- 2) if the member is a reference which signals the need to compel an instance of unitdata exchange with a specified DLSAP-address at a specified service priority (see 6.7.1.17a2)), then the DLE shall

i) transmit an ED or CD DLPDU to that remote DLSAP-address, at the specified service priority, as appropriate, as specified in 10.2.2.1 and 10.2.2.4;

ii) await an immediate reply;

NOTE The maximum DLPDU size of that immediate reply can be determined solely on the basis of the negotiated DLCEP parameters.

iii) delete the referencing member from the DLE's unscheduled-service queue, Q(US).

- 3) if the member is a reference which signals the need to transmit the contents of a sending buffer which is bound to a local DLCEP (see 6.7.1.17a3)), then the DLE shall

i) transmit the DLPDU appropriate to the state of the DLCEP;

ii) if the resultant transmitted DLPDU completes the transmission of a DLSDU (so that no segments of that DLSDU remain to be transmitted), then the DLE shall

— notify the DLS-user of the request's completion, if appropriate;

— delete the referencing member from the DLE's unscheduled-service queue, Q(US).

- 4) if the member is a reference which signals the need to send a DLSDU from the sending queue which is bound to a local DLCEP or "DLSAP-address" (see 6.7.1.17a4)), or to send a DLPDU from the DLE's NODE DL-address, then the DLE shall select a member of the referenced $Q_A(UR)$, of the specified or higher priority, which has been released for transmission and which requires full or partial transmission or retransmission, as follows:

— either the first member of the second partition of $Q_A(UR)$, or

— the first member of the first partition of $Q_A(UR)$ which has one or more segments marked for retransmission.

NOTE Due to potential concurrency in real implementations, the set of segments so marked may not reflect status information from DLPDUs recently received at the DLCEP.

If such a member exists, then

i) the DLE shall transmit the DLPDU with a segment of the DLSDU appropriate to the transmission segment status, $V_{C,K}$ (SS) (see 6.7.4.8), if applicable, of that member of that $Q_A(UR)$;

ii) if the resultant transmitted DLPDU completes the transmission of a DLSDU (so that no segments of that DLSDU remain to be transmitted), then the DLE shall

— advance that member of $Q_A(UR)$ to the already sent partition, or remove it, as appropriate to the type of member;

— notify the DLS-user of the request's completion, if appropriate;

— delete the referencing member from the DLE's unscheduled-service queue, Q(US).

If no such member of the referenced DL-address request queue, $Q_A(UR)$, exists, then

iii) if the DL-address is a DLCEP-address of a local peer or publisher DLCEP, then the DLE shall

A) transmit a DT or RC DLPDU, as appropriate for the state of the DLCEP, with a null user data field, to convey the state of the DLCEP to the correspondent peer or subscriber DLCEP(s);

B) delete the referencing member from the DLE's unscheduled-service queue, $Q(US)$.

iv) otherwise, when iii) does not apply, then the DL-address is not a DLCEP-address and the DLE shall repeat the selection process.

b) If that next member is a reference to a REPETITIVE sequence (see 10.4.3.1d)), then

1) if there are one or more active members of the sequence (that is, not excluded by the most recent DL-SUBSET-SEQUENCE request, if any, specifying the sequence);

2) if one or more of those active members has not been executed since the DLE last received a PT DLPDU specifying RESTART,

then the DLE shall

i) insert a reference to the next active member of that sequence on $Q(US)$ immediately before the reference to the sequence (with the result that the newly-inserted reference will be deleted at completion of the just-scheduled action);

ii) continue as in the preceding case a), as determined by the type of the just-inserted member of the sequence.

NOTE At completion of the just-scheduled action, the next member of $Q(US)$ will be the reference to the REPETITIVE sequence, and the above procedure will continue until all active members of the REPETITIVE sequence have been processed.

3) otherwise, when neither 1) nor 2) applies, then the DLE shall repeat the selection process.

c) If that next member is a reference to a sequence (see 10.4.3.1a)) which is not a REPETITIVE sequence, and if the sequence consists of a single element, then the DLE shall

1) replace the reference to the sequence on $Q(US)$ with the reference to the single element, so that the same member of $Q(US)$ now is the element reference;

2) issue a DL-SCHEDULE-SEQUENCE confirm primitive if it has not already been confirmed;

3) delete the sequence;

4) continue as in the preceding case a), as determined by the type of the sole element of the selected and just-deleted sequence.

All other cases are erroneous and are not permitted.

8.15.4.1.2 Additional considerations

The following considerations also apply:

i) If there are no additional members, of the priority specified in the received PT DLPDU, or of any higher priority, in the DLE's unscheduled-service queue, $Q(US)$, and if immediate retry of the currently-selected transaction is not possible, then the final-token-use subfield in that non-RT DLPDU can be set to the value FINAL before transmission, thereby returning the delegated token to the LAS DLE at the end of the current transaction.

ii) If there are no members, of the priority specified in the received PT DLPDU, or of any higher priority, in the DLE's unscheduled-service queue, Q(US), and the delegated token was not returned by setting a final-token-use subfield to the value FINAL in the last DLPDU transmitted by the DLE, then the DLE shall return the delegated token by transmitting an RT DLPDU.

iii) If at any time

- A) the remaining duration of token usage — as indicated by the remaining-duration down-counter, C(RD) — is inadequate to permit any further use by the receiving DLE;
- B) the DLE's unscheduled-service queue, Q(US), contains additional members whose priority is equal to or higher than the priority specified in the last-received PT DLPDU,

then the receiving DLE shall return the delegated token to the LAS by sending an RI DLPDU specifying the minimum delegation interval for token use required when the token is next returned.

NOTE An RI DLPDU, which is sent to terminate use of a token delegated by a PT DLPDU, indicates to the receiving LAS that additional service is needed and that another PT DLPDU, with a final-token-use subfield specifying CONTINUE, should be sent to the DLE during the current cycle of "circulating the token" if the total link capacity allocated to the DLE by the DLE'th member of V(MTHA) (see 6.7.5.10) so permits.

iv) If none of i) to iii) applies, then the DLE shall transmit the non-RT non-RI DLPDU with a final-token-use subfield value of NOT-FINAL.

8.15.4.2 Additional actions required of a Link-Master class DLE

The DLE may, but is not required to, record the destination address from the DLPDU as the delegation-address, V(DTA) (see 6.7.5.1), for potential use in restarting the circulated token should the current LAS fail and this LM DLE become the new LAS. The DLE may also use that recorded V(DTA) to detect additions to, but not deletions from, the set of DLEs receiving the circulated token, and update its copy of the token circulation list, V(TCL) (see 6.7.5.3) accordingly.

NOTE These actions optimize recovery of the token circulation process in the event of failure of the DLE currently functioning as LAS.

8.15.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.15.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.15.4.4 Additional actions required of the current LAS DLE

If the DLE did not just send the DLPDU, then the DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

8.16 Execute Sequence (ES) DLPDU

An EXECUTE SEQUENCE (ES) DLPDU is used to pass a delegated token from the DLE functioning as an LAS to a DLE on the local link which has previously requested or scheduled local link capacity, or for which such a schedule has been configured by DL-management.

NOTE This usage supports migration from previously-existing national standards.

This DLPDU provides the receiving DLE with the right to initiate DL-transactions for a period of time, specified in the delegating DLPDU; the destination DL-address of that DLPDU is used to determine which set of transactions to initiate.

8.16.1 Structure of the ES DLPDU

Table 27 – Structure of ES DLPDUs

format	Frame control	Destination DLSEP-address	Parameters
1L	1000 1F00	HL.N.S	DD-p
1S	1000 0F00	N.S	DD-p

8.16.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the phase (initial or continuation) of the sequence execution;
- c) the length, number and type of DLPDU addresses.

8.16.1.2 Address field

The address field shall consist of an explicit destination DL-address.

8.16.1.3 Parameters field

The parameter field shall be structured and encoded as specified in 9.10.

8.16.1.4 User data field

The user data field shall be null.

8.16.2 Content of the ES DLPDU

The frame control field shall be encoded as specified in Table 27. The value of the final-use subfield shall be interpreted generally as specified in 7.2.1.2c), and specifically as:

RESTART — this is the initial token delegation within the current cycle of scheduled sequence execution, and so the indicated sequence and any repetitively scheduled transactions should be restarted;

CONTINUE — this is a subsequent (that is, secondary) token delegation within the current cycle of scheduled sequence execution, and so the previously-initiated sequence of scheduled transactions should be continued at the point where execution was last suspended and the ES DLPDU was re-requested (see 8.18.3).

The address field shall specify the DLSEP-address to which the token is being delegated. For format 1L, the address shall be LONG; for format 1S it shall be SHORT.

The Delegation-Duration-parameter (DD-parameter) field shall convey, as specified in 9.10 the duration for which the token is being delegated to execute the addressed scheduled sequence, measured in units of the transmission-period of one octet on the local link (that is, eight times the local link's nominal bit period). It has the range 0 to 65 000 octet-durations.

The user data field shall be null.

8.16.3 Sending the ES DLPDU

The LAS DLE sends an ES DLPDU when so instructed by the local link's current schedule, or in response to requests to extend the time allocated for an earlier scheduled (and executed) token delegation.

8.16.3.1 Sending the ES DLPDU as instructed by the local link's current schedule

An ES DLPDU may be sent on the link when the sending DLE (the LAS) holds a scheduler token which is the dominant token on the local link, and the link's schedule requires transmission of the ES DLPDU. The final-use subfield of such an ES DLPDU shall be dictated by the schedule.

8.16.3.2 Sending the ES DLPDU in response to earlier requests for extensions of scheduled token delegation

An ES DLPDU shall only be sent as a response to an RI DLPDU as specified in clause 11. If the LAS DLE sends an ES DLPDU which is not dictated by the link's current schedule, then the following procedures shall apply:

- a) An ES DLPDU may be sent on the link when the sending DLE (the LAS) holds a scheduler token which is the dominant token on the local link, and when the number of octet-durations of link capacity remaining until the next scheduled activity permits sending the ES DLPDU, plus the minimal required use by the receiving DLE, plus return of the token in an RI DLPDU or recovery of the token if required. The final-use subfield of such an ES DLPDU shall specify CONTINUE.
- b) The overhead in delegating the token, which is not included in the delegated link capacity specified in the DD-parameters of the ES DLPDU, but which is factored into the determination of whether to send the ES DLPDU, is computed as the sum of
 - 1) the link capacity required to send the ES DLPDU;
 - 2) $V(\text{IRRD}) \times V(\text{ST})$ octet-durations for token acceptance and commencement of initial use;
 - 3) the link capacity required for the delegated-token-holder to send an RI DLPDU terminating token usage;
 - 4) the larger of
 - i) the minimum inter-DLPDU delay, $V(\text{MID})$ octet-durations (see 6.7.1.12), which follows the RI DLPDU; or
 - ii) any implementation-dependent period, measured in units of link capacity, required to recover the token upon expiration of the link capacity actually delegated.
- c) The maximum number of octet-durations of link capacity which can be delegated is computed as the smaller of
 - 1) 65 000 octet-durations; or
 - 2) the number of octet-durations of link capacity remaining until the next scheduled activity minus the overhead in delegating the token specified in b), with the result reduced to account for possible differences in the two DLEs' data rates (that is, multiplied by the minimum possible ratio between the LAS DLE's and receiving DLE's Ph-data rates, as reported in 6.4.1.1).
- d) The actual period of token delegation shall be the value returned in the DD-parameters of the RI DLPDU which terminated the prior interval of delegated token usage at the same DL-address.

8.16.3.3 Monitoring the DLE to which the token is delegated and additional considerations

It is a protocol error to specify any DL-address other than a valid DLSEP-address, or to explicitly or implicitly address a DLE not active on the local link. However, an LAS which receives a DLSEP-address from another DLE in a scheduling-related SPDU may presume that such an address is valid.

When sending the ES DLPDU, the LAS DLE shall record the destination address from the DLPDU as the delegation-address, V(DTA) (see 6.7.5.1), for potential DL-management use in case the token is lost and not returned, and for potential association with any received RI or RT DLPDU.

After sending an ES DLPDU, the LAS DLE shall monitor the local link for a period of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations, waiting for a response:

- a) If a Ph-Data indication (see 6.4.4) reporting DATA is received, then the DLE shall proceed as in c).
- b) If a) does not apply, and the monitoring period of immediate-response-recovery-delay slot-times, $V(IRRDR) \times V(ST)$ octet-durations expires, then
 - 1) if the bus is not active at that moment (that is, the last-received Ph-Data indication reported END-OF-ACTIVITY), then the DLE shall proceed as in d).
 - 2) if 1) does not apply, implying that the link is still active at that moment (that is, the last-received Ph-Data indication reported START-OF-ACTIVITY), then

A) for implementations that were not demonstrable on or before 31 December 1995, the DLE shall monitor the local link for a period of one additional slot-time, $V(ST)$ octet-durations, waiting for a Ph-Data indication:

- i) if a Ph-Data indication reporting DATA is received, then the DLE shall proceed as in c).
- ii) if a Ph-Data indication reporting END-OF-ACTIVITY is received, and i) does not apply, then the DLE shall stop further monitoring and proceed as in d);

NOTE Without provision ii), ill-timed low-level noise can cause the LAS DLE not to retry the token pass, resulting in omitting a DLE from the current cycle of schedule execution. Provision ii) enables the LAS DLE to distinguish probable noise events from actual transmissions.

iii) if neither i) nor ii) applies, and the monitoring period expires before a Ph-Data indication is received, then the DLE shall proceed as in c).

B) Implementations that were demonstrable on or before 31 December 1995 alternatively may just proceed as in c).

- c) Until the delegated token is returned, the LAS DLE shall continually monitor local link activity, and if it observes
 - 1) a period of token-recovery-delay slot-times, $P(TRD) \times V(ST)$ octet-durations, of continuous non-activity; or
 - 2) for implementations that were not demonstrable on or before 31 December 1995, a period of token-recovery-delay slot-times, $P(TRD) \times V(ST)$ octet-durations, during which any continuous period of activity (that is, between a Ph-Data indication reporting START-OF-ACTIVITY and the next following Ph-Data indication reporting END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) both

- i) did not result in a Ph-Data indication (see 6.4.4) reporting DATA, and
- ii) had a duration of less than one to two $V(ST)$ octet-durations,

NOTE The vagueness in the extent of permissible activity duration permits some economy of implementation of the measurement.

then it shall

- assume that the scheduler token is dominant on the local link;
 - inform local DL-management of the event;
 - commence transmission on the link.
- d) If the monitoring period of immediate-response-recovery-delay slot-times, $V(\text{IRRD}) \times V(\text{ST})$ octet-durations expires and a) does not apply; or, for implementations that were not demonstrable on or before 31 December 1995, the additional monitoring period of one slot-time, $V(\text{ST})$ octet-durations, expires and c) does not apply; then the LAS DLE shall retry sending the ES DLPDU and monitoring the local link, up to the number of times specified as the maximum-retry-count, $V(\text{MRC})$ (see 6.7.1.5), of the local link, each time using as the value for its duration parameter the value originally-sent, and the re-delegation shall occur only if that period is at least as great as the value specified in 8.16.3.1 or 8.16.3.2d).

This retry shall commence within token-recovery-delay slot-times, $P(\text{TRD}) \times V(\text{ST})$ octet-durations, of the beginning of the current period of link non-activity.

If all attempted retries are unsuccessful, then the LAS DLE shall

- inform local DL-management of the event;
- start the next transmission within token-recovery-delay slot-times, $P(\text{TRD}) \times V(\text{ST})$ octet-durations, of the beginning of the current period of link non-activity.

8.16.4 Receiving the ES DLPDU

Receipt of an ES DLPDU by the addressed DLE enables the DLE to execute all or part of a previously scheduled sequence whose initiation is synchronized by the LAS, either

- until the sequence has been completed, or
- until the portion of the local-link's transmission capacity which was allocated by the DD-parameters of the received ES DLPDU requires return of the delegated token.

NOTE This latter situation can occur

- A) on sequences which permit interruption;
- B) on sequences which permit dynamic appending of DL-COMPEL-SERVICE requests;
- C) when retransmission of the initiating ES DLPDU, or possibly of another DLPDU within the defined sequence, caused link capacity beyond that allocated to be needed.

A received ES DLPDU shall be treated by the receiving DLE as follows.

8.16.4.1 Actions required of all DLEs

8.16.4.1.1 When addressed to a DL-address of the DLE other than a DLSEP-address

If the destination DL-address specified by the DLPDU designates a DL-address of the receiving DLE other than a DLSEP-address, then the DLE shall

- a) inform local DL-management of the event;
- b) initiate transmission within a period of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, as measured at the receiving DLE, of an RT DLPDU.

NOTE The receiving DLE is rejecting the received ES DLPDU.

8.16.4.1.2 When addressed to a DLSEP-address of the DLE

If the destination DL-address specified by the DLPDU designates a DLSEP-address of the receiving DLE, then the receiving DLE shall

- a) copy the value of the duration parameter from the DLPDU into its local remaining-duration down-counter, C(RD);
- b) assume the delegated token;
- c) initiate transmission within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the end of reception of the ES DLPDU, as measured at the receiving DLE;
- d) repeatedly employ the following selection criteria until the delegated token is returned to the LAS, each time initiating transmission within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, of the end of the prior transmission (unless that was a CA, CD, CT or ED transmission), as measured at the ES-receiving DLE.

The counter C(RD) shall decrement and be managed as specified in 6.7.1.11.

Transactions in the sequence associated with the designated DLSEP-address

- 1) shall be initiated successively;
- 2) beginning as follows:
 - starting with the initial transaction of the sequence if the token-use subfield of the received ES DLPDU specified RESTART, or
 - continuing from the point at which the sequence was previously suspended if the token-use subfield of the received ES DLPDU specified CONTINUE;
- 3) subject to the constraint that the worst-case link capacity required to complete the transaction be less than the then-current value of the remaining-duration down-counter, C(RD).

8.16.4.1.2.1 Selection of the next transaction to be executed

The receiving DLE shall execute the transactions in the sequence as follows.

If that next member of the sequence is a reference to a DL-address user-request queue, $Q_A(UR)$, then

- a) if the member is a reference to the remote correspondent of a local peer or subscriber DLCEP (see 10.4.2.1b)1)i)), then the DLE shall
 - 1) transmit a CD DLPDU to the remote DLC endpoint, with SD-parameters conveying the state of the local DLCEP to the remote DLCEP, as specified in 10.2.2.1 and 10.2.2.4;
 - 2) await an immediate reply;

NOTE The maximum DLPDU size of that immediate reply can be determined solely on the basis of the negotiated DLCEP parameters.
 - 3) advance to the next element of the sequence which was addressed by the received ES DLPDU;
- b) if the member is a reference which indicates the need to perform a unitdata exchange with a specified DLSAP-address at a specified service priority (see 10.4.2.1c)1)i)), then the DLE shall
 - 1) transmit an ED or CD DLPDU to that remote DLSAP-address, at the specified service priority, as appropriate, as specified in 10.2.2.1 and 10.2.2.4;
 - 2) await an immediate reply;

NOTE The maximum DLPDU size of that immediate reply can be determined solely on the basis of the negotiated DLCEP parameters.

- 3) advance to the next element of the sequence which was addressed by the received ES DLPDU.

Otherwise, the DLE shall select the first member of the referenced $Q_A(UR)$ which has been released for transmission and which requires full or partial transmission or retransmission.

- c) If such a member exists, then

- 1) the DLE shall transmit the DLPDU appropriate to the state of that member of that $Q_A(UR)$;
- 2) if the resultant transmitted DLPDU completes the transmission of a DLSDU (so that no segments of that DLSDU remain to be transmitted), then the DLE shall

- advance that member of $Q_A(UR)$ to the already-sent partition, or remove it, as appropriate to the type of member;
- notify the DLS-user of the request's completion, if appropriate;
- advance to the next element of the sequence which was addressed by the received ES DLPDU.

- d) If no such member of the referenced DL-address request queue, $Q_A(UR)$, exists, then

- 1) if the DL-address is

- a DLCEP-address of a local peer or publisher DLCEP, or
- a publisher DLCEP-address of a local subscriber DLCEP,

and the sending binding of the DLCEP is to a buffer which is not empty, then

- i) the DLE shall transmit the DLPDU appropriate to the state of the DLCEP;
- ii) if the resultant transmitted DLPDU completes the transmission of a DLSDU (so that no segments of that DLSDU remain to be transmitted), then the DLE shall notify the DLS-user of the request's completion, if appropriate.
- 2) Otherwise, when i) does not apply, and the DL-address is a DLCEP-address of a local peer or publisher DLCEP, then the DLE shall transmit a DT or RC DLPDU, as appropriate for the state of the DLCEP, with a null user data field, to convey the state of the DLCEP to the correspondent peer or subscriber DLCEP(s).
- 3) Whether or not a DLPDU was transmitted, the DLE shall advance to the next member of the sequence which was addressed by the received ES DLPDU.

All other cases are erroneous and are not permitted.

8.16.4.1.2.2 Additional considerations

The following considerations also apply:

- i) If there are no additional members in the sequence, and if immediate retry of the currently-selected transaction is not possible, then the final-token-use subfield in that non-RT DLPDU can be set to the value FINAL before transmission, thereby returning the delegated token to the LAS DLE at the end of the current transaction.
- ii) If the sequence is completed, and the delegated token was not returned by setting a final-token-use subfield to the value FINAL in the last DLPDU transmitted by the DLE, then the DLE shall return the delegated token by transmitting an RT DLPDU.

iii) If at any time

- A) the remaining-duration down-counter, C(RD), is inadequate to permit any further use by the receiving DLE;
- B) the sequence contains additional members,

then the receiving DLE shall return the delegated token to the LAS by sending an RI DLPDU specifying the minimum delegation interval for token use required when the token is next returned.

NOTE An RI DLPDU, which is sent to terminate use of a token delegated by an ES DLPDU, indicates to the receiving LAS that sequence execution was incomplete and that another ES DLPDU, with a final-token-use subfield specifying CONTINUE, should be sent to the same DLSEP-address.

8.16.4.2 Additional actions required of a Link-Master class DLE

None.

8.16.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.16.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.16.4.4 Additional actions required of the current LAS DLE

If the DLE did not just send the DLPDU, then the DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

8.17 Return Token (RT) DLPDU

A RETURN TOKEN (RT) DLPDU is used to return the token delegated by a PT or ES DLPDU to the LAS and to indicate that no additional token delegation, to the DL-address of the last PT or ES DLPDU, is required.

NOTE The delegated token also may be returned by setting the final-token-use designator in any DLPDU to the value FINAL, and then ceasing use of the delegated token after transmitting that DLPDU.

8.17.1 Structure of the RT DLPDU

Table 28 – Structure of RT DLPDUs

Frame control
0011 0100

8.17.1.1 Frame control field

The frame control field shall specify the DLPDU's function.

8.17.1.2 Address field

The address field shall be null.

8.17.1.3 Parameters field

The parameters field shall be null.

8.17.1.4 User data field

The user data field shall be null.

8.17.2 Content of the RT DLPDU

The frame control field shall be encoded as specified in Table 28.

The address, parameters and user data fields shall be null.

8.17.3 Sending the RT DLPDU

An RT DLPDU results only from receipt of a PT or ES DLPDU. The RT DLPDU shall be sent as specified in 8.15.4 and 8.16.4.

8.17.4 Receiving the RT DLPDU

A received RT DLPDU shall be treated as follows by the receiving DLE:

8.17.4.1 Actions required of all DLEs

None.

8.17.4.2 Additional actions required of a Link-Master class DLE

None.

8.17.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.17.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.17.4.4 Additional actions required of the current LAS DLE

When the scheduler token is not dominant on the local link, then

- a) if the token being returned was delegated by a PT DLPDU, then the DLE shall set the value of the V(DTA)'th member of the remaining-token-holding-time-array, V(RTHA) (see 6.7.5.13) to zero;
- b) the DLE shall assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.18 Request Interval (RI) DLPDU

A REQUEST INTERVAL (RI) DLPDU is used to return a token delegated by a PT or ES DLPDU to the LAS and to indicate the minimal useful delegation interval required for the next delegation.

NOTE The delegated token also may be returned by setting the final-token-use designator in any DLPDU to the value FINAL, and then ceasing use of the delegated token after transmitting that DLPDU, or by sending an RT DLPDU.

8.18.1 Structure of the RI DLPDU

Table 29 – Structure of RI DLPDUs

Frame control	Parameters
0010 0000	DD-p

8.18.1.1 Frame control field

The frame control field shall specify the DLPDU's function.

8.18.1.2 Address field

The address field shall be null.

8.18.1.3 Parameters field

The parameter field shall be structured and encoded as specified in 9.10.

8.18.1.4 User data field

The user data field shall be null.

8.18.2 Content of the RI DLPDU

The frame control field shall be encoded as specified in Table 29. The address field shall be null.

The parameters field shall specify the minimum useful duration for the next delegation of the current token. The user data field shall be null.

8.18.3 Sending the RI DLPDU

An RI DLPDU results only from receipt of an ES or PT DLPDU. The RI DLPDU shall be sent as specified in 8.15.4 and 8.16.4.

8.18.4 Receiving the RI DLPDU

A received RI DLPDU shall be treated as follows by the receiving DLE:

8.18.4.1 Actions required of all DLEs

None.

8.18.4.2 Additional actions required of a Link-Master class DLE

None.

8.18.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.18.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.18.4.4 Additional actions required of the current LAS DLE

When the scheduler token is not dominant on the local link, then the DLE shall

- a) if the token being returned was delegated by a PT DLPDU, then the DLE shall set the value of the V(DTA)'th member of the remaining-token-holding-time-array, V(RTHA) (see 6.7.5.13), to account for the link capacity just used by the DLE;
- b) if the token being returned was delegated by an ES or PT DLPDU, then the DLE shall forward the DD-parameter field of this DLPDU, together with V(DTA), to the LAS DLE's upper-level functions (see clause 11) for attempted rescheduling;
- c) the DLE shall assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.19 Claim LAS (CL) DLPDU

A CLAIM LAS (CL) DLPDU is used by an LM DLE to initialize the local link, or to recover from a prolonged period of silence on the local link indicative of the failure of the prior LAS.

8.19.1 Structure of the CL DLPDU

Table 30 – Structure of CL DLPDUs

Frame control	Source NODE-address
0000 0001	N

8.19.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the length, number and type of DLPDU addresses.

8.19.1.2 Address field

The address field shall consist of an explicit source NODE DL-address.

8.19.1.3 Parameters field

The parameters field shall be null.

8.19.1.4 User data field

The user data field shall be null.

8.19.2 Content of the CL DLPDU

The frame control field shall be encoded as specified in Table 30. The address field shall be the NODE DL-address of the DLE which has detected the prolonged link inactivity, whose value is the sender's node-id, V(TN). The parameters and user data fields shall be null.

NOTE The primary purposes of the NODE DL-address in the CL DLPDU are

- a) to permit improved detection of collisions during the LAS claiming procedure;
- b) to identify the DLE which detected inactivity to observing DLEs, which may have utility in real open systems for identifying DLEs with particularly obstructive classes of PhE or DLE faults, sometimes characterized as "faulty transmitter" or "deaf receiver".

8.19.3 Sending the CL DLPDU

A CL DLPDU shall result from the detection in an LM DLE of a period of node-id slot-times, $V(TN) \times V(ST)$ octet-durations, of continuous link inactivity, implying failure of the previous LAS DLE, if any.

Upon such detection, the DLE shall immediately send one CL DLPDU, and shall again monitor the medium for node-id slot-times, $V(TN) \times V(ST)$ octet-durations, of continuous link inactivity after sending the first CL DLPDU.

If no activity is again heard (other than potentially a CL DLPDU with the NODE DL-address that was just sent), then the sending LM DLE shall immediately send a second identical CL DLPDU, after which the DLE shall choose a uniformly distributed random integer in the range zero to three, and shall monitor the medium for that many slot-times, $\text{random}(0..3) \times V(ST)$ octet-durations. The actual random choice shall be statistically independent of similar choices made by other DLEs.

NOTE This requirement for statistical independence minimizes the probability of repeatedly identical choices by identically constructed real devices.

If no activity is again heard (other than potentially the CL DLPDU with the NODE DL-address that was just sent), then the sending LM DLE shall activate its LAS functions, assume the scheduler token and commence active operation as the LAS.

8.19.4 Receiving the CL DLPDU

A received CL DLPDU shall be treated as follows by the receiving DLE:

8.19.4.1 Actions required of all DLEs

- a) The DLPDU shall be reported to the DLE's local DL-management.
- b) If the receiving DLE holds a token on the local link, then the receiving DLE shall drop the token at once and shall inform local DL-management of the event.

8.19.4.2 Additional actions required of a Link-Master class DLE

None

8.19.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.1.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.19.4.4 Additional actions required of the current LAS DLE

The DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

8.20 Transfer LAS (TL) DLPDU

A TRANSFER LAS (TL) DLPDU is used by the current LAS DLE to transfer the scheduler token and the role of LAS to another LM DLE on the local link. The TL DLPDU is sent only after having been requested by the addressed LM DLE, and may be rejected if the addressed DLE determines that its own copy of the local link's schedule is not current.

8.20.1 Structure of the TL DLPDU

Table 31 – Structure of TL DLPDUs

Frame control	Destination NODE-address	User data
0000 0110	N	o-SPDU

8.20.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is URGENT;
- c) the length, number and type of DLPDU addresses.

8.20.1.2 Address field

The address field shall consist of an explicit destination NODE DL-address.

8.20.1.3 Parameters field

The parameters field shall be null.

8.20.1.4 User data field

The user data field shall consist of an optional higher-level LAS-database-status SPDU of 64 octets or less.

8.20.2 Content of the TL DLPDU

The frame control field shall be encoded as specified in Table 31. The address field shall be the NODE DL-address of the DLE to which the LAS role is being transferred. The parameters field shall be null.

The user data field shall convey a higher-level LAS-data-base-status SPDU (see 11.3.2.3) whose maximum size is limited as specified in 8.20.1.4, or shall be null. When present, the contents of this SPDU shall reflect the most recent schedule construction activities, if any, prior to sending the TL DLPDU.

8.20.3 Sending the TL DLPDU

The LAS DLE shall send a TL DLPDU in response to the receipt of an explicit scheduling request received by the LAS from another LM DLE on the local link, requesting transfer of the scheduling-token and the associated LAS role. It is a protocol error to send a TL DLPDU to a non-requesting DLE.

NOTE 1 Such a request can only be made through a higher-level SPDU sent to the current LAS' DL-support functions. DL-management is not permitted to command such a transfer, though it may be able to prompt the requesting LM DLE to request the transfer directly.

If so instructed by DL-management, the LAS DLE may compare the NODE DL-address of its associated LM DLE with that of the requesting DLE and reject the transfer request when the requesting DLE has a higher NODE DL-address.

NOTE 2 This ordering improves predictability of which LM DLE will assume the LAS role on local links which do not employ dynamic determination of that LM DLE which is most capable of assuming the LAS role.

A TL DLPDU may be sent on the link when

- a) the sending DLE, the LAS, holds a scheduler token which is the dominant token on the local link;
- b) the LAS DLE has completed the last cycle of "circulating the token", and would otherwise initiate the next cycle of "circulating the token" by explicitly sending a PT DLPDU to the lowest-numbered NODE DL-address represented in V(LL);

NOTE This restriction on transferring the role of LAS eliminates the need to transfer (reliably) the current value of the array V(RTHA) just before sending the TL DLPDU.

- c) the remaining allocated duration of token usage before the next scheduled activity permits the LAS DLE to repeatedly send a TL DLPDU and monitor the local link for a period of immediate-response-recovery-delay slot-times, $V(\text{IRRD}) \times V(\text{ST})$ octet-durations, for any immediate response from the addressed DLE.

After sending a TL DLPDU, the LAS DLE shall monitor the local link for a period of immediate-response-recovery-delay slot-times, $V(\text{IRRD}) \times V(\text{ST})$ octet-durations, waiting for a response:

- 1) If a Ph-Data indication (see 6.4.4) reporting DATA is received, then the DLE shall
 - stop further monitoring;
 - then wait for receipt of the Ph-Data indication reporting END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY;
 - then proceed as in 3).
- 2) If 1) does not apply, and the monitoring period expires, then
 - i) if the bus is not active at that moment (that is, the last-received Ph-Data indication reported END-OF-ACTIVITY), then the DLE shall proceed as in 4).

- ii) if i) does not apply, implying that the link is still active at that moment (that is, the last-received Ph-Data indication reported START-OF-ACTIVITY), then

α) For implementations that were not demonstrable on or before 31 December 1995, the DLE shall monitor the local link for a period of one additional slot-time, $V(ST)$ octet-durations, waiting for a Ph-Data indication:

A) If a Ph-Data indication reporting DATA is received, then the DLE shall proceed as in 1).

B) If a Ph-Data indication reporting END-OF-ACTIVITY is received, and A) does not apply, then the DLE shall stop further monitoring and proceed as in 4).

C) If neither A) nor B) applies, and the monitoring period expires before a Ph-Data indication is received, then the DLE shall

— wait for receipt of the Ph-Data indication reporting END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY;

— then proceed as in 3).

β) Implementations that were demonstrable on or before 31 December 1995 alternatively may just proceed as in α).C) when neither α).A) nor α).B) applies.

- 3) If the link activity was sufficient to infer that DATA was received as in 1), 2)ii) α).A) or 2)ii) α).C), then

i) If the link activity did not result in a DLPDU, then the LAS DLE shall drop its scheduler token and shall continue or resume monitoring the local link until either 3)i)A) or 3)i)B) occurs.

NOTE This monitoring provides for recovery of the LAS role when the transfer is unsuccessful.

A) If the monitoring DLE observes a DLPDU (with correct FCS) then the monitoring DLE shall cease monitoring for LAS role recovery, and shall inform local DL-management of the event.

B) If the monitoring DLE observes a period of 15 slot-times, $15 \times V(ST)$ octet-durations, of continuous non-activity on the local link, then the monitoring DLE shall

— re-assume the scheduler token;

— inform local DL-management of the event;

— resume active operation as the LAS, and commence transmission on the link.

- ii) If the link activity resulted in a DLPDU (with correct FCS) then

A) if the received DLPDU is an SR DLPDU, specifying a reason of “failure — LAS transfer rejected”, then the monitoring DLE shall

— re-assume the scheduler token;

— inform local DL-management of the event;

— resume active operation as the LAS, and commence transmission on the link.

B) else if the received DLPDU is any other DLPDU, then the monitoring DLE shall cease monitoring for LAS role recovery, and shall inform local DL-management of the event.

- 4) If there was no link activity, or the link activity was not sufficient to infer that DATA was received as in 2)i) or 2)ii)α)B), then the LAS DLE may retry re-sending the TL DLPDU, up to the number of times specified as the maximum-retry-count, $V(MRC)$ (see 6.7.1.5), of the local link.

If all attempted retries are unsuccessful, then the LAS DLE shall

- retain the scheduler token and continue active operation as the LAS;
- inform local DL-management of the event;
- start the next transmission within token-recovery-delay, $P(TRD) \times V(ST)$ octet-durations, of the beginning of the current period of link non-activity.

8.20.4 Receiving the TL DLPDU

A received TL DLPDU shall be treated as follows by the receiving DLE.

8.20.4.1 Actions required of all DLEs

The DLPDU shall be reported to the DLE's local DL-management.

8.20.4.2 Additional actions required of a Link-Master class DLE

If the destination DL-address specified by the DLPDU designates the DLE's NODE DL-address, then

a) If

- 1) the receiving DLE is not awaiting receipt of the TL DLPDU;
- 2) the receiving DLE cannot execute the existing schedule, either due to schedule complexity or length, or because the receiving DLE cannot meet the schedule overhead constraint, $V(MSO)$ (see 6.7.5.6), built into the existing schedule; or
- 3) if the schedule construction and live-list information conveyed in the LAS-database-status SPDU within the received TL DLPDU indicates that the receiving DLE does not have a current copy of either the schedule, or the live-list, or both,

then the receiving DLE shall reply with an SR DLPDU within a period of maximum-response-delay slot-times, $V(MRD) \times V(ST)$ octet-durations, as measured at the receiving DLE, with a status of "failure —LAS transfer rejected".

- b) Otherwise, when a) does not apply, then the receiving DLE shall assume the scheduler token, activate its LAS functions, and re-commence operation as the LAS.

8.20.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.20.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.20.4.4 Additional actions required of the current LAS DLE

If the DLE did not just send the DLPDU, then the DLE shall drop the scheduler token and deactivate its role as LAS; and shall inform local DL-management of the event.

8.21 Wakeup (WK) DLPDU

A WAKEUP (WK) DLPDU is used by the current LAS DLE to stimulate a class B or class C fractional duty cycle (FDC) DLE on the local link (see 3.4.21) to transition to a fully operational state, capable of direct communication.

Transmission of the WK DLPDU usually occurs as a result of previous explicit scheduling, either by the addressed FDC DLE or by DL-management. However, the LAS DLE, which must also be a bridge DLE to support FDC DLEs, may transmit the WK DLPDU on its own initiative, typically when its bridge-function's queue of DLPDUs waiting for transmission to the FDC DLE exceeds some internal threshold.

8.21.1 Structure of the WK DLPDU

Table 32 – Structure of WK DLPDUs

Frame control	Destination NODE-address
0000 0000	N

8.21.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) the DLPDU's implicit priority, which is URGENT;
- c) the length, number and type of DLPDU addresses.

8.21.1.2 Address field

The address field shall consist of an explicit destination NODE DL-address.

8.21.1.3 Parameters field

The parameters field shall be null.

8.21.1.4 User data field

The user data field shall be null.

8.21.2 Content of the WK DLPDU

The frame control field shall be encoded as specified in Table 32. The address field shall be the NODE DL-address of the DLE which is to receive the WK DLPDU. The parameters and user data fields shall be null.

NOTE The specific encoding of the WK DLPDU minimizes the amount of active DLPDU-recognition circuitry required within a "sleeping" FDC DLE.

8.21.3 Sending the WK DLPDU

The LAS DLE shall send a WK DLPDU when requested by prior explicit scheduling. The link capacity used for sending the WK DLPDU shall be deducted (to the extent possible) from the remaining allocated duration of scheduler token usage for scheduled traffic, which is $V(\text{MST}) \times V(\text{TTRT})$ octet-durations, within the current cycle of "circulating the token".

A bridge DLE may also send a WK DLPDU when so requested by the DLE's own bridge-forwarding functions when the DLE holds a scheduler token or delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage prior to the next scheduled activity permits completion of the DLPDU's transmission prior to expiration of the token.

8.21.4 Receiving the WK DLPDU

A received WK DLPDU shall be treated as follows by the receiving DLE:

8.21.4.1 Actions required of all DLEs

If the destination DL-address specified by the DLPDU designates the DLE's NODE DL-address, then the DLE shall activate the remainder of its DLE functions and prepare for active communication on the local link.

NOTE Receipt of this DLPDU has no impact on non-FDC DLEs.

8.21.4.2 Additional actions required of a Link-Master class DLE

None.

8.21.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.21.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.21.4.4 Additional actions required of the current LAS DLE

None.

8.22 Idle (IDLE) DLPDU

An IDLE DLPDU is used by the current token holder to place a DLPDU on the local link which is ignored by all DLEs, other than for its impact on each DLE's sense of link activity, but which may convey information to, or trigger, a data link analysis device.

8.22.1 Structure of the IDLE DLPDU

Table 33 – Structure of IDLE DLPDUs

Frame control	User data
0001 0F10	o-DLSDU

8.22.1.1 Frame control field

The frame control field shall specify

- a) the DLPDU's function;
- b) whether or not transmission of the current DLPDU terminates use of a delegated token.

8.22.1.2 Address field

The address field shall be null.

8.22.1.3 Parameters field

The parameters field shall be null.

8.22.1.4 User data field

The data field shall consist of a single DLSDU whose maximum size is the maximum permitted for an URGENT priority DLPDU. This data field may be null.

8.22.2 Content of the IDLE DLPDU

The frame control field shall be encoded as specified in Table 33. The address and parameter fields shall be null.

The contents of the user data field, possibly null, is not specified by this standard.

8.22.3 Sending the IDLE DLPDU

An IDLE DLPDU may be sent on the link when the sending DLE holds a scheduler token or delegated token which is the dominant token on the local link, and when the remaining allocated duration of token usage, C(RD), permits completion of the IDLE DLPDU's transmission prior to expiration of the token.

NOTE The LAS DLE uses the IDLE DLPDU to fill otherwise unused link capacity until the time of the next scheduled link activity, thereby preventing other DLEs from inferring the LAS DLE's failure from the lack of activity on the link. To facilitate this use, the IDLE DLPDU has a minimum size of one octet.

A local management request may result in the transmission of an IDLE DLPDU.

If the DLE holds a delegated token, and this DLPDU is the final DLPDU which needs to be sent on this use of the delegated token, then the DLE may set the final-token-use subfield of the IDLE DLPDU to the value FINAL; else that subfield shall have the value NOT-FINAL.

8.22.4 Receiving the IDLE DLPDU

A received IDLE DLPDU shall be treated as follows by the receiving DLE:

8.22.4.1 Actions required of all DLEs

The DLE shall ignore the DLPDU on receipt, except for possible local management reporting or special instrumentation purposes.

8.22.4.2 Additional actions required of a Link-Master class DLE

None.

8.22.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.22.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.22.4.4 Additional actions required of the current LAS DLE

If the final-token-use subfield of the received DLPDU has the value FINAL, then the LAS DLE shall

- a) assume that the current use of the delegated token is terminated as if the token had been returned by an RT (see 8.17) DLPDU;
- b) assume that the scheduler token is again dominant on the local link and resume active operation as the LAS.

8.23 Spare DLPDUs

A SPARE DLPDU is reserved for future standard use, and may not be sent by any DLE.

8.23.1 SPARE DLPDU code points

Table 34 – Assumed structure of undefined (spare) DLPDUs

format	Frame control
1	0000 0x11
2	0010 0x01
3	00xx 1xxx
4	0100 xxxx
5	101x xx00

A SPARE DLPDU shall consist of a frame control field. Each spare FC code point is reserved for future standard use.

The DLPDU format and contents of spare code points are subject to change if and when used in a revision of this standard. Until that time they shall not be used.

8.23.1.1 Frame control field

The frame control field shall specify the DLPDU's function — reserved spare. No DLPDU priority is implied.

8.23.1.2 Address field

The address field shall be null.

8.23.1.3 Parameters field

The parameters field shall be null.

8.23.1.4 User data field

The user data field shall be null.

8.23.2 Content of a SPARE DLPDU

The frame control field shall be encoded as specified in Table 34. The address, parameters and user data fields shall be null.

8.23.3 Sending a SPARE DLPDU

SPARE DLPDUs may not be sent or forwarded.

8.23.4 Receiving a SPARE DLPDU

A received SPARE DLPDU shall be treated as follows by the receiving DLE.

8.23.4.1 Actions required of all DLEs

The DLE shall ignore the DLPDU on receipt, except for possible local management reporting or special instrumentation purposes.

8.23.4.2 Additional actions required of a Link-Master class DLE

None.

8.23.4.3 Additional actions required of a Bridge class DLE

- a) Since every Bridge class DLE has Link Master capability, any actions specified in 8.23.4.2 also apply to a Bridge class DLE.
- b) The DLE shall not forward the DLPDU.

8.23.4.4 Additional actions required of the current LAS DLE

None.

8.24 Reserved (not to be used) DLPDUs

A RESERVED (NOT TO BE USED) DLPDU is reserved for future standard use, and may not be sent by any DLE.

8.24.1 RESERVED (NOT TO BE USED) DLPDU code points

Table 35 – Assumed structure of RESERVED (NOT TO BE USED) DLPDUs

format	Frame control
1	0000 0010
2	0000 010x
3	0010 001x
4	0010 0100
5	0011 0000
6	0101 1x00
7	1001 1x00
8	110x 1x00

The FC code point for a RESERVED (NOT TO BE USED) DLPDU shall not be used; these code points are reserved to simplify decoding of the FC octets of other defined DLPDUs. The set of such reserved code points is shown in Table 35.

The frame control field shall specify the DLPDUs function — reserved (not to be used). It shall not be used.

NOTE Since these FC code points are reserved to simplify DLE FC-code-point decoding, their recognition as being distinct from structurally related FC code points cannot be mandated.

8.24.2 Content of a RESERVED (NOT TO BE USED) DLPDU

No content is defined, since DLPDUs with these code points may not be originated.

8.24.3 Sending a RESERVED (NOT TO BE USED) DLPDU

DLPDUs specifying RESERVED (NOT TO BE USED) FC code points shall not be originated. However, they may be forwarded by a bridge if they were not discarded upon receipt.

8.24.4 Receiving a RESERVED (NOT TO BE USED) DLPDU

If the receiving DLE distinguishes between RESERVED (NOT TO BE USED) and other DLPDUs, then the DLE should ignore the RESERVED (NOT TO BE USED) DLPDU on receipt (except for possible local management reporting or special instrumentation purposes). However, the ability to make such a distinction and to perform such discarding is not required.

9 Type 1: DLPDU-parameter structure and encoding

Boolean variables are all encoded with a common representation:

0: FALSE

1: TRUE

The 3-bit DL protocol version number (VVV), which is used in a number of the DLPDU parameters and SPDUs (see clause 11), shall have the value one (1).

9.1 Structure and encoding of EC-PARAMETERS

An EC DLPDU is used to establish a peer DLC between two DLS-users, or a multi-peer DLC between a publishing DLS-user and subscribing DLS-users.

Table 36 – Structure of an EC DLPDU's parameters

DLC basic attributes	DLC attributes when sending	DLC attributes when receiving

The connection parameters (EC-parameters) field, which partitions roughly as shown in Table 36, shall specify the parameters for the proposed DLC:

- a) The first two octets, ordered as shown in Table 37 and Table 38, specify basic attributes of the DLC, beginning with the DL protocol version number:

Table 37 – EC-parameters: 1st octet

Reply request	Publisher-DLCEP-address reuse discriminator			Path diversity	DL-protocol version		
R	NNN			Q	VVV		
7	6	5	4	3	2	1	0

- 1) a 1-bit reply-request subfield (R), specifying whether a reply is requested (=1) or not (=0);

NOTE This field is provided to ensure proper action independent of any state of the receiving DLCEP.

- 2) a 3-bit publisher-DLCEP-address reuse-discriminator subfield (NNN), designating the current set of EC-parameters associated with a PUBLISHER DLCEP-address

i) encoded as zero, when the sender's DLCEP-class (CC) is not PUBLISHER;

ii) assigned by the DLE, when the sender's DLCEP-class (CC) is PUBLISHER

NOTE The DLE assigns a value to this subfield whenever it receives a request to initiate a new PUBLISHER DLCEP from either a local DLS-user in a DL-CONNECT request primitive, or from a remote DLS-user in an EC DLPDU. The choice of value can be random, or can be based on knowledge of recently used values which the DLE avoids.

This subfield is used to discriminate between two different DLCs using the same publisher-DLCEP-address. This could happen when the first DLC had been disconnected and the publishing DLE used the same DLCEP-address for establishing the second DLC, which is different from the first DLC. If a subscriber to the first DLC did not receive the DC DLPDU for that DLC, it could use this field of the just received EC DLPDU to determine that the received EC DLPDU is for a different DLC. Therefore, if the publisher-DLCEP should send an EC DLPDU for the same DLC, it does not change this subfield.

- 3) a one-bit DLL path-diversity subfield (Q), encoded as:

0: any-path;

1: this-path — use the path on which this EC DLPDU was received;

NOTE This subfield provides a means by which a DLE can restrict all communications at a DLCEP to a specific DL-path, thereby providing a means for testing the specific DL-path. The means by which the THIS-PATH value is selected, and by which the actual path employed is chosen, is a DLE-local issue.

- 4) a 3-bit DL protocol version number (VVV), whose value is specified at the beginning of clause 9;

Table 38 – EC-parameters: 2nd octet

Sender's DLCEP class		DL-priority		DL-address size		DLPDU authentication	
CC		PP		SS		XX	
7	6	5	4	3	2	1	0

- 5) a two-bit sender's DLCEP-class subfield (CC), encoded as:

00: reserved for future standard use;
 01: PEER DLCEP;
 10: PUBLISHER DLCEP;
 11: SUBSCRIBER DLCEP;

- 6) a two-bit DLL priority subfield (PP), encoding the DLL priority as (see 7.2.1.3):

00: reserved for future standard use;
 01: URGENT (high) priority;
 10: NORMAL (medium) priority;
 11: TIME-AVAILABLE (low) priority;

- 7) a two-bit DLL address size subfield (SS), encoding the proposed DLL address size (see 7.2.1.1) as:

00: VERY-SHORT — addresses are omitted where possible;
 01: SHORT;
 10: LONG;
 11: reserved for future standard use;

- 8) a two-bit DLPDU-authentication subfield (XX), encoding the required DLPDU authentication (see 6.2.4) as:

00: ORDINARY;
 01: reserved for future standard use;
 10: SOURCE;
 11: MAXIMAL;

- b) The next four octets, ordered as shown in Table 39 and Table 40, specify the maximum confirm delay, in units of 1 ms, for the sender's DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY request primitives, and DL-DATA request primitives, respectively. The DLS-user value UNLIMITED shall be encoded as the value FFFF₁₆ (all ones).

Table 39 – EC-parameters: 3rd and 4th octets

Sender's Maximum Confirm Delay for DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY requests															
MCD_CRS															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

and

Table 40 – EC-parameters: 5th and 6th octets

Sender's Maximum Confirm Delay for DL-DATA requests															
MCD_D															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

NOTE Each two-octet maximum confirm delay is sent most-significant-octet first.

- c) The next four octets, ordered as shown in Table 41 through Table 43, specify proposed attributes for the sender's sending direction of data flow:

Table 41 – EC-parameters: 7th octet

Sender's sending DLC attributes							
DLCEP data delivery features		Residual activity	Queue / buffer	Maximum window size			
TT _S		A _S	B _S	WWW _S			
7	6	5	4	3	2	1	0

- 1) a two-bit sending DLCEP data-delivery features subfield (TT_S), encoded as:

00: UNORDERED DLC;

01: ORDERED DLC;

10: DISORDERED DLC;

11: CLASSICAL DLC;

- 2) a one-bit residual activity subfield (A_S), indicating whether residual activity in the specified (sender-to-receiver) direction will be provided when there are no unacknowledged DLSDUs, encoded as a Boolean;

NOTE Residual activity is required when MAXIMAL DLPDU-authentication is specified, or upon DLS-user or DL-management direction. Other methods of requesting residual activity are for future study.

- 3) a one-bit buffer/queue subfield (B_S), indicating whether the source in the specified (sender-to-receiver) direction is a buffer (=1) or a queue (=0);

- 4) a four-bit window-size subfield (WWW_S), indicating the maximum number of previously sent DLSDUs which will be available for retransmission, and encoded as zero when transmission in that direction is not permitted on the DLC;

Table 42 – EC-parameters: 8th octet

Sender's sending DLC attributes (cont.)							
Basic-DLC-parameters format			2-way data exchange	reserved	Timeliness included	Time stamp format	
FFF _S			E _S	0	G _S	HH _S	
7	6	5	4	3	2	1	0

- 5) a three-bit format subfield (FFF_S), giving the desired format of the basic-DLC-parameters portion of the Status-Data-parameters (SD-parameters) in the specified direction, encoded as a three-bit integer:

0: format A — [9.4.2.1A)];

1: format B — [9.4.2.1B)];

2: format C — [9.4.2.1C)];

- 3: reserved for future standard use;
 - 4: format D — [9.4.2.1D)];
 - 5: format E — [9.4.2.1E)];
 - 6: format F — [9.4.2.1F)];
 - 7: format G — [9.4.2.1G)].
- 6) a one-bit 2-way data exchange subfield (E_S), indicating whether DLSDU data may be sent in the sender-to-receiver direction by ED DLPDUs, encoded as a Boolean;
- 7) a one-bit subfield reserved for future standards use, encoded as zero;
- 8) a one-bit timeliness-included subfield (G_S), indicating the presence of a timeliness parameter (see 9.4.2.1B)3) and 9.4.2.1C)5)) associated with transmitted DLSDUs in transmitted DLPDUs, encoded as a Boolean;
- 9) a two-bit time-stamp-format subfield (HH_S), indicating whether a DL-time-stamp (see 9.4.2.2) is included in SD-parameters which accompany transmitted DLSDUs, encoded as:
- 00: format J — [9.4.2.2J)], which is a null field with no time stamp;
 - 01: format K — [9.4.2.2K)], which is a two-octet field with a time stamp period of over 2 s;
 - 10: format L — [9.4.2.2L)], which is a three-octet field with a time stamp period of over 8 min;
 - 11: format M — [9.4.2.2M)], which is a six-octet field with a time stamp period of over 100 years;

Table 43 – EC-parameters: 9th and 10th octets

Sender's sending DLC attributes (cont.)															
Maximum DLSDU size															
$M \dots M_S$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- 10) a two-octet subfield ($M \dots M_S$), specifying the maximum size DLSDU that can be sent on the DLC.

NOTE 1 The two-octet maximum size is transmitted most-significant-octet first.

NOTE 2 The range for each parameter is dependent on the DLC's priority, and can range between zero and 16 times the maximum data length specified in 7.2.4 for the corresponding priority, inclusively.

- d) The next four octets, identical in form to those specified in c) and ordered as shown in Table 44 through Table 46, specify proposed attributes for the sender's receiving direction of data flow:

Table 44 – EC-parameters: 11th octet

Sender's receiving DLC attributes							
DLCEP data delivery features		Residual activity	Queue / buffer	Maximum window size			
TTR		AR	BR	WWWW _R			
7	6	5	4	3	2	1	0

- 1) a two-bit receiving DLCEP data delivery features subfield (TT_R), encoded as specified in c)1);
- 2) a one-bit residual activity subfield (A_R), indicating whether residual activity in the specified (receiver-to-sender) direction is required, encoded as specified in c)2);
NOTE See note after c)2).
- 3) a one-bit buffer/queue subfield (B_R), indicating whether the sink in the specified (receiver-to-sender) direction is a buffer or a queue, encoded as specified in c)3);
- 4) a four-bit window-size subfield ($WWWW_R$), indicating the maximum number of unacknowledged DLSDUs which can be usefully received;

Table 45 – EC-parameters: 12th octet

Sender's receiving DLC attributes (cont.)							
Basic-DLC-parameters format			2-way data exchange	reserved	Timeliness included	Time stamp format	
FFF_R			E_R	0	G_R	HH_R	
7	6	5	4	3	2	1	0

- 5) a three-bit format subfield (FFF_R), giving the desired format of basic-DLC-parameters portion of the SD-parameters in the specified direction, encoded as specified in c)5).
- 6) a one-bit 2-way data exchange subfield (E_R), indicating whether DLSDU data may be sent in the receiver-to-sender direction by ED DLPDUs, encoded as a Boolean;
- 7) a one-bit subfield reserved for future standards use, encoded as zero;
- 8) a one-bit timeliness-included subfield (G_R), indicating the presence in received DLPDUs of a timeliness parameter (see 9.4.2.1b)3) and 9.4.2.1c)5)) associated with received DLSDUs, encoded as a Boolean;
- 9) a two-bit time-stamp-format subfield (HH_R), indicating whether a DL-time-stamp is included in SD-parameters which accompany received DLSDUs, encoded as specified in c)9);

Table 46 – EC-parameters: 13th and 14th octets

Sender's receiving DLC attributes (cont.)															
Maximum DLSDU size															
$M...M_R$															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

- 10) a two-octet subfield ($M...M_R$), specifying the maximum size DLSDU that can be received on the DLC.

NOTE See notes following c)10).

9.2 Structure and encoding of DC-PARAMETERS

A DISCONNECT CONNECTION (DC) DLPDU is used to disconnect an existing or proposed DLC.

The Disconnect-Connection-parameters (DC-parameters) field shall specify the desired DLC-support action and reason, together with any associated operational parameters. This parameter field is two octets, ordered as shown in Table 47 through Table 49.

Table 47 – DC-parameters and RC-parameters: 1st octet

Reply request	reserved				DL-protocol version		
R	0000				VVV		
7	6	5	4	3	2	1	0

- a) a one-octet subfield, coded as in Table 47, consisting of
- 1) a one-bit subfield (R) specifying whether a reply is requested (=1) or not (=0);
NOTE This field is provided to ensure proper action independent of any state of the receiving DLCEP.
 - 2) a four-bit subfield reserved for future standards use and encoded as zero;
 - 3) a 3-bit DL protocol version number (VVV), whose value is specified at the beginning of clause 9;

Table 48 – DC-parameters and RC-parameters: 2nd octet

Reason for DLPDU							
Reason							
7	6	5	4	3	2	1	0

- b) a one-octet subfield, specifying the reason for the requested DLC support action, based on IEC 61158-3, 8.6.2, and coded in hexadecimal as specified in Table 49. All unused reason codes in Table 49 in the hexadecimal range 40 to 7F are reserved for future standards use; the other codes in the range 00 to 3F may be used as desired, and can be interpreted as reason unspecified (by this part of IEC 61158).

9.3 Structure and encoding of RC-PARAMETERS

A RESET CONNECTION (RC) DLPDU is used to reset or disconnect an existing or proposed DLC.

The Reset-Connection-parameters(RC-parameters) field shall specify the desired DLC-support action and reason, together with any associated operational parameters.

This parameter field is four octets, ordered as shown in Table 47 through Table 52:

a) A one-octet subfield, coded as specified in a).

Table 49 – Disconnect reasons

Code	Reason for disconnect	Reason class	Mapping to IEC 61158-3
00	user-originated disconnection — normal condition	user-originated disconnection (00..1F)	8.6.2.5b)1)
02	user-originated disconnection — abnormal condition		8.6.2.5b)2)
1E	user-originated disconnection or connection rejection — reason unspecified		8.6.2.5b)7)
20	user-originated connection rejection — connection not authorized, permanent condition	user-originated connection rejection (20..3F)	8.6.2.5b)4)
21	user-originated connection rejection — unacceptable QoS, permanent condition		8.6.2.5b)4)
22	user-originated connection rejection — non-QoS reason, permanent condition		8.6.2.5b)5)
24	user-originated connection rejection — transient condition		8.6.2.5b)6)
40	provider-originated disconnection — incorrect DLCEP pairing, permanent condition	provider-originated disconnection (40..5F)	8.6.2.5a)1)
41	provider-originated disconnection — wrong publisher-DLCEP-address reuse-discriminator, permanent condition		8.6.2.5a)1)
42	provider-originated disconnection — other permanent condition		8.6.2.5a)1)
43	provider-originated disconnection — wrong DLPDU format or parameters, permanent condition		8.6.2.5a)1)
44	provider-originated disconnection — wrong DLSDU size, permanent condition		8.6.2.5a)1)
45	provider-originated disconnection — transient condition		8.6.2.5a)2)
46	provider-originated disconnection — timeout		8.6.2.5a)3)
5E	provider-originated disconnection or connection rejection — reason unspecified		8.6.2.5a)12)
60	provider-originated connection rejection — DL(SAP) address unknown	provider-originated connection rejection (60..7D)	8.6.2.5a)5)
62	provider-originated connection rejection — DLSAP unreachable, permanent condition		8.6.2.5a)6)
64	provider-originated connection rejection — DLSAP unreachable, transient condition		8.6.2.5a)7)
65	provider-originated connection rejection — inconsistent DLCEP state, permanent condition		8.6.2.5a)8)
66	provider-originated connection rejection — QoS unavailable, permanent condition		8.6.2.5a)8)
68	provider-originated connection rejection — QoS unavailable, transient condition		8.6.2.5a)9)
7E	disconnection or connection rejection, unknown origin — reason unspecified	unknown origin or type (7E..7F)	8.6.2.5c)

b) A one-octet subfield, specifying the reason for the requested DLC support action, based on IEC 61158-3, 8.7.3.5 and 8.7.3.6, and coded in hexadecimal as specified in Table 50. All unused reason codes in Table 50 in the hexadecimal range C0 to FF are reserved for future standards use; the other codes in the range 80 to BF may be used as desired, and can be interpreted as reason unspecified (by this part of this standard).

Table 50 – Reset reasons

Code	Reason for reset	Reason class	Mapping to IEC 61158-3
80	user-originated reset — resynchronization after user timeout	user-originated reset (80..BF)	8.7.3.6b)1)
82	user-originated reset — resynchronization after user-detected user-state inconsistencies		8.7.3.6b)2)
9E	user-originated reset — reason unspecified		8.7.3.6b)3)
C0	provider-originated reset — resynchronization after activation of a DL-management-established DLCEP	provider-originated reset (C0..FD)	8.7.3.6a)1)
C2	provider-originated reset — resynchronization after timeout		8.7.3.6a)2)
C4	provider-originated reset — resynchronization after maximum number of retransmission requests or attempts		8.7.3.6a)3)
C6	provider-originated reset — resynchronization after detected sequence number error		8.7.3.6a)4)
C8	provider-originated reset — resynchronization after other detected DLCEP state inconsistencies		8.7.3.6a)5)
FC	provider-originated reset — reason unspecified		8.7.3.6a)7)
FE	reset, unknown origin — reason unspecified	unknown origin (FE..FF)	8.7.3.6a)1)

Table 51 – RC-parameters: 3rd octet

Modulus number preceding next DLSDU to be sent, if any							
any value			NDS mod 2 ⁵				
7	6	5	4	3	2	1	0

- c) A one-octet subfield, specifying, in its low-order bits, the five low-order bits of the sequence number NDS preceding the sequence number of the next DLSDU to be sent. The high-order three bits of this octet may have any value (for example, the next higher-order bits of NDS) and shall be ignored on reception (see Table 51).

Table 52 – RC-parameters: 4th octet

Modulus number of last complete DLSDU received, if any							
any value			NDR mod 2 ⁵				
7	6	5	4	3	2	1	0

- d) A one-octet subfield, specifying, in its low-order bits, the five low-order bits of the sequence number NDR of the last DLSDU received, if any. The high-order three bits of this octet may have any value (for example, the next higher-order bits of NDR) and shall be ignored on reception (see Table 52).

9.4 Structure and encoding of SD-Parameters

9.4.1 SD-Parameters in DLPDUs addressed to a DL(SAP)-address

A COMPEL ACKNOWLEDGMENT (CA) DLPDU addressed to a DL(SAP)-address is used as the first phase of the remotely confirmed unitdata transfer service to transfer (or retransfer) a limited amount of transparent user data from one DLS-user to another DLS-user and to request the status of the transfer from the receiving DLE.

A COMPEL DATA (CD) DLPDU addressed to a DL(SAP)-address is used as the first phase of the unitdata exchange service to request the immediate transfer (or retransfer) of a limited amount of transparent user data from one DLS-user to another DLS-user. A CD DLPDU always contains a non-null SD-parameters field.

An EXCHANGE DATA (ED) DLPDU addressed to a DL(SAP)-address is used as the first phase of the unitdata exchange service to transfer (or retransfer) a limited amount of transparent user data from one DLS-user to another DLS-user, and to request the immediate transfer (or retransfer) of a limited amount of transparent user data from that second DLS-user to the first DLS-user. An ED DLPDU always contains a non-null SD-parameters field.

A DATA (DT) DLPDU addressed to a DL(SAP)-address is used

- 1) to transfer a limited amount of transparent user data from one requesting DLS-user to one or more other DLS-users without establishing or later releasing a DLC; or
- 2) as the second phase of the unitdata exchange service or remotely-confirmed unitdata-transfer service to acknowledge the transfer of such data without establishing or later releasing a DLC.

The DLPDUs of 1) always contain a null SD-parameters field; the DLPDUs of 2) always contain a non-null SD-parameters field.

NOTE These DLPDUs are also used within the context of a DLC to provide DL-DATA services.

The various SD-parameter formats, and their potential applicability to the CA, CD, DT and ED DLPDUs addressed implicitly or explicitly to a DLSAP-address, shown in Table 53, are:

Table 53 – Structure of connectionless-mode CA, CD, DT and ED DLPDUs

Connectionless-mode DLPDUs					
format	Frame control	d-DLSAP-address	s-DLSAP-address	Parameters	User data
1	CA	[HL.]N.S	[HL.]N.S	format R	DLSDU
2	CD	[HL.]N.S	[HL.]N.S	format R	—
3	ED	[HL.]N.S	[HL.]N.S	format R	DLSDU
4	DT 1	[HL.]N.S	[HL.]N.S	format P	DLSDU
5	DT 2	[HL.]N.S	—	format U	o-DLSDU
6	DT 3	implied PSA	—	format U	—

format	SD-parameters
	Non-DLC-parameters

P	— (null) —

R	ZZ PP N(LTI)

U	XXXX N(LTI)

- a) The **null** format, format P, of length zero octets, is used for the unitdata transfer service. It is always used in DT DLPDUs which have two explicit addresses (see 8.7.1 formats 1L and 1S).
- b) The **initiator** format, format R (see Table 54), of length one octet, is used for the first phase of the two-phase transaction required for unitdata-exchange and remotely-confirmed unitdata-transfer services. It is always used in CA, CD and ED DLPDUs, all of which always have two explicit addresses (see 8.4.1, 8.5.1 and 8.6.1 formats 1L and 1S).

Table 54 – Short format SD-parameters for connectionless transaction initiators

reserved		DLSDU priority and presence		Initiator's transaction index			
ZZ		PP		N(LTI)			
7	6	5	4	3	2	1	0

The octet comprises

- 1) a two-bit reserved subfield, reserved for future standards use, which shall be zero;
 - 2) a two-bit DLSDU-priority-and-presence subfield, specifying the presence and DLL priority of the accompanying DLSDU (see 7.2.1.3), encoded as
 - 00: no accompanying DLSDU;
 - 01: URGENT (high) priority accompanying DLSDU;
 - 10: NORMAL (medium) priority accompanying DLSDU;
 - 11: TIME-AVAILABLE (low) priority accompanying DLSDU;
 - 3) a four-bit transaction-index subfield, specifying the transaction index assigned during CA, CD or ED formation and transmission as specified in 8.4.3, 8.5.3 and 8.6.3.
- c) The **responder** format, format U (see Table 55), of length one octet, is used for the second phase of the two-phase transaction required for unitdata-exchange and remotely-confirmed unitdata-transfer services. It is always used in DT DLPDUs which do not have an explicit source address (see 8.7.1 formats 2L, 2S and 4):

Table 55 – Short format SD-parameters for connectionless responders

Status				Initiator's transaction index			
XXXX				N(LTI)			
7	6	5	4	3	2	1	0

- 1) a four-bit status subfield, encoded as specified in Table 56;
- 2) a four-bit transaction-index subfield, identical to b)3).

Table 56 – Reply status for unitdata-acknowledgment and exchange-unitdata-reply DT DLPDUs

Short name	Definition	Hexadecimal coding	Mapping to IEC 61158-3
OK	success — no reply data available at responder or service does not provide this information	0	9.5.1.2.5a) 9.5.2.2.5a)
OK_U	success — URGENT priority reply data available at responder	1	9.5.2.2.5a)
OK_N	success — NORMAL priority reply data available at responder	2	9.5.2.2.5a)
OK_TA	success — TIME-AVAILABLE priority reply data available at responder	3	9.5.2.2.5a)
RR	failure —resource limitation in responder — no reply data available at responder or service does not provide this information (note 1)	4	9.5.1.2.5j) 9.5.2.2.5m)
RR_U	failure —resource limitation in responder — URGENT priority reply data available at responder (note 1)	5	9.5.2.2.5c)
RR_N	failure —resource limitation in responder — NORMAL priority reply data available at responder (note 1)	6	9.5.2.2.5c)
RR_TA	failure —resource limitation in responder — TIME-AVAILABLE priority reply data available at responder (note 1)	7	9.5.2.2.5c)
RF	failure —fault in responder	8	9.5.1.2.5k) 9.5.2.2.5e)
RI	failure —responder DL(SAP)-role incompatible with this DLPDU (note 2)	9	9.5.1.2.5g) 9.5.2.2.5h)
RA	failure — response restricted to a different peer DLSAP-address (note 3)	A	9.5.2.2.5g)
—	reserved for compatibility with the SR DLPDU's reply status — not available for other use	B	
BF	failure — fault in intermediary bridge	C	9.5.2.2.5f)
BR	failure — resource limitation in intermediary bridge	D	9.5.1.2.5n) 9.5.2.2.5f)
BOK	reserved for interim success — intermediary bridge is forwarding transaction	E	
DR	interim success — delayed reply; end station needs more time to prepare response	F	
NOTE 1 This status can occur when an addressed responder cannot buffer the received DLSDU (possibly because of a queue-full condition). In such a case the responder is not permitted to send a DLSDU in reply. NOTE 2 This status can be generated only when the DL(SAP)-role associated with the received destination DLSAP-address is BASIC or INITIATOR. NOTE 3 This status can be generated only when the DL(SAP)-role associated with the received destination DLSAP-address is CONSTRAINED RESPONDER.			

9.4.2 SD-Parameters in DLPDUs addressed to a DLCEP

A COMPEL ACKNOWLEDGMENT (CA) or EXCHANGE DATA (ED) DLPDU addressed to a DLCEP is used to transfer (or retransfer) a limited amount of transparent user data from one DLS-user to one or more other DLS-users and to request the status of the remote DLCEP.

A COMPEL DATA (CD) or EXCHANGE DATA (ED) DLPDU addressed to a DLCEP is used to request the immediate transfer (or retransfer) of a limited amount of transparent user data from one DLS-user to one or more other DLS-users. A CD DLPDU may contain an SD-parameters field; a CA or ED DLPDU always contains an SD-parameters field.

A DATA (DT) DLPDU addressed to a DLCEP is used to transfer a limited amount of transparent user data from one requesting DLS-user to one or more other DLS-users within the context of a DLC; or to acknowledge the transfer of such data within the context of a DLC. These DLPDUs always contain an SD-parameters field (which may be null).

All four types of DLPDU assist in the synchronization of DLCEPs and of their DLS-users:

NOTE These DLPDUs are also used outside the context of a DLC to provide DL-UNITDATA services.

- a) When a CD DLPDU is being sent by an LAS DLE, or such a DLPDU is being forwarded by a bridge, then the DLPDU shall not contain an explicit source DLSAP-address and its parameter field (SD-parameters) shall be null.
- b) When a CA, CD, DT or ED DLPDU is being sent from a DLCEP, or being forwarded by a bridge, and the DLPDU does not contain user data, then the negotiated parameter field shall be present, and shall have two subparts:
 - 1) the first shall have fixed structure and length [formats A – G] as determined during the DLC establishment process (see 9.1c)5) or 9.1d)5));
 - 2) the second either
 - shall have the fixed structure and length [formats J – M] determined during the DLC establishment process (see 9.1c)5) or 9.1d)5)), or
 - shall be the null format, format J.
- c) In all other cases, the negotiated parameter field shall be present, and shall have two subparts each with fixed structure and length [formats A – G, concatenated with formats J – M] as determined during the DLC establishment process (see 9.1c)5) or 9.1d)5)).
- d) All DLEs shall support formats A, C, F, G and J.

NOTE Support of formats A and G is required to facilitate migration from existing national standards.
- e) DLEs which support both **timeliness** and a time-synchronism class (see 12.6.3) other than **none** shall support formats K, L and M.

The various SD-parameter formats, and their potential applicability to the CA, CD, DT and ED DLPDUs, are shown in Table 57.

Table 57 – Structure of connection-oriented CA, CD, DT and ED DLPDUs

Connection-oriented DLPDUs					
format	Frame control	d-DLCEP-address	s-DLCEP-address	Parameters	User data
	CA 1	[HL.]N.S	[HL.]N.S	parameters	o-pDLSDU
2	CA 2	[HL.]N.S	—	parameters	o-pDLSDU
3	CD 1	[HL.]N.S	[HL.]N.S	o-parameters	—
4	CD 2	[HL.]N.S	—	o-parameters	—
5	ED 1	[HL.]N.S	[HL.]N.S	parameters	pDLSDU
6	ED 2	[HL.]N.S	—	parameters	pDLSDU
7	DT 1	[HL.]N.S	[HL.]N.S	parameters	o-pDLSDU
8	DT 2	[HL.]N.S	—	parameters	o-pDLSDU
9	DT 3		[HL.]N.S	parameters	o-pDLSDU
10	DT 4	implied PSA	—	parameters	o-pDLSDU
11	DT 5		implied PDA	parameters	o-pDLSDU

format

SD-parameters

Basic-DLC-parameters

DLSDU source-time-stamp

A

— (null) —

B

J

K

T

NDR

NDS

C

NDR

RSN

J

K

T

NDS

TNS

ASN

D

NDR

RSN

J

K

T

NDS

E

NDR

RSN

F

J

K

T

NDS

TNS

ASN

G

J

K

T

NDS

J

— (null) —

K

DL-

time

L

DL-

time

M

DL-

time

9.4.2.1 Parameters conveying DLCEP state and DLSDU timeliness

The formats for the first subpart of the SD-parameters have been given names that reflect their expected usage. These formats are:

- A) the **null** format, format A, of zero octets in length;
- B) the **short** format, format B (see Table 58), of length one octet, is as follows:

Table 58 – Short format SD-parameters for DLCEP state

Retransmit	Acknowledge	Timeliness	Modulus number of post-acknowledged or requested DLSDU		Modulus number of associated DLSDU or of highest-numbered DLSDU sent		
J	K	T	NDR mod 2 ²		NDS mod 2 ³		
7	6	5	4	3	2	1	0

- 1) a one-bit retransmission-request (selective-reject) subfield, requesting (when J=1) retransmission of the NDR'th DLSDU;
- 2) a one-bit acknowledgment subfield, acknowledging (when K=1) that all DLSDUs before the NDR'th have been received and reported to the DLS-user;
- 3) a one-bit timeliness subfield, indicating (when T = 1) that the associated DLSDU originated in a buffer with associated timeliness criteria, and that those timeliness criteria were met;
- 4) a two-bit subfield, specifying the low-order two bits of the sequence number NDR of the DLSDU being requested, or of the DLSDU after the one being acknowledged, or both;
- 5) a three-bit subfield, specifying the low-order three bits of the sequence number NDS of the associated DLSDU (if this DLPDU contains user data) or of the highest-numbered DLSDU which has been sent (if this DLPDU does not contain user data);
- 6) the implied values for the fields of 9.4.2.1c) which are omitted from 9.4.2.1b) are zero.

NOTE DLSDUs sent from a DLCEP are assigned consecutive sequence numbers, starting with one, before transmission.

- C) The **long** format, format C (see Table 59 through Table 61), of length three octets, is as follows:

Table 59 – Long format SD-parameters for DLCEP state: 1st octet

Modulus number of post-acknowledged or requested DLSDU				Zero-based Requested Segment Number			
NDR mod 2 ⁴				RSN			
7	6	5	4	3	2	1	0

- 1) a four-bit subfield, specifying the low-order four bits of the sequence number NDR of the DLSDU being requested, or of the DLSDU after the one being acknowledged, or both;
- 2) a four-bit subfield, specifying the zero-based segment number of the DLSDU segment being requested, or zero when no segment is being requested;

Table 60 – Long format SD-parameters for DLCEP state: 2nd octet

Retransmit	Acknowledge	Timeliness	Modulus number of associated DLSDU or highest-numbered DLSDU Sent				
J	K	T	NDS mod 2 ⁵				
7	6	5	4	3	2	1	0

- 3) a one-bit retransmission-request (selective-reject) subfield, requesting (when J=1) retransmission of the NDR'th DLSDU;
- 4) a one-bit acknowledgment subfield, acknowledging (when K=1) that all DLSDUs before the NDR'th have been received in their entirety and reported to the DLS-user;
- 5) a one-bit timeliness subfield, indicating (when T = 1) that the associated DLSDU originated in a buffer with associated timeliness criteria, and that those timeliness criteria were met;

- 6) a five-bit subfield, specifying the low-order five bits of the sequence number NDS of the associated DLSDU (if this DLPDU contains user data) or of the highest-numbered DLSDU which has been sent (if this DLPDU does not contain user data);

NOTE If this DLPDU contains no user data, then all previously-requested DLSDUs must have been sent in their entirety.

Table 61 – Long format SD-parameters for DLCEP state: 3rd octet

Zero-based Total Number of Segments in DLSDU				Zero-based Associated Segment Number			
TNS				ASN			
7	6	5	4	3	2	1	0

- 7) a four-bit subfield, specifying the zero-based total number of segments in the associated DLSDU, or zero when the DLPDU contains no user data;
- 8) a four-bit subfield, specifying the zero-based segment number of the associated DLSDU segment, or zero when the DLPDU contains no user data.

NOTE This representation has the following benefits:

- a) The zero-based segment number, coupled with the fact that all segments except the last are the maximum size permitted by the DLPDU's priority, facilitates DLSDU reassembly during reception.
- b) The total number of segments in the actual DLSDU can be used to allocate a buffer or queue record of appropriate size upon first receipt of any segment of the DLSDU.

- D) The **unsegmented long** format, format D, of length two octets, consists of the first two octets of format C (see Table 59 and Table 60).
- E) The **subscriber** format, format E, of length one octet, consists of the first octet of format C (see Table 59), with implied values for the fields J, K and T of the absent second octet. The implied values for the J, K and T fields of 9.4.2.1C), which are omitted from 9.4.2.1E), are one (1), zero (0) and zero (0), respectively.
- F) The **publisher** format, format F, of length two octets, consists of the last two octets of format C (see Table 60 and Table 61). The values for the J and K fields of 9.4.2.1C) shall be zero (0).
- G) The **unsegmented publisher** format, format G, of length one octet, consists of the second octet of format C (see Table 60). The values for the J and K fields of 9.4.2.1C) shall be zero (0).

Only formats A, C, F and G need be supported; support for formats B, D and E is optional.

9.4.2.2 Parameters conveying DLSDU time-of-production

The formats for the second subpart of the SD-parameters are

- J) the **no-time** format, format J, of zero octets in length;
- K) the **2-second** format, format K, of length two octets, consists of the third-lowest-order and second-lowest-order octets of the originating DLE's DL-time at the moment of DL-Put request execution, in that order. It is adequate to distinguish the timing of events less than 2 s apart, and provides a potential resolution of 2^{-5} ms;
- L) the **8-minute** format, format L, of length three octets, consists of the fourth-lowest-order, third-lowest-order and second-lowest-order octets of the originating DLE's DL-time at the moment of DL-Put request execution, in that order. It is adequate to distinguish the timing of events less than 8 min apart, and provides a potential resolution of 2^{-5} ms;
- M) the **full-time** format, format M, of length six octets, consists of the high-order six octets of the originating DLE's DL-time at the moment of DL-Put request execution, encoded most-significant-octet first. It is adequate to distinguish the timing of any two events, and provides a potential resolution of 2^{-5} ms.

9.5 Structure and encoding of SR-parameters

There are two SR-parameter formats:

- a) The **null** format, format X, of length zero octets, is identical to SD-parameter format P and is used to indicate that the sending bridge was able to buffer for forwarding a received CA, CD or ED DLPDU, when the reply token was created by that received DLPDU. By using this format, the bridge implicitly returns the symbolic status “BOK” as specified in Table 62.

Table 62 – Reply status for SR DLPDUs

Short name	Definition	Hexadecimal coding
RR	failure — resource limitation in responder	4
RI	failure — responder DLE-class incompatible with this request	9
RTR	failure — LAS transfer rejected	B
BF	failure — fault in intermediary bridge	C
BR	temporary failure — resource limitation in intermediary bridge	D
BOK	interim success — intermediary bridge is forwarding transaction	E

NOTE This table is necessarily a subset of Table 56.

- b) The **error** format, format Y (see Table 63), is based on SD-parameter format U (see Table 55), and so is of length one octet. Format Y is used
- to indicate that a bridge which should have forwarded a received CA, CD or ED DLPDU was unable to buffer the received DLPDU, when the reply token was created by that received DLPDU (reasons BF or BR);
 - to reject a CA, CD or ED DLPDU received at a DLCEP under certain error conditions;
 - to reject an attempt to transfer the LAS role to the replying DLE, when the reply token was created with a TL DLPDU (reasons RR, RI and RNC).

The subfields of the SR-parameters field’s error format are.

- 1) a four-bit status subfield, XXXX, encoded as specified in Table 62;

NOTE The only two values possible for the returned status are “BF” and “BR”.

- 2) a four-bit subfield, ZZZZ, coded as zero and reserved for future standard use.

Table 63 – Short format SR-parameters

Status				must be zero			
XXXX				ZZZZ			
7	6	5	4	3	2	1	0

9.6 Structure and encoding of TD-parameters

The TD-parameters field shall be composed of four subfields (see Table 64):

Table 64 – Structure of TD-parameters

TD-parameters and the variable values transmitted					
Field type	Link originating DL-time	DL-time quality	DL-time offset	DL-time prior to end-of-transmission	DL-time adjustment
Value as sent	$V_S(\text{TSL})$	$V_S(\text{TQ})$	$V_S(\text{DLTO})$	$V_S(\text{DLTO}) + V_S(\text{LSTO}) + C_S(\text{NT})_1$	$P_S(\text{TD}) = C_S(\text{NT})_2 \sim C_S(\text{NT})_1$
Name at receipt	$N_S(\text{TSL})$	$N_S(\text{TQ})$	$N_S(\text{DLTO})$	$N_S(\text{DLT})$	$N_S(\text{DLTA})$
Field size	two octets	one octet	seven octets	seven octets	three octets

- a) The link designator of the DL-time-source DLE — the DLE which originates the sense of DL-time on the extended link — shall be expressed in two octets as specified in 6.3.

NOTE When a bridge interconnects two or more single links into an extended link, this is the link-id of the root link of the spanning tree formed by the bridges which interconnect the links of the extended link. It is the local link-id only for DLEs on that root link, or when no bridges interconnect the local link with other links; in this latter case, the link-id may have the value zero.

- b) The quality of DL-time shall be expressed in one octet as follows (see Table 65):

Table 65 – Structure and encoding of the DL-time-quality measures

Limiting time-synchronism class			Number of intervening links			Time-source type	
TTT			LLL			SS	
7	6	5	4	3	2	1	0

- 1) a three-bit subfield, TTT, specifying the least-capable time-synchronism class (see 12.6.3) of all of the DLEs on the DL-time propagation path from the DL-time source to the DLE, inclusive, encoded as:

000 NONE;
 001 1 s;
 010 100 ms;
 011 10 ms;
 100 1 ms;
 101 100 µs;
 110 10 µs;
 111 1 µs;

- 2) a three-bit subfield, LLL, specifying the number of intervening links on the DL-time propagation path from the DL-time source DLE to the sending DLE, expressed as 0 to 7, where

0 indicates that the DL-time originates with the DLE itself;
 1 indicates that the DL-time originates with another DLE on the local link;
 2 indicates that the DL-time originates with a DLE on a link one bridge removed, and so forth;

- 3) a two-bit subfield, SS, expressing the method of synchronization of the DL-time source DLE with Universal Coordinated Time (UTC), the worldwide time standard, encoded as

00 some DLE's node-time, locally generated and not received from an extra-DLS-provider source;

01 local time (not UTC) received (directly or indirectly) from a human source;

10 UTC received (directly or indirectly) from a human source or an unreliable electronic source, or previously (but no longer) received (directly or indirectly) from a reliable electronic source;

11 UTC continuously received (directly or indirectly) from a reliable electronic source.

NOTE 1 Reliable electronic sources include radio receivers for national time broadcasts, atomic clocks, and similar sources which provide intrinsic worldwide time coordination.

NOTE 2 The mechanisms for synchronizing a DLE which originates the extended link's sense of DL-time with an external time source is necessarily implementation-specific and beyond the scope of standardization.

- c) The DL-time-offset, $V_S(DLTO)$, of the sending DLE during DLPDU formation shall be expressed as a signed seven-octet integer encoded most-significant-octet first, where the least significant bit represents a time granularity of approximately 2^{-13} ms as shown in Table 66.
- d) The DL-time, $C_S(NT)_1 + V_S(LSTO) + V_S(DLTO)$, of the sending DLE at a moment during DLPDU formation, less than 1 s prior to transmission, shall be expressed as a seven-octet non-negative integer encoded most-significant-octet first (see Table 66), where the most significant bit is always zero and the least significant bit represents a time granularity of approximately 2^{-13} ms.

NOTE The granularity of encoded DL-time does not imply that a DLE increments DL-time at that granularity, but rather that each time the DLE increments its sense of DL-time, it does so by an amount that causes the average rate of incrementation to be approximately 2^{13} counts per millisecond (which is $2^{13} \times 10^3$ counts per second).

Table 66 – Approximate numeric significance of the bits of seven-octet DL-time

Octet of DL-time in transmission order	Symbolic contents	Approximate weight of the low-order bit of the octet
1	0YYYYYYY	1,09 year
2	DDDDDDDD	1,55 day
3	HHHHHMMM	8,74 min
4	MMSSSSSS	2,05 s
5	Smmmmmmm	8,00 ms
6	mmm•µµµµµµ	31,25 µs
7	µµµµµnnn	122 ns

- e) The time-adjustment, $P_S(TD) = C_S(NT)_2 \sim C_S(NT)_1$, is a small adjustment to account for systemic offsets in the accompanying DL-time subfield d), caused by known implementation considerations within the sending real end system when sending a TD DLPDU, such as early sampling of $C_S(NT)_1$ during TD DLPDU formation and transmission, such that the sum of the values specified by d) and e) will be the DL-time,

$$V_S(DLTO) + V_S(LSTO) + C_S(NT)_2,$$

at which the END-OF-DATA-AND-ACTIVITY PhIDU (see 7.1) will be sent to the associated PhE.

The time-adjustment shall be expressed as a three-octet unsigned integer encoded most-significant-octet first (see Table 67), where the least significant bit represents a time granularity of approximately 2^{-13} ms.

NOTE 1 This field is included to avoid the requirement that the DLE perform an equivalent real-time adjustment to the full seven octet DL-time specified by c).

NOTE 2 This adjustment will vary inversely with the DLE's actual instantaneous rate of data transmission.

Table 67 – Approximate numeric significance of the bits of three-octet short time

Octet of short time in transmission order	Symbolic contents	Approximate weight of the low-order bit of the octet
1	Smmmmmmm	8,00 ms
2	mmm•µµµµµµ	31,25 µs
3	µµµµµnnn	122 ns

NOTE The objective of the computation of d) and e) is to ensure that sum of subfields d) and e) is larger than the sender's $V_S(NTO)$ such that

- A) were the sender to measure its round-trip-delay, $V_S(MD)$ (see 10.4.1.5), when communicating with itself through the PhE and medium, then
- B) if the sender were to receive its own transmissions of TD DLPDUs by receiving while transmitting, then
- C) the sum of subfields d) and e) and, as computed by the sender from the received TD DLPDU, would be equal to the current value of
 $V_S(DLTO) + V_S(LSTO) + C_S(NT)$
 at the sender at the moment that the sender's PhE reports completion of reception of the TD DLPDU from the local medium.

9.7 Structure and encoding of RQ-parameters

The RQ-parameters field shall be composed of one subfield (see Table 68):

Table 68 – Structure of RQ-parameters

RQ-parameters
$C_S(NT)$
short-time ₁
three octets

- a) The parameter subfield shall equal the low-order 24 bits of the node-time, $C_S(NT)$, of the sending DLE at the moment of RQ DLPDU formation just prior to (or during) transmission, representing the low-order three octets of the DLS-provider's current node-time in units of 2^{-13} ms.

NOTE The contents of this subfield reflect the relative time at which the RQ DLPDU is transmitted.

If the sending DLE's time-synchronization class (see 12.6.3) is NONE and the DLE does not maintain even an estimated $C(NT)$, then it may use any value for this subfield.

Table 67 shows the transmission order and approximate numeric significance of the octets of this subfield. It is encoded as a three-octet unsigned integer sent most-significant octet first.

9.8 Structure and encoding of RR-parameters

The RR-parameters field shall be composed of four subfields (see Table 69):

Table 69 – Structure of RR-parameters

RR-parameters			
$N_S(TQ)_{7-5}$	$N_1(NT)$	$N_2(NT)$	$C_S(NT)$
measurement quality	short-time ₁	short-time ₂	short-time ₃
one octet	three octets	three octets	three octets

- a) The first parameter subfield shall be encoded as specified in Table 70, and shall specify
- 1) the time-synchronism class of the replying DLE, TTT, encoded as in b)1);
 - 2) whether a DLE with a time-synchronism class of NONE maintains an estimated $C(NT)$, E, encoded
 - i) as a Boolean (0=FALSE, 1=TRUE) when TTT has the value 000;
 - ii) as 1 (TRUE) when TTT has a value other than 000;
 - 3) a 4-bit subfield, encoded as zero.

Table 70 – Structure and encoding of the RR-time-quality measures

Limiting time-synchronism class			DLE maintains estimated $C(NT)$	reserved			
TTT			E	0000			
7	6	5	4	3	2	1	0

- b) The second and third parameter subfields shall equal the received and locally appended parameter subfields, respectively, of the received RQ DLPDU.
- c) The fourth parameter subfield shall equal the low-order 24 bits of the node-time, $C_S(NT)$, of the replying DLE at the moment of RR DLPDU formation just prior to (or during) transmission, representing the low-order three octets of the DLS-provider's current node-time in units of 2^{-13} ms.

NOTE The contents of this subfield reflect the relative time at which the RR DLPDU is transmitted. The resulting round-trip-delay computations are accurate, independent of any internal delays within the DLEs, provided that the sum of the propagation delays between the two DLEs is less than 2 s.

If the sending DLE's time-synchronization class (see 12.6.3) is NONE and the DLE does not maintain even an estimated $C(NT)$, then it shall encode this subfield as zero.

Table 67 shows the transmission order and approximate numeric significance of the octets of each of the last three subfields. Each of these subfields is encoded as a three-octet unsigned integer sent most-significant octet first.

9.9 Structure and encoding of PN-parameters

The PN-parameters field specifies the current values of those DLE and PhE parameters necessary for a receiving DLE to configure itself and its associated PhE so that they can reply to a subsequent PN DLPDU. Once set, the values of these parameters can not be changed while the receiving DLE remains ONLINE, other than by reception of another PN DLPDU as specified in 12.1.3.

NOTE Theoretically, the current LAS DLE could change the link's parameters by sending a PN DLPDU to each of the other DLEs on the local link, forcing each of them OFFLINE, after which they would learn the new link parameters in the process of again coming ONLINE. But this DL-protocol does not specify a means of requesting this LAS action.

The PN-parameters field shall be composed of seven subfields (see Table 71 through Table 76):

Table 71 – Structure of PN-parameters

PN-parameters and the variable values transmitted						
PhL maximum inter-channel signal skew	Version	PhL post-transmission-gap extension units	PhL preamble extension units	Slot-time	Maximum-response-delay	Minimum-inter-DLPDU-delay
V(PhIS)	0VVV	V(PhGE)	V(PhPE)	V(ST)	V(MRD)	V(MID)
one quartet	one quartet	one quartet	one quartet	two octets	one octet	one octet

The PN-parameters field, which partitions as shown in Table 71, shall specify the parameters needed to reply to the PN DLPDU:

- a) The first two octets, ordered as shown in Table 72 and Table 73, specify the DL protocol version number and the PhL-parameters of the local link, defined in IEC 61158-2:2003³, Table 4, required for generating a reply to a received PN DLPDU:

Table 72 – PN-parameters: 1st octet

PhL maximum inter-channel signal-skew				zero	Version		
V(PhIS)				0	VVV		
7	6	5	4	3	2	1	0

- 1) a 4-bit subfield which specifies the required PhL maximum inter-channel signal skew, defined in IEC 61158-2:2003⁴, 6.2.2.2 and 8.2.4.2e), for the PhEs of the local link;
- 2) a 1-bit subfield, encoded as zero;
- 3) a 3-bit DL protocol version number (VVV), whose value is specified at the beginning of clause 9;

Table 73 – PN-parameters: 2nd octet

PhL post-transmission-gap extension units				PhL preamble extension units			
V(PhGE)				V(PhPE)			
7	6	5	4	3	2	1	0

- 4) a 4-bit subfield which specifies the required number of PhL post-transmission-gap extension units, defined in IEC 61158-2:2003⁵, 6.2.2.2 and 8.2.4.1d), for the PhEs of the local link;

NOTE Although this measure is named “post-transmission-gap-extension” in IEC 61158-2, it is a measure of the amount of observable non-transmission required between any two transmissions on the local medium, as observed from any point on that medium, whether the two transmissions are from the same source PhE or from two different source PhEs. Thus it could just as correctly be named “pre-transmission-gap-extension”, or even more correctly “inter-transmission-gap-extension”.

³ Expected date of concurrent publication.

⁴ Expected date of concurrent publication.

⁵ Expected date of concurrent publication.

- 5) a 4-bit subfield which specifies the required number of PhL preamble extension units, defined in IEC 61158-2:2003⁶, 6.2.2.2 and 8.2.4.1c), for the PhEs of the local link;
- b) The next four octets, ordered as shown in Table 74 through Table 76, specify the DLL-parameters for the local link required for generating a reply to a received PN DLPDU:

Table 74 – PN-parameters: 3rd and 4th octets

Slot-time															
V(ST)															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Table 75 – PN-parameters: 5th octet

Maximum-response-delay							
V(MRD)							
7	6	5	4	3	2	1	0

Table 76 – PN-parameters: 6th octet

Minimum-inter-DLPDU-delay							
V(MID)							
7	6	5	4	3	2	1	0

- 1) The fifth parameter subfield shall convey, in two octets transmitted most significant octet first, the current value of V(ST), slot-time, defined in 6.7.1.1.
- 2) The sixth parameter subfield shall convey, in one octet, the current value of V(MRD), maximum-response-delay, defined in 6.7.1.3.
- 3) The seventh parameter subfield shall convey, in one octet, the current value of V(MID), minimum-inter-DLPDU-delay, defined in 6.7.1.12.

9.10 Structure and encoding of DD-parameters

The DD-parameters (Delegation-Duration parameters) field shall be composed of one subfield (see Table 77):

Table 77 – Structure of DD-parameters

DD-parameters
delegated or requested duration
V(RD)
two octets

- a) The subfield shall specify the duration for which the token is being delegated or for which delegation is being requested. It is measured in octet-durations, and is encoded as a two-octet unsigned integer sent most-significant octet first. Its range is 0 to 65 000.

NOTE The minimum practical value for using this duration for a transmission is about 16. The value 0 is used only during maintenance of the local link's live-list V(LL).

⁶ Expected date of concurrent publication.

10 Type 1: DL-service elements of procedure

NOTE 1 The elements of procedure in this clause correspond generally to the highest DLL sublevel, as specified in 6.1.3.

Throughout this clause, whenever a conditional action is specified and the specified enabling condition does not occur, then the corresponding action also does not occur. The net effect is that events which do not meet any of the enabling conditions specified in a service procedure will have no consequential actions with respect to that specific service procedure.

Throughout this clause, the value $(V(NRC)+1)$ is used as a link-independent indication of the maximum number of times to perform a procedure — once initially, plus $V(NRC)$ repetitions — before concluding that an unrecoverable error situation exists and that alternate more drastic action is required. When the DLE can ascertain, from the involved DL(SAP)-addresses or DLCEP-addresses, that all of the DLEs involved in a DLC are local to a single link, then the DLE may substitute in a consistent manner the link-dependent value $(V(MRC)+1)$ for the link-independent value $(V(NRC)+1)$.

NOTE 2 Throughout this clause, a DLE receives a transmit opportunity, either by receipt of an appropriate immediate-reply opportunity or by receipt of an appropriate token.

10.1 Operation of the DL(SAP)-address, buffer and queue management services

The DL(SAP)-address, buffer and queue management services are the create and delete buffer or queue services, the bind and unbind DL(SAP)-address services, the put buffer service, and the get buffer or queue service.

10.1.1 Receipt of a DL-CREATE request primitive

When the DLE receives a DL-CREATE request, it shall

- a) allocate a buffer of the specified DLSDU size, or queue of the specified DLSDU size and maximum number of entries, and initialize it to empty;
NOTE Statistical allocation techniques using a shared multi-use storage area are permissible.
- b) assign a buffer-or-queue DL-identifier, and any provided DLS-user-identifier, to that buffer or queue;
- c) return that identifier to the DLS-user, together with a status of “success”.

Alternatively, an appropriate failure status shall be returned to the DLS-user.

10.1.2 Receipt of a DL-DELETE request primitive

When the DLE receives a DL-DELETE request, if the specified buffer or queue

- a) was created by the DLS-user and not by DL-management action;
- b) is not currently bound to any DLCEP or DL(SAP)-address

then the DLE shall delete the specified buffer or queue and return a status of “success”.

Otherwise the DLE shall return an appropriate failure status.

10.1.3 Receipt of a DL-BIND request primitive

When the DLE receives a DL-BIND request, it shall check

- a) that the specified DL(SAP)-address is a group DL-address not currently in use by the requesting DLS-user, or a DLSAP-address not currently in use by the DLE;
- b) that the specified link and node components of the DL(SAP)-address are compatible with the one or more link address components and one or more node address components assigned to the DLE (see 6.3);

- c) that any specified explicit bindings to DL-queues or DL-buffers are valid and do not conflict with other existing bindings to those queues or buffers;
- d) that any specified static or dynamic QoS attributes are valid and permitted by local DL-management.

Failure shall be reported to the DLS-user with an appropriate status. Otherwise

- 1) the DL(SAP)-address shall be associated to the requesting DLS-user, and to any provided DLS-user-identifier;
- 2) all explicitly-specified queues and buffers shall be bound as specified to the DL(SAP)-address;
- 3) the static and dynamic QoS attributes shall be bound to the DLSAP-address;
- 4) the specified DL(SAP)-address shall be activated for reception;
- 5) a DL(SAP)-address identifier shall be assigned to the DL(SAP)-address and returned to the DLS-user, together with a status of “success”;
- 6) if either
 - i) the DLE is a fractional-duty-cycle (FDC) DLE, or
 - ii) the DL(SAP)-address has a link-designator component (see 6.3.2.1) whose value specifies a flat non-local DL-address (between 0001₁₆ and 0FFF₁₆, inclusive), and the variable V(TL) has a non-zero value,

then the DLE shall send a DL-address report SPDU as specified in 11.3.6.3, with an appropriate reason for the report SPDU and with a request identifier of zero, to the DL-support functions of all bridges on the local link (see 6.3.3.2).

10.1.4 Receipt of a DL-UNBIND request primitive

When the DLE receives a DL-UNBIND request,

- a) if the specified DL(SAP)-address currently is not bound to the requesting DLS-user, or if the specified DL(SAP)-address was bound by DL-management action, then the DLE shall ignore the request primitive;
- b) otherwise the DLE shall
 - disconnect any DLCEPs associated with the DL(SAP)-address;
 - confirm with an appropriate error status any unconfirmed connection-oriented or connectionless-mode service requests outstanding at that DL(SAP)-address or its DLCEPs;
 - unbind all queues and buffers from the specified DL-address;
 - disassociate the specified DL-address from the requesting DLS-user.

If the DLE has no other DLS-users associated with that DL-address (which is always the case for DLSAP-addresses) then the DLE shall deactivate reception of that DL-address, in which case if either

- 1) the DLE is a fractional-duty-cycle (FDC) DLE, or
- 2) the DL(SAP)-address has a link-designator component (see 6.3.2.1) whose value specifies a flat non-local DL-address (between 0001₁₆ and 0FFF₁₆, inclusive), and the variable V(TL) has a non-zero value,

then the DLE shall send a DL-address report SPDU as specified in 11.3.6.3, with an appropriate reason for the report SPDU and with a request identifier of zero, to the DL-support functions of all bridges on the local link (see 6.3.3.2).

10.1.5 Receipt of a DL-PUT request primitive

When the DLE receives a DL-PUT request for a buffer

- a) which is associated with the requesting DLS-user;
- b) which is not bound as a receiving buffer
 - 1) to a peer or subscriber DLCEP, or
 - 2) to a DLSAP-address whose DL(SAP)-role is INITIATOR or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER;

then

- c) if a DLSDU is presented, and the size of the DLSDU is less than or equal to the buffer-size, then the DLE shall
 - 1) set the contents of the buffer equal to the presented DLSDU;
 - 2) indicate for each DLCEP which has a sending binding to the buffer that the buffer contains a new DLSDU;
 - 3) set the timeliness-status of buffer-writing, $V_B(TS)$ (see 6.7.4.21), to the DLS-user-specified timeliness, or FALSE if the user did not specify timeliness;
 - 4) if the timeliness-status of buffer-writing is TRUE, then set the variable time-of-last-buffer-write, $V_B(TW)$ (see 6.7.4.19), associated with the buffer to the current DL-time;
 - 5) if the timeliness-status of buffer-writing is TRUE, then set the time-of-production associated with the buffer, $V_B(TP)$ (see 6.7.4.20), to the current DL-time;
 - 6) return a status of "success";
- d) else if c) does not apply, and no DLSDU is being presented (that is, the buffer is being set empty), and the buffer is bound to a DLSAP-address, then the DLE shall set the buffer empty and shall return a status of "success";
- e) else if neither c) nor d) applies, then the DLE shall return an appropriate failure status.

Any ongoing accesses to the contents of a buffer or to its associated timeliness information, which are incomplete at the time of a DL-PUT request, shall not be affected by the DL-PUT request.

NOTE This constraint ensures that each access to a buffer is logically atomic.

10.1.6 Receipt of a DL-GET request primitive

- a) If the DLE receives a DL-GET request for a buffer which is associated with the requesting DLS-user, then
 - 1) if the buffer was written by a DLCEP which specified RESIDENCE timeliness or UPDATE timeliness or SYNCHRONIZED timeliness or TRANSPARENT timeliness, then the DLE shall evaluate the associated timeliness criteria as specified in 10.1.7, using the current DL-time as the time of buffer readout, and shall return the result as the local-DLE-timeliness attribute of the request;
 - 2) otherwise, if 1) does not apply, then the DLE shall return the value FALSE as the local-DLE-timeliness attribute of the request;
 - 3) the DLE shall return the timeliness-status, $V_B(TS)$ (see 6.7.4.21), associated with the buffer's writing as the sender-and-remote-DLE-timeliness attribute (see 10.2.2.5.3) of the request.

NOTE This sender-and-remote timeliness attribute always has the value FALSE when the buffer was written by an instance of the unitdata exchange service.
 - 4) If the sender-and-remote-DLE-timeliness attribute associated with the buffer is TRUE, and the buffer was written by a DLCEP which provides DL-time-of-production, then the DLE shall return the DL-time-of-production associated with the buffer, $V_B(TP)$ (see 6.7.4.20).

- 5) The DLE shall return the current contents of the buffer, with a status of “success” if the buffer is non-empty, and a status of “possible failure — buffer empty” if the buffer is empty.
- 6) If the buffer is a non-retentive buffer (BUFFER-NR), then the buffer shall be set empty.
- 7) Reading a buffer and its associated timeliness information shall be logically atomic with respect to writing the buffer.

NOTE When an implementation provides access to a buffer for an extended period of time during buffer read or write, then in the worst case this atomicity restriction requires that separate copies of the buffer's contents and timeliness information be provided for each reader and for the one writer, in addition to the actual buffer with its contents and timeliness. Then each reader may be in the middle of an extended access to a different epoch of the buffer's contents, and the writer may be writing a tentative buffer during reception, which will become the current buffer only if a Ph-error or FCS error is not detected before the end of the reception process.

- b) If the DLE receives a DL-GET request for a DLS-user specified queue which is
- associated with the requesting DLS-user;
 - not bound as a sending queue either to a DLCEP or to a DLSAP-address;
 - non-empty

then the DLE shall

- 1) return the next DLSDU contained in the queue, together with the called DLC identifier associated with that DLSDU, or DLL priority and called and calling DL(SAP)-addresses or DL(SAP)-address-identifiers associated with that DLSDU;
- 2) return local- and remote-DLE-timeliness attributes of FALSE;
- 3) remove that DLSDU from the queue;
- 4) return a status of “success”.

Otherwise the DLE shall return an appropriate failure status.

10.1.7 Computation of DL-timeliness

The DLE shall compute the intrinsic DL-timeliness for the buffer based on the type of buffer access — writing or reading — and the corresponding type of DL-timeliness which was specified on the corresponding local DL-CONNECT request or response primitive, as follows, where $P_C(NP.\Delta T)$ is the appropriate time-window-size specified in that request or response primitive:

a) RESIDENCE

DL-timeliness $\equiv$ TRUE when $0 \leq (\text{current-DL-time} - V_B(TW))$
 and $(\text{current-DL-time} - V_B(TW)) \leq P_C(NP.\Delta T)$;
 DL-timeliness $\equiv$ FALSE otherwise.

b) UPDATE

DL-timeliness $\equiv$ TRUE when $0 \leq (V_B(TW) - V_C(TNA))$
 and $(V_B(TW) - V_C(TNA)) \leq P_C(NP.\Delta T)$;
 DL-timeliness $\equiv$ FALSE otherwise.

c) **SYNCHRONIZED**

DL-timeliness $\equiv$ TRUE when $0 \leq (V_B(TW) - V_C(TNA))$
 and $(V_B(TW) - V_C(TNA)) \leq (\text{current-DL-time} - V_C(TNA))$
 and $(\text{current-DL-time} - V_C(TNA)) \leq P_C(NP.\Delta T)$;
 DL-timeliness $\equiv$ FALSE otherwise.

d) **TRANSPARENT**

DL-timeliness $\equiv$ TRUE.

e) **NONE**

DL-timeliness $\equiv$ FALSE.

10.2 Operation of the connection-mode services

The connection-mode services are the DLCEP establishment and DLCEP release services, the DLC data transfer and DLCEP reset service, and the DLC subscriber query service.

IEC 61158-3:2003⁷, Figure 16, shows a conceptual state transition diagram for sequences of DLC service primitives at a DLCEP. A corresponding state transition diagram for this DL-protocol is shown in Figure 13 and in Annex B, where similar states have identical state numbers.

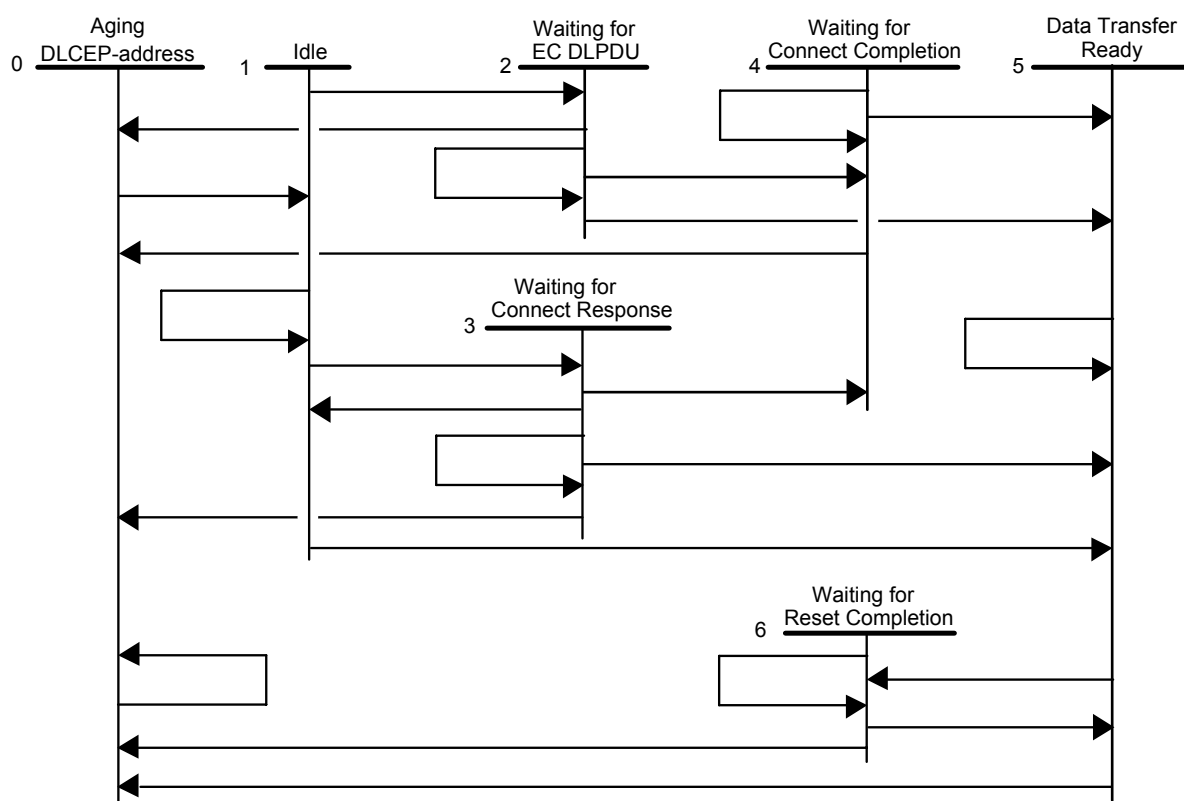


Figure 13 – State transition diagram for a DLCEP

⁷ Expected date of concurrent publication.

10.2.1 Operation of the DLCEP establishment and DLCEP release services

The primitives of the DLCEP-establishment service are DL-CONNECT request, indication, response and confirm, and DL-CONNECTION-ESTABLISHED indication. The primitives of the DLCEP-release service are DL-DISCONNECT request and indication. The associated local DLS-user, or the DLCEP's remote peer or publisher DLS-user, or the DLS provider, may invoke the DLCEP-release service any time after DLCEP establishment has begun.

During DLCEP establishment and release, if the DLCEP is a peer or publisher DLCEP, and the DLCEP-address used by the DLCEP is not sent as a source DL-address in a DC or EC DLPDU, and if either

- a) the DLE is a fractional-duty-cycle (FDC) DLE, or
- b) the DLCEP-address has a link-designator component (see 6.3.2.1) whose value specifies a flat non-local DL-address (between 0001₁₆ and 0FFF₁₆, inclusive), and the variable V(TL) has a non-zero value,

then the DLE shall send a DL-address report SPDU as specified in 11.3.6.3, including the specified DLCEP-address, with an appropriate reason for the report SPDU and with a request identifier of zero, to the DL-support functions of all bridges on the local link (see 6.3.3.2).

10.2.1.1 DLC negotiation rules

The DLS-user-visible aspects of the DLC negotiation rules are specified in IEC 61158-3. Additional negotiation rules which do not impact DLS-user visible aspects of the DLC are permitted to be specified in 9.1 of this standard. In case of apparent conflict, the rules specified in this subclause take precedence over any which may be specified in 9.1, which in turn take precedence over those specified in IEC 61158-3.

NOTE Where a subscribing DLS-user which is attempting to join an existing DLC requests DLC attributes which are not provided by the attributes of that existing DLC, then the publishing DLS-user may either

- A) indicate its inability to provide the requested attributes by replying with a DL-DISCONNECT request primitive to the proposed subscriber,
 - B) create a new DLC with the desired attributes and connect the subscriber to that DLC, or
 - C) modify the existing DLC to include the requested characteristics, when permitted by this part of IEC 61158.
- a) If the publisher, or either peer, of a DLC specifies a DLPDU-authentication attribute of MAXIMAL, then

NOTE DLPDU-authentication of MAXIMAL is provided primarily for use in safety systems. For this reason it maximizes the amount of state information exchanged in each DLPDU sent on the DLC and prohibits two-way-user data exchange in a single transaction, centralized schedule execution and other activities in which multiple DLEs need to have consistent state information.

- 1) each DLPDU sent from each DLCEP of the DLC shall contain the maximum permitted number of explicit addresses;
- 2) the EC-parameters in each EC DLPDU shall be constrained as follows:

- i) the address-size subfield (SS) shall specify LONG;
 - ii) the DLPDU-authentication subfield (XX) shall specify MAXIMAL;
 - iii) the residual-activity subfield (A) shall specify TRUE in the publisher-to-subscriber direction, or in all sending peer-to-peer directions, of data transfer;
- NOTE** Residual activity is not meaningful in the subscribers-to-publisher direction.
- iv) both two-way data exchange subfields (E) shall specify FALSE;
 - v) SD-parameter format B (subfield FFF), and time-stamp formats K and L (subfield HH), shall not be requested or used in either direction on the DLC;

- 3) if a DL-SCHEDULE-SEQUENCE request specifies a sequence in which one or more sequence elements specify the local or remote DLCEP of this DLC, then that sequence may be either locally or centrally scheduled, but shall be locally executed (see 10.4.3.1).
- b) If a) does not apply then
 - i) if the publisher, or either peer, of a DLC specifies a DLPDU-authentication attribute of SOURCE, then the DLPDU-authentication subfield (XX) in the EC-parameters shall specify SOURCE, and each DLPDU sent from each DLCEP of the DLC shall contain the maximum permitted number of explicit addresses;
 - ii) if a subscriber of a DLC specifies a DLPDU-authentication attribute of MAXIMAL in a DL-CONNECT request primitive, then the DLPDU-authentication subfield (XX) in the resulting EC-parameters DLPDU shall specify MAXIMAL. If the requested DLC was already established, then
 - A) if that DLC was not established with DLPDU-authentication attribute of MAXIMAL then the publishing DLE shall reject the connection establishment request from that subscriber;
 - B) otherwise, when A) does not apply, then the publishing DLE shall attempt to add that subscriber to the existing DLC;
 - iii) if a subscriber of a DLC specifies a DLPDU-authentication attribute of SOURCE in a DL-CONNECT request primitive, then the DLPDU-authentication subfield (XX) in the resulting EC-parameters DLPDU shall specify SOURCE. If the requested DLC was already established, then
 - A) if that DLC was established with a DLPDU-authentication attribute of ORDINARY then the publisher's DLE shall change the DLPDU-authentication to SOURCE and each DLPDU sent from each DLCEP of the DLC shall thereafter contain the maximum permitted number of explicit addresses;
 - B) otherwise, when A) does not apply, then the publishing DLE shall attempt to add that subscriber to the existing DLC;
 - iv) else if none of i) to iii) applies, then the DLPDU-authentication subfield (XX) in the EC-parameters shall specify ORDINARY, and each DLPDU sent from each DLCEP of the DLC shall contain the minimum permitted number of explicit addresses;
- c) The DLL path-diversity subfield (Q) of the EC-parameters shall specify ANY-PATH as a default. Negotiation of this subfield is from ANY-PATH to THIS-PATH.

NOTE See the note in 9.1a)3) for use of the value THIS-PATH and choice of the actual path in the THIS-PATH case.
- d) The address-size subfield of the EC-parameters shall be determined as follows.
 - 1) If required by a)2)i), or if any of the DL-addresses of the EC DLPDU have only a LONG representation, then the address-size subfield of the EC-parameters shall specify LONG.

NOTE A DL-address has a SHORT representation when there is an equivalent two-octet DL-address (see 7.2.1.1).
 - 2) Else, when 1) does not apply, and either b)i) applies or any member of the DLC is a fractional-duty-cycle (FDC) DLE, then the address-size subfield of the EC-parameters shall specify SHORT.
 - 3) Otherwise, when 1) and 2) do not apply, the address-size subfield of the EC-parameters shall specify VERY-SHORT.

NOTE The address-size VERY-SHORT applies only to DT DLPDUs sent using a reply token (see 8.7.1 formats 4 and 5); in all other cases the address-size SHORT is actually used.

- e) The DLCEP-data-delivery-features subfield (TT) of the EC-parameters shall specify, independently for each direction of the DLC, the provided data-delivery features, as specified in IEC 61158-3, except that the value NONE shall be replaced by UNORDERED with a maximum window size (WWWW) of zero and maximum-DLSDU-size subfield (M...M) of zero in the corresponding direction, indicating a simplex DLC.
- f) The residual-activity subfield (A) of the EC-parameters shall specify TRUE in a publisher-to-subscriber or sending peer-to-peer direction of data transfer when so required by a)2)iii), or by DL-management, or by a publishing or peer DLS-user, and shall specify FALSE otherwise. Negotiation of this subfield is from FALSE to TRUE.
- g) Window size negotiation occurs independently for each direction of the DLC. The actual maximum window size for a given direction of transmission shall be the smaller of the sender's maximum window size and the receiver's maximum window size in that direction, and the maximum-window-size subfield (WWWW) of the EC-parameters shall specify zero only when the maximum-DLSDU-size subfield (M...M) in the same direction is zero, indicating a simplex DLC.
- h) The SD-parameter-format subfield (FFF) of the EC-parameters shall specify the negotiated format for each direction of data transmission. The initial formats for the two directions of data transmission shall be chosen to meet the following constraints.

NOTE 1 These constraints are imposed by the DLC protocol and selected service features.

- 1) If a) applies, then format B shall not be chosen for either direction.
- 2) If the DLCEP-data-delivery-features subfield (TT) in either direction specifies ORDERED, DISORDERED, or CLASSICAL, then
 - i) if the maximum-window-size subfield (WWWW) in that direction specifies a value of four (4) or more, then format B shall not be chosen for that direction;
 - ii) if the DLCEP-class subfield (CC) specifies PEER, then the format chosen for the other direction shall contain both J and K subfields;
 - iii) if the DLCEP-class subfield (CC) specifies PUBLISHER, then the format chosen for that direction shall be format F or G, and the format chosen for the other direction shall be format C or D or E;
 - iv) if the DLCEP-class subfield (CC) specifies SUBSCRIBER, then the format chosen for that direction shall be format C or D or E, and the format chosen for the other direction shall be format F or G.

NOTE Format C is included in iii) and iv) only because support of formats D and E are not mandatory.

- 3) If a DLCEP-data-delivery-features subfield (TT) specifies ORDERED, DISORDERED, or CLASSICAL, then the format chosen for that direction shall contain an NDS subfield, and the format chosen for the other direction shall contain an NDR subfield.
- 4) If a maximum-DLSDU-size subfield (M...M) specifies a value greater than the amount of DLS-user data that can be conveyed in a single DLPDU of the DLC's priority, as determined by the DLC's DLL priority subfield (PP), then the format chosen for that direction shall contain TNS and ASN subfields, and the format chosen for the other direction shall contain an RSN subfield.
- 5) If a timeliness-included subfield (G) specifies TRUE, then the format chosen for that direction shall contain a T subfield.
- 6) The format chosen shall be the shortest (fewest octets in length) possible of those which the DLE supports and which meets all of constraints 1) to 5).

NOTE 2 The expected set of formats in the sending direction are as follows.

- a) Formats A, and potentially E, from a PEER DLCEP when the sending direction's DLCEP-data-delivery-features subfield (TT) specifies UNORDERED, depending on the corresponding DLCEP-data-delivery-features subfield (TT) in the other direction;
- b) Formats B, C and D, and potentially F and G, from a PEER DLCEP when the sending direction's DLCEP-data-delivery-features subfield (TT) specifies ORDERED, DISORDERED, or CLASSICAL, depending on the corresponding DLCEP-data-delivery-features subfield (TT) in the other direction;
- c) Formats A, F and G from a PUBLISHER DLCEP;

- d) Formats A, C, D, E and G from a SUBSCRIBER DLCEP, where formats D and G are chosen when subscriber-to-publisher DLSDUs have associated timeliness, and format C is chosen when format D would otherwise be chosen but is unavailable because its support is not mandatory. For formats C, D and G, the publishing DLE will ignore the NDS and K subfields within the SD-parameters of received CA, CD, DT and ED DLPDUs.
- i) The 2-way-data-exchange subfield (E) shall specify FALSE when the maximum-DLSDU-size subfields (M...M) is zero in one direction of data transmission, or when requested by a peer DLE during negotiation of a PEER DLC, or when required by a)2)iv) or by local DL-management or by DLE construction; and shall specify TRUE otherwise. Negotiation of this subfield is from TRUE to FALSE.
- j) Timeliness attributes of the DLCEP are communicated but not negotiated:
- 1) The timeliness-included subfield (G) of the EC-parameters shall specify FALSE when the specified sender-timeliness is NONE, and shall specify TRUE otherwise.
 - 2) The time-stamp-format subfield (HH) of the EC-parameters shall specify
 - i) format J when there is no sender timeliness, or the specified sender-timeliness is NONE, or time-of-production is not requested;
 - ii) format K when there is sender timeliness, and time-of-production is requested, and the timeliness-class is RESIDENCE or SYNCHRONIZED, and the associated time-window-size (ΔT) is 1 s or less;
 - iii) format L when there is sender timeliness, and time-of-production is requested, and the timeliness-class is RESIDENCE or SYNCHRONIZED, and the associated time-window-size (ΔT) is greater than 1 s;
 - iv) format M when there is sender timeliness, and time-of-production is requested, and the timeliness-class is UPDATE or TRANSPARENT.
- k) If one direction of data communication is not required for the DLC, because the DLS-user-specified data delivery features for that direction specified NONE, then in that direction
- 1) the residual-activity subfield (A) shall be specified as FALSE;
 - 2) the Queue/Buffer (B) subfield shall be specified as QUEUE;
 - 3) the timeliness subfield (G) shall be specified as FALSE;
 - 4) the time-stamp-format subfield (HH) shall be specified as FORMAT J.

10.2.1.2 Receipt of a DL-CONNECT request primitive

When the DLE receives a DL-CONNECT request primitive from a DLS-user, the DLE shall perform the following series of actions, and if any error is detected during the process, then the DLCEP shall be disconnected as specified in 10.2.1.8.

- a) The DLE shall assign a new DLCEP-identifier, and the provided DLS-user-identifier, to the DLCEP which may result from the request, and provide that DLCEP-identifier to the DLS-user as the single output parameter of the request.
- b) The DLE shall create and start a user-request timer $T_U(\text{MCD})$ with a duration based on the user-specified maximum confirm delay for DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be $\frac{V_C(\text{NP}).\text{MCD}_{\text{CRS}}}{V(\text{NRC})+1}$; otherwise the duration should be $\frac{60 \text{ s}}{V(\text{NRC})+1}$. DL-management may override these preferred durations.

NOTE See the permission at the start of clause 10 with regard to use of $V(\text{NRC})$ versus $V(\text{MRC})$.

- c) The DLE shall validate the calling-DLSAP-address or DLCEP-identifier provided by the DLS-user; if invalid, the DLE shall reject the DL-CONNECT request with a DL-DISCONNECT indication.

NOTE The structure and value ranges of a DLSAP-address are specified in 6.3.

- d) The DLE shall validate the self-consistency of the requested QoS parameter set, where all static and non-specified dynamic parameters assume the default values associated with the calling-DLSAP-address, and where the following automatic adjustments to that QoS occur.

- 1) Where any parameter is in violation of a local DL-management-imposed limit, then that parameter shall be set equal to that limit, if permitted by the negotiation rules specified in 10.2.1.1, or the DLE shall reject the DL-CONNECT request with a DL-DISCONNECT indication.
 - 2) If any maximum DLSDU size equals zero or the sending DLCEP data delivery features specify NONE, then the corresponding sending DLCEP data delivery features shall be set to UNORDERED.
NOTE This special case is not considered to be a violation of the negotiation rules of 10.2.1.1.
 - 3) if any maximum DLSDU size is greater than the maximum which can be conveyed in a single DLPDU of the specified DLL priority (see 6.2.3), and the sending DLCEP data delivery features in the corresponding direction are UNORDERED, then those sending DLCEP data delivery features shall be set to ORDERED.
- e) if the calling address identifier is a DLCEP-identifier for an existing DLCEP, then
- 1) If the existing DLCEP is a publisher DLCEP, then the DLE shall
 - i) use the specified DLCEP's local DLSAP-address as the calling-DLSAP-address for this new DL-CONNECT request;
 - ii) set each QoS parameter equal to the corresponding parameter of the specified DLCEP, if permitted by the negotiation rules of 10.2.1.1, or the proposed DL-CONNECT request shall be rejected with a DL-DISCONNECT indication specifying "connection rejection — QoS not available, permanent condition", terminating processing of the DL-CONNECT request;
 - iii) use the specified DLCEP's publisher-DLCEP-address reuse-discriminator in the EC-parameter;
 - iv) consider the new DL-CONNECT request to be a request for adding new subscribers to the existing DLC and continue with j).
 - 2) If the existing DLCEP's DLCEP-class is PEER or SUBSCRIBER, then the DLE shall reject the DL-CONNECT request with a DL-DISCONNECT indication.
- f) Otherwise, if e) does not apply, then the DLE shall determine the maximum send and receive window sizes based on the respective buffer-and-queue bindings, as follows
- 1) If the DLCEP-features are NONE, then the corresponding window size shall be zero (0).
 - 2) Otherwise, if 1) does not apply, then
 - i) if the binding was to a BUFFER, then the corresponding window size shall be one (1);
 - ii) if the binding was to a QUEUE-K, then the corresponding window size shall be the smaller of K or 15;
 - iii) if the default binding was used, then the corresponding window size shall be at least one (1);
 - iv) in all cases, DL-management can further constrain this window size.
- g) If the optional calling-DLCEP-address was specified in the request primitive, then the DLE shall assign that DLCEP-address to the DLCEP; if any conflicting assignment is detected, then the DLCEP shall be terminated as specified in 10.2.1.8.
- h) Otherwise, if g) does not apply, and if the called address is not a DLCEP-address presumed to be for a publisher DLCEP, then a DLCEP-address allocated to the DLE and not currently in use shall be assigned to the DLCEP; where possible this shall be a SHORT DL-address of the local link.

When the called address is a DLCEP-address presumed to be for a publisher DLCEP, then this assignment is optional.

- i) The DLC shall initialize the DLCEP's $V_C(NP)$, $V_C(N)$, $V_C(R)$, $V_C(A)$, $V_C(M)$, $V_C(MS)$, $V_C(H)$, $V_C(HS)$ and $V_C(L)$ variables as specified in 6.7.4.

- j) The DLE shall encode an EC DLPDU as specified in 8.1 and 9.1:
 - 1) If the called-DL-address parameter specifies a DL(SAP)-address or DLCEP-address, then the DLE shall form an EC DLPDU with three addresses, whose values shall be, respectively,
 - i) the called-DL(SAP)-address or DLCEP-address;
 - ii) the DLCEP-address assigned to the DLCEP, or calling-DLSAP-address if no such assignment was done as in h);
 - iii) the calling-DLSAP-address.
 - 2) If the called-DL(SAP)-address parameter specifies UNKNOWN, then
 - i) if the DLS-user did not specify a calling-DLCEP-address, then the DLE shall reject the DL-CONNECT request with a DL-DISCONNECT indication;
 - ii) else if i) does not apply, and the DLCEP-class is PUBLISHER, then the DLE shall form an EC DLPDU with two addresses, whose values shall be the DLCEP-address assigned to the DLCEP, and the calling-DLSAP-address, respectively;
 - iii) else if i) does not apply, and the DLCEP-class is PEER or SUBSCRIBER, then the DLE shall not form and send an EC DLPDU, but shall await the receipt of a complementary EC DLPDU from the remote PEER or PUBLISHER.
- k) If the DLCEP class of the DLE is to be either PEER or SUBSCRIBER, then
 - 1) if an EC DLPDU was formed, then
 - i) the DLE shall set the reply-requested field in the EC-parameters in the DLPDU;
 - ii) the DLE shall queue the DLPDU at TIME-AVAILABLE priority as specified in 10.4.5.
 - 2) the DLE shall activate recognition of the DLCEP's local DLCEP-address and change the DLCEP state, $V_C(ST)$, to WAITING-FOR-EC-DLPDU.
- l) If the sending DLCEP class of the DLE is to be PUBLISHER, then
 - 1) the DLE shall clear the reply-requested field in the EC-parameters in the DLPDU;
 - 2) if the source DLCEP-address is not that of an existing DLCEP, the DLE shall assign a new value to the publisher-DLCEP-address reuse-discriminator subfield (NNN) of the EC-parameters (see 9.1a)2)):
 - i) if the DLE is capable of recording the publisher-DLCEP-address reuse-discriminator between DLCEP incarnations, then it should maximize the interval between reuse of the same discriminator value;
 - ii) otherwise, when i) does not apply, the DLE shall choose the discriminator value randomly with an approximately uniform distribution;
 - 3) the DLE shall queue the DLPDU at TIME-AVAILABLE priority as specified in 10.4.5;
 - 4) the DLE shall issue the DL-CONNECT confirm for the DLCEP immediately after transmission of the EC DLPDU;
 - 5) the DLE shall cancel the user-request timer $T_U(MCD)$;
 - 6) if e) did not apply, then the DLE shall activate recognition of the DLCEP's local DLCEP-address and change the DLCEP state, $V_C(ST)$, to DATA-TRANSFER-READY.

10.2.1.3 Receipt of a DL-CONNECT response primitive

When the DLE receives a DL-CONNECT response primitive from a DLS-user, the DLE shall perform the following series of actions; if any error is detected during the process, then the DLCEP shall be disconnected as specified in 10.2.1.8.

- a) The DLE shall validate the DLCEP-identifier, and the responding DLSAP-address or DLCEP-identifier, provided by the DLS-user, and shall associate the provided DLS-user-identifier with the DLCEP.
- b) If the identified DLCEP is not in the WAITING-FOR-CONNECT-RESPONSE state, the DLCEP shall be disconnected.
- c) The DLE shall validate the self-consistency of the response QoS parameter set, where all static and non-specified parameters assume their default values associated with the responding-DLSAP-address, and where the automatic adjustments to that QoS specified in 10.2.1.2d) occur. The DLE shall then validate the consistency of the resulting QoS parameter set with the corresponding parameters from the received EC DLPDU, and the adherence to the rules of parameter negotiation specified in 10.2.1.1.
- d) If the responding address identifier in the DL-CONNECT response was for a DLCEP, then the DLE shall consider the new DL-CONNECT response as requesting a merger of the response's attempted DLC with an existing DLC, after which the DLCEP-identifier which was specified in the associated DL-CONNECT indication will no longer be valid and any DLS-provider state information (including publisher-DLCEP-address reuse-discriminator) related to the indication's attempted DLC shall be discarded:
 - 1) If the existing DLC is a multi-peer DLC, then the DLE shall consider the DL-CONNECT response to be a request for adding a new subscriber to the existing DLC:
 - i) If the specified DLCEP is a publisher DLCEP, then the DLE shall
 - A) use the specified DLCEP's local DLCEP-address and DLSAP-address as the responding DLCEP-address and DLSAP-address for this DL-CONNECT response;
 - B) set each QoS parameter, and the publisher-DLCEP-address reuse-discriminator, equal to the corresponding parameter of the specified DLCEP, if permitted by the negotiation rules of 10.2.1.1;
 - C) if necessitated by the rule of 10.2.1.1d), then change the address size of the existing DLC from VERY-SHORT to SHORT or from SHORT to LONG.
 - ii) If no negotiation-rule violation is detected, then the DLE shall
 - A) encode an EC DLPDU not requesting a reply, with three addresses as specified in 8.1 and 9.1, which are, respectively,
 - the first of the two source DL-addresses from the received EC DLPDU which resulted in the DL-CONNECT indication and its consequent DL-CONNECT response;
 - the DLCEP-address of the existing DLC;
 - the DLSAP-address associated with this responding DLCEP-address, respectively;
 - B) queue the DLPDU at TIME-AVAILABLE priority as specified in 10.4.5;
 - C) stop the timer which was started in 10.2.1.4.2b)4)iv).
 - iii) Otherwise, when i) does not apply because the specified DLCEP is a subscriber DLCEP, or when ii) does not apply because a negotiation rule violation was detected, then the DLE shall reject the received DLC-establishment request and terminate processing of the received EC DLPDU, as follows:

If the destination DL-address of the received EC DLPDU was not a group DL-address, then

 - A) The DLE shall encode a DC DLPDU as specified in 8.2 and , with a reason of "connection rejection — QoS not available, permanent condition", and schedule the DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5.
 - B) The DC DLPDU shall have both destination and source addresses (see 8.2.1 formats 1L and 1s), the destination address shall be identical to the first source DLCEP-address of the received EC DLPDU, which resulted in the DL-CONNECT indication and its consequent DL-CONNECT response, and the

source address shall be identical to the destination DL-address of that received EC DLPDU.

- 2) If the existing DLCEP's DLCEP-class is PEER, then the DLE shall consider the DL-CONNECT response to be a request for resolving a DL-CONNECT request collision on a peer DLC which is in the WAITING-FOR-EC-DLPDU state. The DLE shall apply the negotiation rules of 10.2.1.1 jointly to the QoS parameters of the two DLCEPs and shall reflect the results in the QoS of the originally-requested DLCEP.

If a negotiation rule violation is detected, or if the DLCEP specified by the responding-address is not in the WAITING-FOR-EC-DLPDU state, then both DLCEPs shall be disconnected as specified in 10.2.1.8, but with a reason of "connection rejection — QoS not available, permanent condition".

Otherwise, when no negotiation-rule violation was detected, then the DLE shall

- i) use the source DLCEP-address and DLSAP-address, of the received EC DLPDU, which resulted in the DL-CONNECT indication and its consequent DL-CONNECT response, as the remote DLCEP-address and DLSAP-address of the DLCEP which was in the WAITING-FOR-EC-DLPDU state;
 - ii) consider the reply-requested field of the received EC DLPDU, which resulted in the DL-CONNECT indication and its consequent DL-CONNECT response, to have been set;
 - iii) cause the DLCEP specified as the responding-address to
 - send a three-address EC DLPDU specifying that a reply is not requested;
 - stop the timer which was started in 10.2.1.4.2b)4)iv);
 - start a timer as in 10.2.1.2b), with a duration equal to the value for the maximum confirm delay on DL-CONNECT as specified in the DL-CONNECT response primitive;
 - activate recognition of the DLCEP's local DLCEP-address;
 - change the specified DLCEP's state, $V_C(ST)$, to WAITING-FOR-CONNECT-COMPLETION.
- e) If the responding address identifier in the DL-CONNECT response was a DLSAP-address, then
- that DLSAP-address shall be used as the local DLSAP-address;
 - the DLE shall determine the local maximum send and receive window sizes based on the respective buffer-and-queue bindings, possibly further restricted by DL-management, as specified in 10.2.1.2.

The DLE shall then determine the actual maximum send window size as the smaller of the local send window size and the received EC DLPDU's receive window size, and the actual maximum receive window size as the smaller of the local receive window size and the received EC DLPDU's send window size, as specified in 10.2.1.1. The DLE also shall perform all other required negotiations, as specified in 10.2.1.1.

- f) If the optional DLCEP-address was specified in the response primitive, then the DLE shall assign that DLCEP-address to the DLCEP; if any conflicting assignment is detected, then the DLCEP shall be disconnected as specified in 10.2.1.8 with a reason of "disconnection — incorrect DLCEP pairing, permanent condition". Otherwise a DLCEP-address not currently in use shall be assigned to the DLCEP; where possible this shall be a SHORT DL-address of the local link.

When the DLE is serving only as a subscriber in the DLC, then no reply EC DLPDU is permitted, and so no assignment of a DLCEP-address is required.

NOTE After DLCEP establishment is completed, a subscriber substitutes its calling-DLSAP-address where a sending DLCEP-address otherwise would be required in a CA, CD, ED or DT DLPDU.

- g) If the responding DLCEP class is SUBSCRIBER, then the DLE shall

- accept the next received sequence number of the DLC as the sequence number of the first DLSDU;
 - stop the timer which was started in 10.2.1.4.2b)iv);
 - issue a DL-CONNECTION-ESTABLISHED indication;
 - activate recognition of the DLCEP's remote (publisher) DLCEP-address;
 - change the DLCEP's state, $V_C(ST)$, to DATA-TRANSFER-READY.
- h) Otherwise, if g) does not apply, then the DLE shall
- 1) encode an EC DLPDU not requesting a reply, with three addresses as specified in 8.1 and 9.1, where its addresses are, respectively,
 - the first of the two source DL-addresses from the received EC DLPDU which resulted in the DL-CONNECT indication and its consequent DL-CONNECT response;
 - the DLCEP-address just assigned to the DLCEP;
 - the responding DLSAP-address, respectively;
 - 2) assign a publisher-DLCEP-address reuse-discriminator when the responding DLCEP-class is PUBLISHER;
 - 3) schedule the DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5.
- NOTE This procedure causes a publisher DLCEP to reply to an EC DLPDU from a proposed subscriber DLCEP by sending an EC DLPDU directly to that subscriber DLCEP's temporary calling DLCEP-address. Alternatively, the publishing DLE can create an independent publisher DLCEP (through use of a DL-CONNECT request) after receiving the DL-CONNECT indication from the proposed subscriber, and then merge the subscriber's requested publishing DLCEP into the one just established. This latter approach will cause an EC DLPDU to be sent to the publisher's Called DL(SAP)-address before the response EC DLPDU is sent to the requesting subscriber.
- i) If the responding DLCEP class is PUBLISHER, then the DLE shall
- stop the timer which was started in 10.2.1.4.2b)iv);
 - issue a DL-CONNECTION-ESTABLISHED indication;
 - activate recognition of the DLCEP's local DLCEP-address;
 - change the DLCEP's state, $V_C(ST)$, to DATA-TRANSFER-READY.
- j) If the responding DLCEP class is PEER, then the DLE shall
- stop the timer which was started in 10.2.1.4.2b)iv);
 - start a timer as in 10.2.1.2b), with a duration equal to the value for the maximum confirm delay on DL-CONNECT as specified in the DL-CONNECT response primitive;
 - activate recognition of the DLCEP's local DLCEP-address;
 - change the DLCEP's state, $V_C(ST)$, to WAITING-FOR-CONNECT-COMPLETION.

10.2.1.4 Receipt of an EC DLPDU

When the DLE receives an EC DLPDU, the DLE shall determine the version number of the DL-protocol in use, as specified in the received EC DLPDU, and shall interpret the other EC-parameters of the DLPDU accordingly.

10.2.1.4.1 Receipt of an EC DLPDU with two addresses

The DLE shall perform the following series of actions, and if any error is detected during the process, then the DLC shall be disconnected as specified in 10.2.1.8:

- a) If the first (source) address of the received EC DLPDU is a PUBLISHER DLCEP-address for an existing SUBSCRIBER DLCEP, and if that SUBSCRIBER DLCEP is in the WAITING-FOR-EC-DLPDU state, then the DLE shall
 - i) validate the received DLC parameters against those earlier requested for the SUBSCRIBER DLCEP (and possibly not sent in an EC DLPDU);

- ii) set the parameters of the existing DLCEP to equal the received DLC parameters, if permitted by the negotiation rules of 10.2.1.1.

If an error is detected, then the DLE shall disconnect the SUBSCRIBER DLCEP as specified in 10.2.1.8, with a reason of "connection rejection — QoS not available, permanent condition".

If no error is detected during this validation, then the DLE shall

- iii) set the publisher-DLCEP-address reuse-discriminator equal to that in the received EC DLPDU;
 - iv) cancel the user-request timer $T_U(\text{MCD})$;
 - v) accept the next received sequence number of the DLC as the sequence number of the first DLSDU;
 - vi) issue a DL-CONNECT confirm to the DLS-user;
 - vii) change the SUBSCRIBER DLCEP to the DATA-TRANSFER-READY state.
- b) When the first (source) address of the received EC DLPDU is a PUBLISHER DLCEP-address for an existing SUBSCRIBER DLCEP, but a) does not apply because the SUBSCRIBER DLCEP is not in the WAITING-FOR-EC-DLPDU state, then the DLE shall compare the publisher-DLCEP-address reuse-discriminator of the existing DLCEP with that in the received EC DLPDU. If the two values are equal then the DLE shall set the address size of the existing DLCEP to that in the received EC DLPDU.
- c) Otherwise, when a) and b) do not apply, then the existing SUBSCRIBER DLCEP shall be disconnected as specified in 10.2.1.8, with a reason of "disconnection — wrong publisher-DLCEP-address reuse-discriminator, permanent condition".

10.2.1.4.2 Receipt of an EC DLPDU with three addresses

The DLE shall perform the following series of actions, and if any error is detected during the process, then the DLC shall be disconnected as specified in 10.2.1.8:

- a) If the first address in the received EC DLPDU is a group-DL-address associated with more than one of the DLE's DLS-users, then the DLE shall treat each of those DLS-users as if that user had individually received the EC DLPDU. However, no DC DLPDU shall be sent as a direct response to the received EC DLPDU.
- b) If the first address of the received EC DLPDU is a DLSAP-address, then
 - 1) the DLE shall validate the self-consistency of the received EC DLPDU, where all static and non-specified dynamic parameters assume the default values associated with that called-DLSAP-address, and where any parameter in violation of a local DL-management-imposed limit shall be set equal to that limit, if permitted by the negotiation rules of 10.2.1.1, or the DLCEP shall be disconnected as specified in 10.2.1.8 with a reason of "connection rejection — QoS not available, permanent condition";
 - 2) the DLE shall check whether a DLS-user associated with that DLSAP-address has an active DLCEP whose remote DLCEP-address equals the source DLCEP-address specified in the received EC DLPDU;
 - 3) if such an active DLCEP exists, then
 - i) if
 - the DLCEP is in the WAITING-FOR-CONNECT-COMPLETION state;
 - the received EC DLPDU requests a reply
 then a return EC DLPDU, addressed to the first source DL-address specified in the received EC DLPDU, with source addresses equal to the DLCEP's local DLCEP- and DLSAP-addresses, and specifying the parameters of the active DLCEP, and not requesting a reply, shall be encoded and shall be queued at TIME-AVAILABLE priority as specified in 10.4.5;

- ii) else when i) does not apply, then the DLE shall disconnect the DLCEP as specified in 10.2.1.8 with a reason of "connection rejection — inconsistent DLCEP state, permanent condition";

NOTE This reconfirmation of the existing DLC does not cause a change in the responding DLCEP's state.

- 4) if no such active DLCEP exists, then the DLE shall assign a new DLCEP identifier to the DLCEP, and shall apply the negotiation rules of 10.2.1.2d). If any violation of the negotiation rules occurs, then the DLE shall disconnect the proposed DLCEP as specified in 10.2.1.8 with a reason of "connection rejection — QoS not available, permanent condition". If no violation is detected, then for each DLS-user associated with the DLSAP-address or group DL-address which was the first address of the received EC DLPDU, the DLE shall
 - i) create a DLCEP, initializing its $V_S(NP)$, $V_S(N)$, $V_S(R)$, $V_S(A)$, $V_S(M)$, $V_S(MS)$, $V_S(H)$, $V_S(HS)$ and $V_S(L)$ variables as specified in 6.7.4;
 - ii) record the source DLCEP-address and source DLSAP-address from the received EC DLPDU as the DLCEP's remote DLCEP-address and remote DLSAP-address, respectively and when sender's DLCEP class is PUBLISHER, also record the publisher-DLCEP-address reuse-discriminator of the EC DLPDU as the DLCEP's local publisher-DLCEP-address reuse-discriminator;
 - iii) report a DL-CONNECT indication to the DLS-user;
 - iv) start a timer to monitor for the DLS-user's response to the DL-CONNECT indication, as specified in 10.2.1.2b);
 - v) change the DLCEP state, $V_C(ST)$, to WAITING-FOR-CONNECT-RESPONSE.
- c) Else if the first address of the received EC DLPDU is a DLCEP-address for an existing DLCEP, and if the addressed DLCEP is in the WAITING-FOR-EC-DLPDU state, then the DLE shall validate the received DLC parameters, and the received DLPDU's source DL-addresses when their expected values are known, against those sent in an earlier EC DLPDU, and if an error is detected, then
 - 1) if the called address of the DLCEP was a group DL-address, then the DLE shall reply with a DC DLPDU not requesting a reply, addressed to the first source DL-address specified in the received EC DLPDU, with a source address equal to the DLCEP's local DLCEP-address, and shall otherwise ignore the received EC DLPDU.
 - 2) otherwise, when the called address of the DLCEP was a DLSAP-address, then the DLE shall disconnect the DLCEP as specified in 10.2.1.8 with a reason of "connection rejection — QoS not available, permanent condition".

If no error is detected during the validation of the received EC DLPDU, then

- 3) If the receiving DLCEP's DLCEP-class is PEER and the received EC DLPDU requests a reply, then
 - i) the two source DL-addresses of the received EC DLPDU shall be noted as the remote-DLCEP-address and remote-DLSAP-address of the DLCEP;
 - ii) a return EC DLPDU, addressed to the first source DL-address specified in the received EC DLPDU, with source addresses equal to the DLCEP's local DLCEP- and DLSAP-addresses, and specifying the parameters of the active DLCEP, and not requesting a reply, shall be encoded and shall be queued at TIME-AVAILABLE priority as specified in 10.4.5;
 - iii) the DLE shall start a timer as specified in 10.2.1.2b) with a duration equal to the value for the maximum confirm delay on DL-CONNECT as specified in the DL-CONNECT request primitive;
 - iv) the DLE shall change the state to WAITING-FOR-CONNECT-COMPLETION.

- 4) Else when the receiving DLCEP's DLCEP-class is SUBSCRIBER or the received EC DLPDU does not request a reply, then
- i) if the receiving DLCEP's DLCEP-class is PEER or SUBSCRIBER, then the two source DL-addresses of the received EC DLPDU shall be noted as the remote-DLCEP-address and remote-DLSAP-address of the DLCEP;
 - ii) if the DLCEP-class of the receiving DLCEP is PEER, then
 - a DT DLPDU not containing DLS-user data;
 - with a destination address equal to the first source DL-address specified in the received EC DLPDU;
 - when the DLCEP's attributes require the DLPDU to have a source address, with a source address equal to the DLCEP's local DLCEP-address
 shall be encoded and shall be queued at the DLCEP's priority as specified in 10.4.5, to notify the peer DLE of the successful receipt of the confirming EC DLPDU
 - iii) if the receiving DLCEP's DLCEP-class is SUBSCRIBER then the DLE shall accept the next received sequence number of the DLC as the sequence number of the first DLSDU;
 - iv) the DLE shall issue a DL-CONNECT confirm primitive, conveying the negotiated DLCEP-attributes, to the requesting DLS-user;
 - v) the DLE shall cancel the user-request timer $T_U(MCD)$ and change the DLCEP state to DATA-TRANSFER-READY.
- d) Else, if the first address of the received EC DLPDU is a DLCEP-address for an existing DLCEP, and if the addressed DLCEP is in the WAITING-FOR-CONNECT-COMPLETION state, then the DLE shall validate the received DLC parameters against those sent in an earlier EC DLPDU, and if an error is detected, then the DLE shall disconnect the DLCEP as specified in 10.2.1.8 with a reason of "connection rejection — QoS not available, permanent condition".

If no error is detected during this validation, then

- 1) if the received EC DLPDU requests a reply, then
 - i) a return EC DLPDU, addressed to the first source DL-address specified in the received EC DLPDU, with source addresses equal to the DLCEP's local DLCEP- and DLSAP-addresses, and specifying the parameters of the active DLCEP, and not requesting a reply, shall be encoded and shall be queued at TIME-AVAILABLE priority as specified in 10.4.5;
 - ii) the DLE shall restart the $T_U(MCD)$ timer with the same period as the previous time.
- 2) else, when 1) does not apply, then
 - i) a DT DLPDU not containing DLS-user data,
 - with a destination address equal to the first source DL-address specified in the received EC DLPDU;
 - if the DLCEP's attributes require the DLPDU to have a source address, with a source address equal to the DLCEP's local DLCEP-address
 shall be encoded and shall be queued at the DLCEP's priority as specified in 10.4.5, to notify the peer DLE of the successful receipt of the confirming EC DLPDU;
 - ii) if the state before WAITING-FOR-CONNECT-COMPLETION was WAITING-FOR-EC-DLPDU, then the DLE shall issue a DL-CONNECT confirm primitive, conveying the negotiated DLCEP-attributes, to the requesting DLS-user;
 - iii) else, when ii) does not apply, then the state before WAITING-FOR-CONNECT-COMPLETION was WAITING-FOR-CONNECT-RESPONSE, and the DLE shall issue a

- DL-CONNECTION-ESTABLISHED indication primitive to the responding DLS-user;
- iv) the DLE shall cancel the user-request timer $T_U(\text{MCD})$ and change the DLCEP state to DATA-TRANSFER-READY.
- e) Else if the first address of the received EC DLPDU is a DLCEP-address for an existing DLCEP, and the received EC DLPDU requests a reply, and if the addressed DLCEP is in the DATA-TRANSFER-READY state, then
- 1) if the existing DLCEP is a publisher DLCEP, then the DLE shall
 - i) set each QoS parameter, and the publisher-DLCEP-address reuse-discriminator, equal to the corresponding parameter of the specified DLCEP, if permitted by the negotiation rules of 10.2.1.1;
 - ii) if necessitated by the rule of 10.2.1.1d), then change the address size of the existing DLC from VERY-SHORT to SHORT or from SHORT to LONG.
 - 2) If no negotiation-rule violation is detected, then the DLE shall
 - i) encode an EC DLPDU not requesting a reply, with two addresses as specified in 8.1 and 9.1, where its addresses are, respectively,
 - the DLCEP-address of the existing DLC;
 - the DLSAP-address associated with this existing DLCEP-address, respectively;
 - ii) schedule the EC DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5.
 - 3) When 2) does not apply, because a negotiation rule violation was detected, then the DLE shall reject the received DLC-establishment request and terminate processing of the received EC DLPDU, as follows:
 - i) The DLE shall encode a DC DLPDU as specified in 8.2 and , with its reply-requested field set to FALSE, with a reason of “provider-originated disconnection — QoS not available, permanent condition”, and schedule the DC DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5.
 - ii) The DC DLPDU shall have both destination and source addresses (see 8.2.1 formats 1L and 1S), the destination address shall be identical to the first source DL-address of the received EC DLPDU, and the source address shall be identical to the destination DL-address of that received EC DLPDU.
- f) Otherwise, the DLE shall ignore the received EC DLPDU.

10.2.1.5 Expiration of the timer $T_U(\text{MCD})$

If the timer $T_U(\text{MCD})$ expires, then if the DLCEP state, $V_C(\text{ST})$, is

- a) waiting-for-EC-DLPDU, then
- 1) if this is the $(V(\text{NRC})+1)$ 'th consecutive expiration, then

NOTE See the permission at the start of clause 10 with regard to use of $V(\text{NRC})$ versus $V(\text{MRC})$.

 - i) the DLE shall terminate processing of the request;
 - ii) if the user-specified maximum confirm delay on the DL-CONNECT request primitive specified a value other than UNLIMITED, then
 - A) the DLE shall initiate a DL-DISCONNECT indication reporting “connection rejection — DLSAP unreachable, transient condition, local origin”;
 - B) if the called address was either a DLSAP-address or a DLCEP-address, and the DLCEP's DLCEP-class is PEER, then the DLE

— shall encode a DC DLPDU requesting disconnect, with a reason of “reason unspecified”, to the same DL-address as that to which the previous EC DLPDU had been sent;

— shall be queued at TIME-AVAILABLE priority as specified in 10.4.5;

iii) otherwise, when the user-specified maximum confirm delay on the DL-CONNECT request primitive specified a value of UNLIMITED, the DLE shall leave the request queued, and the DLCEP in its current state, but shall initiate no other action with respect to the request.

NOTE The DLE may still respond to receipt of an EC DLPDU which attempts to establish a connection with the above DLCEP, by either accepting or rejecting the proffered connection.

2) otherwise, if 1) does not apply, then the DLE shall

i) restart the $T_U(\text{MCD})$ timer with the same period as the previous time;

ii) requeue the same EC DLPDU for retransmission at TIME-AVAILABLE priority as specified in 10.4.5;

b) WAITING-FOR-CONNECT-RESPONSE, then the DLE shall disconnect the DLCEP as specified in 10.2.1.8, specifying a disconnect reason of “provider-originated disconnection — timeout”;

c) waiting-for-connect-completion, then

1) if this is the $(V(\text{NRC})+1)$ 'th consecutive expiration, then the DLE shall disconnect the DLCEP as specified in 10.2.1.8, specifying a disconnect reason of “provider-originated disconnection — timeout”;

NOTE See the permission at the start of clause 10 with regard to use of $V(\text{NRC})$ versus $V(\text{MRC})$.

2) otherwise, if 1) does not apply, then the DLE shall resend the three-address EC DLPDU which was sent when the WAITING-FOR-CONNECT-COMPLETION state was entered, and again restart the $T_U(\text{MCD})$ timer as specified in 10.2.1.2b).

d) DATA-TRANSFER-READY, then the DLE shall act as specified in 10.2.2.10;

e) WAITING-FOR-RESET-COMPLETION, then the DLE shall act as specified in 10.2.2.18.

10.2.1.6 Receipt of a DL-DISCONNECT request primitive

When the DLE receives at a DLCEP a DL-DISCONNECT request from a DLS-user, then the DLE

a) shall encode a DC DLPDU as specified in 8.2 and , requesting disconnect and specifying the DLS-user-given reason, and shall schedule the DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5, except when the DLCEP

1) is in the WAITING-FOR-CONNECT-RESPONSE state, and the destination DL-address of the EC DLPDU which activated the DLCEP is a group DL-address; or

2) is in the WAITING-FOR-EC-DLPDU state, and the called DL(SAP)-address is a group DL-address or is UNKNOWN; or

3) is a SUBSCRIBER DLCEP;

If a DC DLPDU is encoded, then

i) if the DLCEP being disconnected is a PEER DLCEP, then the DC DLPDU shall have both destination and source addresses (see 8.2.1 formats 1L and 1S), and the destination address shall be the remote DLCEP-address of the DLC, if known, or the called-DL(SAP)-address of the initiating EC DLPDU in all other cases. The reply-requested field shall be set to TRUE in the DC-parameters of the initiating DC DLPDU.

ii) if the DLCEP being disconnected is a PUBLISHER DLCEP, then the DC DLPDU shall have only a source address (see 8.2.1 formats 2L and 2S). The reply-requested field shall be set to FALSE in the DC-parameters of the initiating DC DLPDU.

iii) the source address of the DC DLPDU shall be the local DLCEP-address, if one exists; or the responding or calling local DLSAP-address, if one exists, or the called-DLSAP-address of the initiating EC DLPDU in all other cases.

- b) shall terminate the DLCEP, including
 - 1) for each outstanding (that is, not-yet-confirmed) DL-DATA request:
 - i) remove the request from the appropriate DLCEP user-request queue, $Q_A(UR)$, and references to the request from all DLE queues;
 - ii) initiate a DL-DATA confirm with the associated request identifier reporting “failure — reset or disconnection”;
 - iii) delete the timer $T_U(MCD)$ associated with the request.
 - 2) if an DL-RESET request is outstanding, then initiate a DL-RESET confirm with the DLS-user’s DLCEP-identifier reporting “failure — disconnection”;
 - 3) delete all timers associated with the DLCEP.

After disconnection, if a local DLCEP-address associated with the DLCEP was included in any DLPDU sent on the link, then the DLE shall ensure that the DLCEP-address is not reused for a period of time exceeding the greater of

- 1) twice the maximum DLPDU lifetime in the network, $V(NDL)$;
- 2) the DLCEP’s $D...D_{CR}$ delay, when that delay is not UNLIMITED.

In the state machine of Figure 13, this procedure is modeled as entry into an extra state, Aging-DLCEP-address, which is exited at the end of the above-specified period of time.

10.2.1.7 Receipt of a DC DLPDU

When the DLE receives a DC DLPDU, specifying that the DLCEP should be disconnected, then

- a) the DLE shall determine the version number of the DL-protocol in use, as specified in the received DC DLPDU, and shall interpret the other DC-parameters of the DLPDU accordingly;
- b) if the received DC DLPDU requests a reply, then a return DC DLPDU, addressed to the source DL-address specified in the received DC DLPDU, and specifying a disconnect reason of “disconnection or connection rejection, unknown origin — reason unspecified”, and not requesting a reply, shall be encoded and shall be queued at TIME-AVAILABLE priority as specified in 10.4.5;
- c) if the received DC DLPDU
 - 1) specifies only a source address (see 8.2.1 formats 2L and 2s) and the source address is a DLCEP-address of a multi-peer DLC to which the DLE is a subscriber; or
 - 2) specifies both destination and source addresses (see 8.2.1 formats 1L and 1s), where
 - i) the destination address is a DL(SAP)-address, and the DLE has a DLCEP, at a DLSAP to which that DL(SAP)-address is bound, whose remote DLCEP-address has the same value as the received source DL-address; or
 - ii) the destination address is a DLCEP-address, and the remote DLCEP-address of the identified DLCEP has the same value as the received source DL-address; or
 - iii) the destination address is a DLCEP-address, and the called DLSAP-address of the identified DLCEP has the same value as the received source DL-address
 then if the DLCEP is known to the local DLS-user, then
 - 3) the DLE shall report a DL-DISCONNECT indication to the local DLS-user specifying both the non-local origin and the reason for the DL-DISCONNECT indication as received in the DC DLPDU;
 - 4) the DLE shall terminate the DLCEP as specified in 10.2.1.6b);
 - 5) after disconnection, the DLE shall ensure that any DLCEP-address which had been assigned to the DLCEP is not reused for a period of time exceeding the greater of
 - i) twice the maximum DLPDU lifetime in the network, $V(NDL)$;

- ii) the DLCEP's D_{DLCEP} delay, when that delay is not UNLIMITED.

10.2.1.8 DLE-initiated disconnection

When the DLE determines on its own that it is necessary to disconnect the DLCEP, then

- a) if the DLCEP is known to the local DLS-user, then the DLE shall report a DL-DISCONNECT indication to the local DLS-user, specifying both the reason for the DL-DISCONNECT indication and that its origin was local.

NOTE The DLCEP will not be known to the local DLS-user if the disconnection occurs while processing a received EC DLPDU whose receipt had just triggered the DL to create the DLCEP.

- b) If

- 1) the DLCEP's DLCEP-class is PEER or PUBLISHER;
- 2) the called DL(SAP)-address of the EC DLPDU which activated the DLCEP was not a group DL-address,

then

- i) the DLE shall encode a DC DLPDU as specified in 8.2 and , and shall schedule the DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5;
- ii) if the DLCEP being disconnected is a PEER DLCEP, then the DC DLPDU shall have both destination and source addresses (see 8.2.1 formats 1L and 1S), and the destination address shall be the remote DLCEP-address of the DLC, if known, or the called-DL(SAP)-address of the initiating EC DLPDU in all other cases. The reply-requested field shall be set to TRUE in the DC-parameters of the initiating DC DLPDU;
- iii) if the DLCEP being disconnected is a PUBLISHER DLCEP, then the DC DLPDU shall have only a source address (see 8.2.1 formats 2L and 2S) and the reply-requested field shall be set to FALSE in the DC-parameters of the initiating DC DLPDU;
- iv) the source address of the DC DLPDU shall be the local DLCEP-address, if one exists; or the responding or calling local DLSAP-address, if one exists, or the called-DLSAP-address of the initiating EC DLPDU in all other cases.

- c) the DLE shall terminate the DLCEP as specified in 10.2.1.6b).

After disconnection, the DLE shall ensure that any DLCEP-address which had been assigned to the DLCEP is not reused for a period of time exceeding the greater of

- 1) twice the maximum DLPDU lifetime in the network, $V(NDL)$;
- 2) the DLCEP's D_{DLCEP} delay, when that delay is not UNLIMITED.

10.2.2 Operation of the DLC data transfer and DLCEP reset services

The primitives of the DLC data transfer service are DL-DATA request, indication and confirm, DL-BUFFER-SENT indication, and DL-BUFFER-RECEIVED indication. The primitives of the DLCEP reset services are DL-RESET request, indication, response and confirm, and DL-RESET-COMPLETED indication. A DLCEP reset may be invoked, by any DLS-user or the DLS-provider, at any time after DLCEP establishment and before DLCEP release.

If the sending DLCEP data delivery features in either direction of the DLC are ORDERED, DISORDERED, or CLASSICAL, then the linear (conceptually unlimited) DLCEP variables specified in and Figure 9 are projected onto a cyclic ring of 2^N sequence identifiers as shown in Figure 14. Each DLSDU is associated with one of these identifiers prior to transmission, and all DLPDUs which convey segments of the DLSDU also specify that associated identifier. The dynamic partitioning of these identifiers, and the resultant categorization of DLSDUs associated with these identifiers, is also shown in Figure 14.

NOTE The relationships among the linear (conceptually unlimited) DLCEP variables specified in and Figure 9 are repeated here to assist in the understanding of Figure 14.

$$H-W \leq M-W \leq A \leq L \leq H \leq M \leq A+W \leq L+W$$

$$M-W \leq R \leq M$$

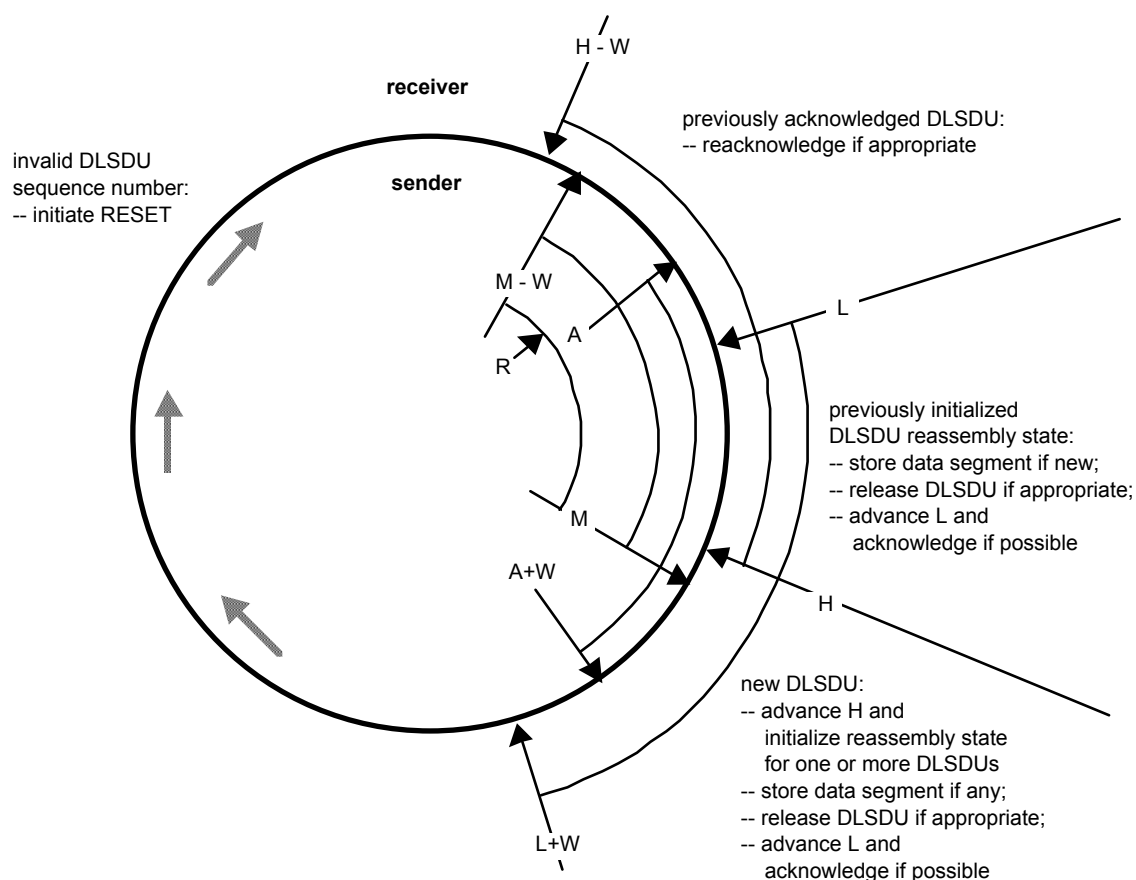


Figure 14 – Projection of the sending and receiving DLCEP sequence-number variables of Figure 9 onto the cyclic sequence-number parameters of CA, CD, DT, ED and RC DLPDUs, with consequent determination of required actions

Each DLPDU with non-null SD-parameters can convey to the peer DLE the low-order 2 to 5 bits of the sender's $V_C(L)$ and $V_C(M)$, as the $(NDR \bmod 2^R)$ and $(NDS \bmod 2^S)$ subfields of the SD-parameters.

10.2.2.1 Selection of the format of a CA, CD, DT and ED DLPDUs

The address format of all CA, CD, ED, and DT DLPDUs sent from a DLCEP shall be chosen as determined during the DLCEP-establishment process (see 10.2.1.1) and as specified in 8.4.3, 8.5.3, 8.6.3 and 8.7.3, respectively. The SD-parameter format of all such CA, CD, ED and DT DLPDUs formed by the DLE shall be the same as that negotiated for the sending DLCEP (see 9.1c)5), 9.1d)5), and 10.2.1.1).

All CD DLPDUs sent from an LAS DLE as part of its schedule execution activities, and not from a DLCEP of the LAS DLE, shall specify an explicit destination address of the length negotiated in 10.2.1.1 and shall omit both source address and SD-parameters.

NOTE An address format of VERY-SHORT is always realized by use of SHORT addresses in any associated CA, CD, DC, ED and RC DLPDUs.

10.2.2.2 Receipt of a DL-DATA request primitive

If the request is accepted, as indicated by a returned status of “success” for the DL-DATA request, then upon completion of the request, either successfully or after failure, the DLE shall issue a DL-DATA confirm with the same request identifier as specified by the DLS-user in the corresponding DL-DATA request primitive, conveying the status of the request to the DLS-user.

The DLCEP source specified in the DL-DATA request should be bound to either an explicit (user-controlled) queue or to an implicit (DLE-controlled) queue. If the queue is full, or if the specified DLSDU length, $P_U(L)$, is invalid, or if the DLCEP-state $V_C(ST)$ is not DATA-TRANSFER-READY, then the DLE shall immediately return the corresponding DL-DATA confirm indicating the reason for failure.

Otherwise

- a) the DLE shall create and start a user-request timer $T_U(MCD)$ with a duration based on the user-specified maximum confirm delay for DL-DATA primitives. If the specified value was other than UNLIMITED, then the duration of this timer shall be equal to that user-specified maximum confirm delay; otherwise the duration should be 60 s. DL-management may override these preferred durations;
- b) the DLE shall assign the next unassigned sequence number $N = V_C(N)$ to the request and its associated DLSDU;
- c) the DLE shall initialize the variable $V_{C,N}(SS)$ based on the length, $P_N(L)$, of the N th DLSDU, to indicate that all segments of the N th DLSDU, and no other segments of that DLSDU, need transmission;
- d) the DLE shall append the request to the DLCEP-address's user-request queue, $Q_A(UR)$, as follows.
 - 1) If the DL-scheduling-policy attribute of the source DLCEP-address is EXPLICIT, then the request shall be placed in the third partition of $Q_A(UR)$, where it shall await a DL-COMPEL-SERVICE request to release it for transmission.
 NOTE Release of a deferred request applies to DLSDUs, not DLPDUs.
 - 2) Else if 1) does not apply, and
 - i) if $N > V_C(A) + P_C(WS)$, and the sending DLCEP is a CLASSICAL or DISORDERED peer, or
 - ii) if $N > V_C(R) + P_C(WS)$, and the sending DLCEP is ORDERED and $V_C(R)$ exists, or is a CLASSICAL or DISORDERED publisher or subscriber,

then the request shall be placed in the third partition of $Q_A(UR)$.

- 3) Else if 1) and 2) do not apply, then the third partition of $Q_A(UR)$ is empty, and so the request shall be placed in the second partition of $Q_A(UR)$, and the DLE shall append to the DLE's unscheduled-service queue, $Q(US)$, a reference to $Q_A(UR)$ of the same priority as the just-appended request.

NOTE $Q(US)$ never needs to have more references to a $Q_A(UR)$ than the number of DLSDUs waiting for transmission or retransmission.

- e) The DLE shall increment $V_C(N)$.

10.2.2.3 Transmission of a DT DLPDU from a DLCEP

In the following, let MUD be defined as the maximum number of user data octets which may be conveyed in a single DT DLPDU of the DLCEP's priority, as specified in 6.2.3.

Upon receipt of a transmission opportunity for a queued or buffered DLSDU, the DLE shall form and send a DT DLPDU of the specified priority; with DL-address field and SD-parameter field formats as specified in 8.7, 9.4 and 10.2.2.1; with the remote (destination) and local (source) DLCEP-addresses of the DLC, as appropriate; with user data-field length and contents, and SD-parameter-field contents, as follows:

NOTE With the exception of the final-token-use subfield of the DT DLPDU's first (frame control) octet; probably of the NDR, RSN, J, K and T subfields, if present, of the SD-parameter format negotiated for transmission from the sending DLCEP, the remainder of the data DT DLPDU may be formed when the request is queued, and need not be formed dynamically at the moment of transmission.

10.2.2.3.1 Formation of the user data field and related SD-parameter subfields

The T, NDS, TNS, ASN and truncated-DL-time subfields of the SD-parameters field, and the DLPDU's user data field, shall be formed as follows.

- a) If
 - i) the sending DLCEP is a DISORDERED or CLASSICAL PUBLISHER DLCEP, or an ORDERED PEER or PUBLISHER DLCEP and $V_C(R)$ exists;
 - ii) $V_C(M) = V_C(R) + P_C(WS)$ and $V_C(M) + 1 < V_C(N)$ and $V_C, V_C(M)(SS)$ is empty;
 - iii) a DL-DATA confirm primitive has been issued for the $(V_C(R)+1)$ 'th DL-DATA request
 then the DLE shall
 - 1) cancel the timer associated with the $(V_C(R)+1)$ 'th DL-DATA request;
 - 2) increment $V_C(R)$.
- b) If the sending DLCEP is bound to a sending queue, and is
 - i) a SUBSCRIBER DLCEP, or an UNORDERED PEER or PUBLISHER DLCEP, and there is a smallest K such that $V_C(M) < K \leq V_C(N)-1$ and $V_C, K(SS)$ is non-empty;

NOTE In this case $V_C, K(SS)$ can contain only a single element, since such DLSDUs cannot require segmentation.
 - ii) a DISORDERED or CLASSICAL PEER DLCEP, and there is a K such that $V_C(A) < K \leq \min(V_C(A) + P_C(WS), V_C(N)-1)$ and $V_C, K(SS)$ is non-empty;
 - iii) a DISORDERED or CLASSICAL PUBLISHER DLCEP, or an ORDERED PEER or PUBLISHER DLCEP and $V_C(R)$ exists; and there is a K such that $V_C(R) < K \leq \min(V_C(R) + P_C(WS), V_C(N)-1)$ and $V_C, K(SS)$ is non-empty; or
 - iv) an ORDERED PEER or PUBLISHER DLCEP and $V_C(R)$ does not exist; and there is a K such that $V_C(M) \leq K \leq \min(V_C(M) + P_C(WS), V_C(N)-1)$ and $V_C, K(SS)$ is non-empty; or
 then the DLE shall form the remainder of the DLPDU as follows.
 - 1) The NDS subfield, if present, shall convey the lowest-order three or five bits of the value K , as appropriate.
 - 2) If S is the zero-origin index of the lowest-numbered member of the set $V_C, K(SS)$, which is the segment number of a segment of the K 'th DLSDU requiring transmission, then the ASN subfield, if present, shall convey the value of S .

NOTE It is possible for an implementation of this DL-protocol to perform DLSDU segmentation at the time of DLPDU transmission. Such an implementation approach could minimize the DLE's processing of DLS-user data.
 - 3) The TNS subfield, if present, shall convey the value $(P_K(SDUL) - 1)/MUD$, rounded down, which is the zero-origin number of segments in the K 'th DLSDU.
 - 4) The user data field shall consist of octets

$S \times MUD + 1$ to $\min((S+1) \times MUD, P_K(SDUL))$, inclusive, of $P_K(SDU)$.
 - 5) The T subfield shall specify FALSE.

6) The truncated-DL-time subfield shall not be present.

the DLE shall remove member number S from $V_{C,K}(SS)$.

c) If the sending DLCEP is bound to a sending buffer, then

- 1) the DLE shall increment $V_C(N)$.if the buffer has been written since the last transmission from the buffer on this DLCEP;
- 2) the DLE shall let K equal $V_C(N)-1$;
- 3) if K is not equal to zero, then the DLE shall form the remainder of the DLPDU as specified in b)1) to b)4) and as follows.

i) If the DLSDU has no timeliness attribute, then the T subfield shall specify FALSE.

ii) If the DLCEP has a sender's-DL-timeliness class other than NONE, then the DLE shall

A) compute the timeliness of the S th segment of the K 'th DLSDU as specified in 10.1.7;

B) perform a logical AND of that computed timeliness status with the timeliness-status associated with writing the buffer, $V_B(TS)$ (see 6.7.4.21);

C) convey that result in the T subfield of the DLPDU.

iii) If present, the truncated-DL-time subfield shall convey the appropriate octets of the time-of-production associated with the buffer.

the DLE shall remove member number S from $V_{C,K}(SS)$.

d) If there is no such K as in b), or if K is equal to zero in c), or if the DLE is required to send the DLPDU without user data as in 8.4.4.1c), 8.5.4.1c), and 8.6.4.1c); or 8.4.4.1d), 8.5.4.1d), and 8.6.4.1d), then

1) the T , TNS , ASN and truncated-DL-time subfields of the SD-parameters, if present, shall be encoded as zero (0); NDS shall be encoded as the appropriate number of low-order bits of $V_C(M)$;

2) the truncated DL-time subfield may be null (that is, omitted and not transmitted);

3) the user data field shall be null.

10.2.2.3.2 Formation of the other SD-parameter subfields

The J , K , NDR and RSN subfields of the SD-parameters shall be formed as follows:

a) If the sending DLCEP is a PUBLISHER DLCEP, then the J , K , NDR and RSN subfields of the SD-parameters shall be encoded as zero (0).

b) If the sending DLCEP is a PEER or SUBSCRIBER DLCEP, then

1) if there is a smallest K such that $V_C(L) < K \leq V_C(H)$ and $V_{C,K}(RRS)$ is non-empty, then if S is the zero-origin index of the lowest-numbered member of the set $V_{C,K}(RRS)$, which is the segment number of a missing segment of the K 'th DLSDU, and either $K < V_C(H)$ or $S \leq V_C(HS)$, then

i) the J subfield of the SD-parameters, if present, shall be encoded as one (1);

ii) the NDR subfield, if present, shall convey the lowest-order two or four bits of the value K ;

iii) the RSN subfield, if present, shall convey the value of S ;

iv) segment S shall be removed from $V_{C,K}(RRS)$.

- 2) Otherwise, if 1) does not apply, either the SD-parameter field shall be null, if permitted by 10.2.2.1; or
 - i) the J and RSN subfields shall be encoded as zero (0), if present;
 - ii) the NDR subfield, if present, shall convey the lowest-order two or four bits of $(V_C(L)+1)$.
- 3) If the K and NDR subfields of the SD-parameters are both present, and the value of the NDR subfield equals the value of the corresponding lowest-order bits of $(V_C(L)+1)$, then the K subfield shall be encoded as one (1); otherwise the K subfield shall be encoded as zero (0).

10.2.2.3.3 Transmission completion

- a) If the just-transmitted DLPDU contained DLS-user data, then the DLE shall update $V_C(M)$ and $V_C(MS)$ from the local variables K and S of 10.2.2.3.1 as follows.
 - If $K > V_C(M)$, or $K = V_C(M)$ and $S > V_C(MS)$, then $V_C(M)$ shall be set equal to K , and $V_C(MS)$ shall be set equal to S .
- b) If the sending DLCEP
 - is a peer DLCEP whose sending DLCEP features are UNORDERED, or is a publisher DLCEP whose sending DLCEP features are UNORDERED, or is a subscriber DLCEP;
 - the DT DLPDU has a non-null user data field which contains the last, or only, segment of a DLSDU;

then

 - 1) if the sending source is a buffer, then the DLE shall issue a DL-BUFFER-SENT indication primitive specifying the DLS-user-identifier if known, or the DL-identifier otherwise, for the DLCEP; or
 - 2) if the sending source is a queue and there is an unconfirmed DL-DATA request for that DLSDU, then the DLE shall
 - i) issue a DL-DATA confirm primitive with the same request identifier as that DL-DATA request, reporting “success”;
 - ii) cancel and delete the associated timer $T_U(MCD)$.
- c) If the sending DLCEP
 - is a publisher DLCEP whose sending DLCEP features are ORDERED, DISORDERED or CLASSICAL, or is a peer DLCEP whose sending DLCEP features are ORDERED;
 - the DT DLPDU has a non-null user data field which contains the last, or only, segment of a DLSDU;

then

 - 1) if the sending source is a buffer, then the DLE shall issue a DL-BUFFER-SENT indication primitive specifying the DLS-user-identifier, if known, or the DL-identifier otherwise, for the DLCEP; or
 - 2) if the sending source is a queue and there is an unconfirmed DL-DATA request for that DLSDU, then
 - i) the DLE shall issue a DL-DATA confirm primitive with the same request identifier as that DL-DATA request, reporting “success”;
 - ii) if $V_C(R)$ exists, and $V_C(M) - P_C(WS) > V_C(R)$, then the DLE shall set $V_C(R)$ equal to $V_C(M) - P_C(WS)$;

- iii) the DLE shall retain the DLSDU for potential retransmission to subscribers
 - until the expiration of the associated timer $T_U(\text{MCD})$; or
 - until the need to transmit another DLSDU with a resultant increase in $V_C(\text{M})$, when coupled with the negotiated window size $P_C(\text{WS})$, requires the discard of the retained DLSDU with a resultant increase in $V_C(\text{R})$.
- d) If this DLCEP has been specified as a synchronizing DLCEP during the establishment of one or more other local DLCEPs, and if a DL-BUFFER-SENT indication primitive was issued in b)1) or c)1), then the DLE shall record the DL-time of network access, $V_C(\text{TNA})$, for use in the timeliness computations of those referencing DLCEP(s).

10.2.2.4 Transmission of a CA, CD or ED DLPDU from a DLCEP

10.2.2.4.1 Transmission of a CA DLPDU

Upon receipt of a transmission opportunity to compel transmission from a remote DLCEP, when

- a) the DLC is
 - 1) simplex, with DLS-user data transmission only from the local DLCEP to the remote DLCEP; or
 - 2) duplex, with DLS-user data transmission from the local DLCEP to the remote DLCEP, and from that remote DLCEP to the local DLCEP (and possibly other DLCEPs), and the DLCEP attributes do not permit sending an ED DLPDU from the DLCEP to the remote DLCEP;
- b) the sending DLE needs to determine the state of the remote DLCEP;

then the DLE shall form and send a CA DLPDU of the specified priority; with DL-address field and SD-parameter field formats as specified in 8.5, 9.4 and 10.2.2.2; with the remote (destination) and local (source) DLCEP-addresses of the DLC, as appropriate; and with SD-parameter-field contents and user data-field contents as specified in 10.2.2.3.

10.2.2.4.2 Transmission of a CD DLPDU

This subclause does not apply to the LAS DLE when it sends CD DLPDUs as part of its scheduled activity and not from a DLCEP of the LAS DLE; such DLPDUs are constrained as specified in 8.5 and 10.2.2.

Upon receipt of a transmission opportunity to compel transmission from a remote DLCEP, when

- a) the DLC is
 - 1) simplex, with DLS-user data transmission only from the remote DLCEP to the local DLCEP (and possibly other DLCEPs); or
 - 2) duplex, with DLS-user data transmission from the local DLCEP to the remote DLCEP, and from that remote DLCEP to the local DLCEP (and possibly other DLCEPs), and either
 - i) the DLCEP attributes do not permit sending an ED DLPDU from the local DLCEP to the remote DLCEP, or
 - ii) there are no DLSDU segments awaiting transmission to the remote DLS-user as specified by the selection criteria of 10.4.2.1a);
- b) the local execution of a DL-COMPEL-SERVICE request primitive, or of a scheduled compel-service action, compels transmission from a remote peer or publisher DLCEP;

then the DLE shall form and send a CD DLPDU of the specified priority; with DL-address field and SD-parameter field formats as specified in 8.5, 9.4 and 10.2.2.2; with the remote (destination) and local (source) DLCEP-addresses of the DLC, as appropriate; and with SD-parameter-field contents, as follows:

NOTE With the exception of the final-token-use subfield of the CD DLPDU's first (frame control) octet; and probably of the NDR, RSN, J and K subfields, if present, of the SD-parameter format negotiated for transmission from the sending DLCEP, the remainder of the CD DLPDU may be formed when the request is queued, and need not be formed dynamically at the moment of transmission.

- 1) The T, NDS, TNS, ASN and truncated-DL-time subfields of the SD-parameters, if present, shall be encoded as zero (0).
- 2) The truncated DL-time subfield may be null (that is, omitted and not transmitted).
- 3) The user data field shall be null.
- 4) the J, K, NDR and RSN subfields of the SD-parameters shall be formed as specified in 10.2.2.3.2.

10.2.2.4.3 Transmission of an ED DLPDU

Upon receipt of a transmission opportunity to compel transmission from a remote DLCEP, when

- a) the DLC is duplex, with DLS-user data transmission from the local DLCEP to the remote DLCEP, and from that remote DLCEP to the local DLCEP (and possibly other DLCEPs);
- b) the DLCEP attributes permit sending an ED DLPDU from the DLCEP to the remote DLCEP;
- c) one or more DLSDU segments await transmission to the remote DLS-user as specified by the selection criteria of 10.4.2.1a);
- d) the local execution of a DL-COMPEL-SERVICE request primitive, or of a scheduled compel-service action, compels transmission from a remote peer or publisher DLCEP;

then the DLE shall form and send an ED DLPDU of the specified priority; with DL-address field and SD-parameter field formats as specified in 8.5, 9.4 and 10.2.2.2; with the remote (destination) and local (source) DLCEP-addresses of the DLC, as appropriate; and with SD-parameter-field contents and user data-field contents as specified in 10.2.2.3.

10.2.2.5 Validation and processing of SD-parameters in a CA, CD, ED or DT DLPDU received at a DLCEP

If the DLCEP state, $V_C(ST)$, is

- 1) waiting-for-connect-completion, then
 - i) if the state before WAITING-FOR-CONNECT-COMPLETION was WAITING-FOR-EC-DLPDU, then the DLE shall issue a DL-CONNECT confirm primitive, conveying the negotiated DLCEP-attributes, to the requesting DLS-user and cancel the associated user request timer $T_U(MCD)$;
 - ii) else, when i) does not apply, then the state before WAITING-FOR-CONNECT-COMPLETION was WAITING-FOR-CONNECT-RESPONSE, and the DLE shall issue a DL-CONNECTION-ESTABLISHED indication primitive to the receiving DLS-user and cancel the associated user request timer $T_U(MCD)$;
 - iii) the DLE shall change the DLCEP state, $V_C(ST)$, to DATA TRANSFER READY; and shall apply the remainder of this subclause.
- 2) waiting-for-reset-completion, then
 - i) the DLE shall issue a DL-RESET-COMPLETED indication primitive to the receiving DLS-user specifying the DLS-user-identifier for the DLCEP if known, or the DL-identifier for the DLCEP otherwise;
 - ii) the DLE shall cancel the associated user request timer $T_U(MCD)$;

- iii) the DLE shall change the DLCEP state, $V_C(ST)$, to DATA-TRANSFER-READY; and shall apply the remainder of this subclause.
- 3) not WAITING-FOR-CONNECT-COMPLETION, and not WAITING-FOR-RESET-COMPLETION, and not DATA-TRANSFER-READY, then the received DLPDU shall be ignored by the upper-level DLC functions.

Otherwise, the DLE shall validate and process the SD-parameters of the received DLPDU according to the SD-parameter format, $P_C(NP.FFF_R)$, negotiated for this (receiving) direction of DLC transmission. This validation and processing shall be as specified in the remainder of 10.2.2.5, with format-dependent considerations as follows, based on the SD-parameter format (A – G) and the truncated DL-time format (J – M). The format-dependent value of the sending modulus MOD_S shall also be used in the procedures of 10.2.2.6.

- Format A) The sending and receiving SD-parameters of the DLPDU are implicit and thus always valid; the implied values of RSN, T, TNS, ASN and truncated DL-time are all zero; and any accompanying user data is a complete DLSDU. 10.2.2.5.2 does not apply.
- Format B) The sending and receiving SD-parameters of the DLPDU are explicit; the sending modulus MOD_S equals 2^3 ; the receiving modulus MOD_R equals 2^2 ; the implied values of RSN, TNS and ASN are zero; and any accompanying user data is a complete DLSDU.
- Format C) The sending and receiving SD-parameters of the DLPDU are explicit; the sending modulus MOD_S equals 2^5 ; the receiving modulus MOD_R equals 2^4 ; and any accompanying user data may be only a partial DLSDU.
- Format D) The sending and receiving SD-parameters of the DLPDU are explicit; the sending modulus MOD_S equals 2^5 ; the receiving modulus MOD_R equals 2^4 ; the implied values of TNS and ASN are zero; and any accompanying user data is a complete DLSDU.
- Format E) The sending SD-parameters of the DLPDU are non-existent; the receiving SD-parameters of the DLPDU are explicit; the receiving modulus MOD_R equals 2^4 ; the implied values of J, K and T are as specified in 9.4.2.1E); the implied value of NDS is $V_C(H)+1$; the implied values of TNS and ASN are zero; and there cannot be accompanying user data.
- Format F) The sending SD-parameters of the DLPDU are explicit; the receiving SD-parameters of the DLPDU are non-existent; the sending modulus MOD_S equals 2^5 ; the implied value of NDR is $V_C(M)+1$; the implied value of RSN is zero; and any accompanying user data may be only a partial DLSDU.
- Format G) The sending SD-parameters of the DLPDU are explicit; the receiving SD-parameters of the DLPDU are non-existent; the sending modulus MOD_S equals 2^5 ; the implied value of NDR is $V_C(M)+1$; the implied values of RSN, TNS and ASN are zero; and any accompanying user data is a complete DLSDU.

10.2.2.5.1 Validation of the NDS, TNS, ASN and truncated-DL-time subfields of the received SD-parameters

In the following, $P_C(NP.WWWW_R)$ is the negotiated receive window size and $P_C(NP.TT_R)$ is the negotiated receiving DLCEP data delivery features.

- a) If $P_C(NP.TT_R)$ specifies UNORDERED, as is always the case with format A, then if the received DLPDU's user data field is non-null, then the receiving DLE

- i) shall increment $V_C(H)$, and shall let K equal the new value of $V_C(H)$;
 - ii) shall create the variable $V_{C,K}(MRS)$ with a value indicating that segment number zero (0) of the K 'th DLSDU is missing;
 - iii) shall process the received user data as specified in 10.2.2.6.
- b) Otherwise, when $P_C(NP.TT_R)$ specifies ORDERED, DISORDERED or CLASSICAL, and if the receiving DLCEP is a subscriber DLCEP, and this is the first DT DLPDU received after the DLCEP state was changed to DATA-TRANSFER-READY, then the DLE shall set the variables $V_C(L)$ and $V_C(H)$ to the value of the $N_R(NDS)$ subfield of the received DT DLPDU.

The DLE shall compute

$$TEMP = (N_R(NDS) + P_C(NP.WWWW_R) - V_C(H) - 1) \text{ modulo } MOD_S \quad (16)$$

1) If

$$TEMP > (V_C(L) + 2 \times P_C(NP.WWWW_R) - V_C(H) - 1) \text{ modulo } MOD_S$$

then

- i) if $P_C(NP.TT_R)$ is ORDERED, then the DLE shall

A) set $V_C(L)$ equal to

$$V_C(L) + ((N_R(NDS) - (P_C(NP.WWWW_R) + V_C(L))) \text{ modulo } MOD_S);$$

B) cancel all timers $T_{C,N}(RRS)$ which may exist, where N is less than or equal to $V_C(L)$. All DLE resources devoted to reception and reassembly of DLSDUs with sequence numbers less than or equal to $V_C(L)$ should be released;

- ii) if $P_C(NP.TT_R)$ is CLASSICAL or DISORDERED, then the received DLSDU sequence number is invalid; the procedures of 10.2.2.6 do not apply; and the DLE shall initiate a reset at the DLCEP (see 10.2.2.19).

2) Else if 1)ii) does not apply, then if either

— $TEMP > (P_C(NP.WWWW_R) - 1)$, or

— $TEMP = (P_C(NP.WWWW_R) - 1)$ and $N_R(ASN) > V_C(HS)$

then

- i) the DLE shall set N equal to $TEMP - (P_C(NP.WWWW_R) - 1)$;

- ii) if $N > 0$, then the received DLSDU sequence number is for a new DLSDU, not previously received or inferred; the DLE shall repeat the following step A) N times, followed by step B) once.

A) The DLE shall increment $V_C(H)$. Let K equal the just-incremented value of $V_C(H)$. Then $V_{C,K}(MRS)$ shall be created and shall indicate that all possible segments of the K 'th DLSDU, based on the negotiated maximum-DLSDU-size, are missing; and $V_{C,K}(RRS)$ shall be created and shall indicate that segment number zero (0) of the K 'th DLSDU is missing.

NOTE 1 Alternatively, the DLE can simply set $V_{C,K}(MRS)$ to indicate that all (16) segments are missing; later procedures will correct the number of missing segments to those of the actual DLSDU.

NOTE 2 This combination of values for $V_{C,K}(MRS)$ and $V_{C,K}(RRS)$ ensures that all segments of the K 'th DLSDU will be received before the reassembled DLSDU is delivered to the DLS-user.

NOTE 3 The DLE repeats the above step N times.

B) if the received SD-parameters contain an explicit TNS field, then the DLE shall modify $V_{C,K}(MRS)$ to indicate that all segments whose zero-origin number is greater than the value of TNS are not missing, where K equals the new value of $V_C(H)$ after step A).

- iii) if $N = 0$, then the DLE shall set K equal the value of $V_C(H)$.

- iv) for all values of N ,
- A) the DLE shall set the variable $V_C(HS)$ equal to the value of the $N_R(ASN)$ field. If there is any accompanying user data in the received DLPDU, then the DLE shall modify both $V_{C,K}(MRS)$ and $V_{C,K}(RRS)$ to indicate that the segment whose zero-origin number is equal to the value of $N_R(ASN)$ field is not missing, and the procedures of 10.2.2.6 also shall be applied.
 - B) if there is any $V_{C,K}(RRS)$, as created in b)2)ii)A), which is not empty and which therefore requires a retransmission request, and if the receiving DLCEP is a CLASSICAL or DISORDERED DLCEP, or optionally is an ORDERED DLCEP, then
 - the DLE shall check for a reference to the DLCEP on the DLE's unscheduled-service queue, $Q(US)$;
 - if no such reference is found then the DLE shall add a reference to the DLCEP onto the DLE's unscheduled-service queue, $Q(US)$, to ensure that another DLPDU requesting retransmission of the missing segment, is sent from the receiving DLCEP.
- 3) Else if 1) and 2) do not apply, and
- $$TEMP < (V_C(L) + P_C(NP.WWWW_R) - V_C(H)) \text{ modulo } MOD_S$$
- then the received DLSDU sequence number is for a previously delivered, and on peer DLCs previously acknowledged, DLSDU. If there is any accompanying user data in the received DLPDU, and the DLCEP is a CLASSICAL or DISORDERED peer DLCEP, then the DLE shall check for a reference to the DLCEP on the DLE's unscheduled-service queue, $Q(US)$, and if not found then add a reference to the DLCEP to the DLE's unscheduled-service queue, $Q(US)$, to ensure that another DLPDU reacknowledging the just-referenced DLSDU is sent from the receiving DLCEP.
- If the DLCEP's receive binding is explicitly or implicitly to a queue, then the procedures of 10.2.2.6 do not apply. If the DLCEP's receive binding is to a buffer, then any DLS-user data in the DLPDU shall be discarded and the receipt of the duplicate DLPDU shall be reported to the DLS-user with a DL-BUFFER-RECEIVED indication specifying that the reported DLSDU is a duplicate DLSDU.
- 4) Else if 1), 2) and 3) do not apply, then the received DLSDU sequence number is for a previously received or inferred, but not yet acknowledged, or delivered, or both, DLSDU.

$$\text{Let } K = V_C(H) + TEMP + 1 - P_C(NP.WWWW_R).$$

If the received SD-parameters contain an explicit TNS subfield, then $V_{C,K}(MRS)$ and $V_{C,K}(RRS)$ both shall be modified to indicate that all segments whose zero-origin number is greater than TNS are not missing. If there is any accompanying user data in the received DLPDU, and $V_{C,K}(MRS)$ indicates that the user data has not previously been received, then the DLE shall modify both $V_{C,K}(MRS)$ and $V_{C,K}(RRS)$ to indicate that the segment whose zero-origin number is equal to the value of $N_R(ASN)$ field is not missing, and the procedures of 10.2.2.6 also shall be applied.

10.2.2.5.2 Validation of the NDR, RSN, J and K subfields of the received SD-parameters

In the following, $P_C(NP.WWWW_R)$ is the negotiated receive window size and $P_C(NP.TT_R)$ is the negotiated receiving DLCEP data delivery features.

- a) If the DLCEP is a subscriber DLCEP, and the NDR, RSN, J and K subfields, if present, of the received SD-parameters are not all zero, then the DLE shall disconnect from the DLCEP as specified in 10.2.1.8 with a reason of "provider-originated disconnection — wrong DLPDU format or parameters, permanent condition".
- b) If the DLCEP is a CLASSICAL or DISORDERED peer DLCEP, and the J and K subfields of the received SD-parameters are not both zero, then the DLE shall compute

$$TEMP = (N_R(NDR) - V_C(A)) \text{ modulo } MOD_R \quad (17)$$

$$N = TEMP + V_C(A) \quad (18)$$

The received DLPDU is acknowledging a previously unacknowledged transmitted DLSDU ($K=1$), or requesting retransmission of a segment of a previously transmitted DLSDU ($J=1$), or both.

If $K=1$, and the DLCEP is a CLASSICAL or DISORDERED peer DLCEP, and $V_C(A) < N \leq V_C(M) + 1$, then the DLE shall

- i) set $V_C(A)$ equal to $N-1$;
- ii) issue, in the order originally requested, a DL-DATA confirm for each DL-DATA request which was acknowledged by the received NDR;
- iii) cancel the set of associated user request timers $\{ T_U(MCD) \}$ for the just-confirmed DL-DATA requests;
- iv) cancel any retransmission timers $T_{C,K}(SS)$ associated with the just-confirmed DL-DATA requests, or the simplified timer $T_C(SS)$ associated with the DLCEP, and in this latter case (using $T_C(SS)$), if $V_C(A) < V_C(M)$, which implies that there are unacknowledged DLSDUs, then $T_C(SS)$ shall be restarted;
- v) where possible and permitted, move DL-DATA requests from the third partition to the second partition of the corresponding user-request queue, $Q_A(UR)$, as specified in 10.2.2.2d);

if the $V_{C,K}(SS)$ associated with the just-confirmed DL-DATA requests were not empty, then the DLE may cancel such retransmission requests and set the corresponding $V_{C,K}(SS)$ to empty.

If $J=1$, and N is greater than $V_C(A)$, and either

- $N < V_C(M)$, or
- $N = V_C(M)$ and $RSN \leq V_C(MS)$,

then the DLE shall add the RSN 'th member to the set $V_{C,N}(SS)$; and if the set $V_{C,N}(SS)$ was previously empty, then the DLE shall

- cancel any retransmission timers $T_{C,N}(SS)$ associated with the N 'th DLSDU, or $T_C(SS)$ associated with the DLCEP;
- add to the DLE's unscheduled-service queue, $Q(US)$, a reference to $Q_A(UR)$ of the receiving DLCEP, to ensure that the requested DLPDU is sent from the receiving DLCEP.

NOTE $Q(US)$ never needs to have more references to a $Q_A(UR)$ than the number of DLSDUs waiting for transmission or retransmission.

- c) If the DLCEP is an ORDERED DLCEP or a publisher DLCEP, and $V_C(R)$ exists, and the J subfield of the received SD-parameters is not zero, then the DLE shall compute

$$TEMP = (N_R(NDR) - V_C(R)) \text{ modulo } MOD_R \quad (19)$$

- 1) If $TEMP > (V_C(M) - V_C(R))$, or
 $TEMP = (V_C(M) - V_C(R))$ and $RSN > V_C(MS)$,
 then the received sequence number residue $N_R(NDR)$ for an acknowledged or a re-requested DLSDU is obsolete or invalid and shall be ignored.

NOTE Misordering of transmitted DLPDUs, which can result in this condition, is possible when the communications path between the sending and receiving DLEs includes active or backup redundant bridges or DL-paths.

- 2) Else if 1) does not apply, then the received DLPDU is requesting retransmission of a segment of a previously-transmitted DLSDU ($J=1$). Let N equal $TEMP + V_C(R)$. If $J=1$ and N is greater than $V_C(R)$, then the DLE shall add the RSN 'th member to the set $V_{C,N}(SS)$; and if the set $V_{C,N}(SS)$ was previously empty, then the DLE shall add to the DLE's unscheduled-service queue, $Q(US)$, a reference to $Q_A(UR)$ of the receiving DLCEP, to ensure that the requested DLPDU is sent from the receiving DLCEP.

NOTE $Q(US)$ never needs to have more references to a $Q_A(UR)$ than the number of DLSDUs waiting for transmission or retransmission.

10.2.2.5.3 Processing of the T and truncated DL-time subfields of the received SD-parameters

If the DLCEP's receive binding is to a buffer, then

- a) if the receiving DLCEP has a sender's DL-timeliness class of NONE, then the timeliness-status, $V_B(TS)$ (see 6.7.4.21), associated with writing the buffer shall be set to FALSE;
- b) otherwise, when a) does not apply, then

- 1) If the received DLPDU conveyed the first-received segment of the DLSDU, then
 - i) the buffer's associated timeliness-status, $V_B(TS)$ (see 6.7.4.21), shall be set equal to the T subfield of the received DLPDU;
 - ii) if that timeliness-status is TRUE, and if the SD-parameter included time-of-production, format K to M, then the time-of-production of the buffer, $V_B(TP)$ (see 6.7.4.20), shall be inferred as the most recent DL-time whose residue under the negotiated DLPDU format would give rise to the DL-time residue conveyed by the received DLPDU;

NOTE This inference will cause the current DL-time to be inferred when the SD-parameters field of the received DLPDU does not convey octets of DL-time.

- iii) the DL-time of reception of the DLPDU shall be used as the time of writing the buffer, $V_B(TW)$ (see 6.7.4.19).
- 2) if 1) does not apply, so that the received DLPDU conveyed a not-previously-received segment of a multi-segment DLSDU for which at least one segment had been previously received, then
 - i) the value of the T subfield from the newly received DLPDU shall be ANDed into the buffer's associated timeliness-status, $V_B(TS)$ (see 6.7.4.21);
 - ii) any DL-time conveyed in the SD-parameter of the DLPDU shall be ignored;
 - iii) the DL-time of reception of the DLPDU shall be used as the time of writing the buffer, $V_B(TW)$ (see 6.7.4.19).

10.2.2.6 Validation and processing of user data received in a CA, DT or ED DLPDU

In the following, MUD is defined to be the maximum number of user data octets which may be conveyed in a single CA, DT or ED DLPDU of the DLCEP's priority, as specified in 6.2.3.

If a received CA, DT or ED DLPDU has a non-null user data field following its SD-parameters field, then the contained data is the ASN 'th segment, of TNS segments, of the K 'th DLSDU, where ASN and TNS are both zero-origin, and K has the value last given it in 10.2.2.5.1a)i) or 10.2.2.5.1b)2)ii)A) or 10.2.2.5.1b)2)iii) or 10.2.2.5.1b)4).

The receiving DLE shall check whether $(ASN \times MUD \text{ plus the length of the received user data})$ is less than or equal to the permitted maximum DLSDU size, $P_C(NP.M...M_R)$, negotiated for this (receiving) direction of DLC transmission. If ASN is less than TNS, then the DLE shall also check whether the length of the received user data is equal to MUD. If either of these requirements is violated, then the DLE shall disconnect the DLCEP as specified in 10.2.1.8, with a reason of “provider-originated disconnection — wrong DLSDU size, permanent condition”:

If member number ASN is a member of the set variable $V_{c,k}(MRS)$, then

- a) the receiving DLE shall determine whether a received-data record has already been allocated to the K 'th DLSDU, and if not, shall allocate a record which may contain at least $\min(P_C(NP.M...M_R), TNS \times MUD)$ octets of DLS-user data and shall associate that record with the K 'th received DLSDU;
- b) the receiving DLE shall copy the user data field from the received DLPDU to that received-data record, starting with octet $(ASN \times MUD + 1)$ of the received-data record;
- c) if the set variable $V_{C,K}(MRS)$ is now empty, then the receiving DLE shall attempt to deliver the DLSDU as specified in 10.2.2.7.

NOTE It is possible for an implementation of this DL-protocol to perform DLSDU reassembly at the time of DLPDU receipt. Such an implementation approach could minimize the DLE's processing of DLS-user data.

10.2.2.7 Delivery of an entire DLSDU which has been completely received at a DLCEP

Let K be the sequence number associated with the completely received DLSDU, as determined by 10.2.2.5.1 at the time that DLSDU reception was completed.

If K is greater than $V_C(L)$, and the current state $V_C(ST)$ of the DLCEP is DATA TRANSFER READY, then the DLCEP's receiving data-delivery features and the type of receive buffer or queue binding determine the DLSDU delivery policy.

- a) If the DLC is a CLASSICAL DLC, and K is greater than $V_C(L) + 1$, then the DLSDU has been received out of order, and the DLE shall retain but not deliver the DLSDU at this time.
- b) Otherwise,
 - 1) if a receiving buffer is bound to the DLCEP, then the DLE shall do as specified in 10.2.2.7.1;
 - 2) if an explicit receiving queue is bound to the DLCEP, then the DLE shall do as specified in 10.2.2.7.2; or
 - 3) if no receiving queue is bound to the DLCEP, which is the OSI default situation, then the implementation shall consistently either
 - i) do as specified in 10.2.2.7.3; or
 - ii) treat this as a variant of case b), using a receiving queue which the implementation has assigned for this purpose.

NOTE This queue may be unique to this DLCEP, or may be shared with other similar-priority DLCEPs at this DLSAP.

10.2.2.7.1 Delivery to a receive buffer

- a) If the receiving DLCEP has a receiver timeliness attribute other than NONE, then the DLE shall set the variable $V_B(TW)$ (see 6.7.4.19) associated with writing the buffer to the current DL-time.

NOTE 1 This is the local time of receipt, and not the remote time-of-production.

- b) The DLE shall deliver the complete DLSDU as the new contents of the buffer, and shall associate any timeliness information received in the conveying DLPDU(s).
- c) The DLE shall report a DL-BUFFER-RECEIVED indication to the DLS-user. If the receiving DLCEP-class was ORDERED, and the received DLSDU is a duplicate of a prior-received DLSDU, then the duplicated-DLSDU attribute of the DL-BUFFER-RECEIVED indication shall specify TRUE; in all other cases it shall specify FALSE.

If this DLCEP has been specified as a synchronizing DLCEP during the establishment of one or more other local DLCEPs, then the DLE shall record the DL-time of network access, $V_C(TNA)$, for use in the timeliness computations of those referencing DLCEP(s).

- d) Any ongoing accesses to the contents of the buffer which are incomplete at the time of DLSDU delivery shall not be affected by the DLSDU delivery.

NOTE 2 This constraint ensures that each access to a buffer is logically atomic.

- e) The DLE shall set $V_C(L)$ equal to K ; and shall cancel all timers $T_{C,N}(RRS)$ which may exist, where N is less than or equal to K . All DLE resources devoted to reception and reassembly of DLSDUs with sequence numbers less than or equal to K should be released.

10.2.2.7.2 Delivery to a receive queue

The DLE shall attempt to append the complete DLSDU, together with identification of the receiving DLCEP, to the receiving queue.

If unsuccessful, the DLE shall inform local DL-management of this queue-full situation.

NOTE This DL-management notification may take the form of incrementing a counter of discarded DLSDUs.

If successful,

- a) the DLE shall report a DL-DATA indication to the DLS-user;
- b) the DLE shall cancel the timer $T_{C,K}(RRS)$ if it exists;
- c) if the DLC is an UNORDERED or ORDERED DLC, then
 - 1) the DLE shall set $V_C(L)$ equal to K ;
 - 2) the DLE shall cancel all timers $T_{C,N}(RRS)$ which may exist, where N is less than K . All DLE resources devoted to reception and reassembly of DLSDUs with sequence numbers less than or equal to K should be released.
- d) if the DLC is a DISORDERED DLC, and if K equals $(V_C(L) + 1)$, then
 - 1) the DLE shall set $V_C(L)$ equal to K ;
 - 2) if K is less than $V_C(H)$, then the DLE shall increment K . If the set variable $V_{C,K}(MRS)$ is empty, then the DLE shall set $V_C(L)$ equal to K and shall repeat this step;
 - 3) if the DLC is a PEER DLC, then if the DLE's DL-address unscheduled-service queue, $Q(US)$, does not already contain a reference to the DLCEP, then the DLE shall append to that $Q(US)$ a reference to the DLCEP, to ensure that an acknowledgment of DLSDU receipt is sent from the receiving DLCEP;
- e) if the DLC is a CLASSICAL DLC, then
 - 1) the DLE shall set $V_C(L)$ equal to K ;
 - 2) if K is less than $V_C(H)$, then the DLE shall increment K . If the set variable $V_{C,K}(MRS)$ is empty, then the DLE shall repeat the entire data delivery procedure (see beginning of 10.2.2.7.2) using the new value of K ;
 - 3) if the DLC is a PEER DLC, then if the DLE's DL-address unscheduled-service queue, $Q(US)$, does not already contain a reference to the DLCEP, then the DLE shall append to that $Q(US)$ a reference to the DLCEP, to ensure that an acknowledgment of DLSDU receipt is sent from the receiving DLCEP.

10.2.2.7.3 OSI-default delivery

The DLE shall report a DL-DATA indication to the DLS-user, conveying the received DLSDU as a parameter; after which the DLE shall do as specified in 10.2.2.7.2b) to 10.2.2.7.2e).

10.2.2.8 Receipt of a DT DLPDU addressed to a DLCEP

In the following, MUD is defined to be the maximum number of user data octets which may be conveyed in a single DT DLPDU of the DLCEP's priority, as specified in 6.2.3.

When the DLE receives a DT DLPDU addressed to a DLCEP of the DLE, the DLE shall perform the following series of actions:

- a) If the DLCEP state, $V_C(ST)$, is WAITING-FOR-RESET-completion and the DLE is waiting only for receipt of a DT DLPDU at the DLCEP, then the DLE shall change the DLCEP state to DATA-TRANSFER-READY.
- b) The DLE shall validate that
 - 1) the priority of the received DT DLPDU is as expected;
 - 2) in a received DT DLPDU addressed to all subscribers of a PUBLISHER DLCEP, the length of the publisher's DL-address is greater than or equal to that expected;
 - 3) in a received DT DLPDU addressed to a PUBLISHER DLCEP, the number of DL-addresses is as expected;
 - 4) that in a received DT DLPDU addressed to a PEER DLCEP
 - i) the length and number of the DL-address(es) is as expected (only LONG, or only SHORT, or either SHORT or VERY-SHORT at the sender's option);
 - ii) when two addresses are expected, that the second DL-address of the DLPDU is the DLCEP-address of the remote peer of the DLCEP addressed by the DT DLPDU's first DL-address.

If this validation fails, then

- if the DLCEP is PEER DLCEP, the DLE shall disconnect the DLCEP from the DLC as specified in 10.2.1.8 with a reason of "provider-originated disconnection — wrong DLPDU format or parameters, permanent condition";
 - else the DLE shall discard the DT DLPDU.
- c) If the DLCEP is a PEER or SUBSCRIBER DLCEP whose negotiated residual-activity attribute is TRUE, then the DLE shall restart the DLCEP's $T_C(RAM)$ as specified in 10.2.2.14.
- NOTE See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.
- d) If the remaining number of octets in the DLPDU is less than the number of octets in the negotiated SD-parameters format for the applicable sender-to-receiver direction of transmission, then
 - if the DLCEP is PEER or SUBSCRIBER DLCEP, then the DLE shall disconnect the DLCEP from the DLC as specified in 10.2.1.8 with a reason of "provider-originated disconnection — wrong DLPDU format or parameters, permanent condition";
 - else the DLE shall discard the DT DLPDU.

Otherwise the DLE shall parse and process the applicable-format SD-parameters from those remaining octets as specified in 10.2.2.5; and if the remaining number of octets in the DLPDU, after the SD-parameters, is greater than zero, then the DLE shall process that user data as specified in 10.2.2.6 and possibly 10.2.2.7.

10.2.2.9 Receipt of a CA, CD or ED DLPDU

When the DLE receives a CA, CD or ED DLPDU addressed to a DLCEP of the DLE, the DLE shall perform the following series of actions:

- a) If the DLCEP is a PEER DLCEP then the DLE shall validate that
 - 1) the length and number of DL-addresses is as expected;
 - 2) when present and the receiving DLCEP is a PEER DLCEP, that the second DL-address of the DLPDU is the DLCEP-address of the remote peer of the receiving DLCEP;

3) the DLL priority of the DLCEP is equal to the priority specified in the received DLPDU.

If this validation fails, then the DLE shall disconnect the DLCEP from the DLC as specified in 10.2.1.8 with a reason of “provider-originated disconnection — wrong DLPDU format or parameters, permanent condition”.

- b) If the DLCEP is a PEER DLCEP whose negotiated residual-activity attribute is TRUE, then the DLE shall restart the DLCEP's $T_C(\text{RAM})$ with a duration based on the user-specified maximum confirm delay for DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be $\frac{V_C(\text{NP}).\text{MCD\_CRS}}{V(\text{NRC})+1}$; otherwise the duration should be $\frac{60 \text{ s}}{V(\text{NRC})+1}$. DL-management may override these preferred durations.

NOTE See the permission at the start of clause 10 with regard to use of $V(\text{NRC})$ versus $V(\text{MRC})$.

- c) If the received DLPDU is a CD DLPDU, then

- 1) if the DLCEP is a PEER DLCEP, and if the remaining number of octets in the DLPDU is not zero, and is not equal to the number of octets in the negotiated SD-parameters format for the applicable sender-to-receiver direction of transmission, then the DLE shall disconnect the DLCEP from the DLC as specified in 10.2.1.8 with a reason of “provider-originated disconnection — wrong DLPDU format or parameters, permanent condition”;

- 2) otherwise, when 1) does not apply, then

i) if the remaining number of octets in the DLPDU is equal to the number of octets in the negotiated SD-parameters format for the applicable sender-to-receiver direction of transmission, then the DLE shall parse and process the applicable-format SD-parameters from those remaining octets as specified in 10.2.2.5;

ii) when the receiving DLCEP's features are ORDERED, and the DLCEP is bound to a sending buffer, and the DLSDU in the buffer was too large to transmit completely in the immediate response to the received CD DLPDU, then the DLE shall

A) for the appropriate K , modify the variable $V_{C,K}(\text{SS})$, based on the length of the K 'th DLSDU, to indicate that all segments of the DLSDU except segment zero (0) require transmission;

B) append to the DLE's unscheduled-service queue, $Q(\text{US})$, a reference to the DLCEP's $Q_A(\text{UR})$. The reference signals the need to transmit the contents of the sending buffer which is bound to the DLCEP.

- d) If the received DLPDU is a CA or ED DLPDU, then

- 1) if the remaining number of octets in the DLPDU is equal to or greater than the number of octets in the negotiated SD-parameters format for the applicable sender-to-receiver direction of transmission; then the DLE shall

i) parse and process the applicable-format SD-parameters from those remaining octets as specified in 10.2.2.5;

ii) then process any remaining octets of the DLSDU as received DLS-user data as specified in 10.2.2.6 and 10.2.2.7;

- 2) if 1) does not apply and the DLCEP is a PEER DLCEP, then the DLE shall disconnect the DLCEP from the DLC as specified in 10.2.1.8 with a reason of “provider-originated disconnection — wrong DLPDU format or parameters, permanent condition”.

10.2.2.10 Starting, cancellation and expiration of the timer $T_U(\text{MCD})$ on a DL-DATA request

The timer $T_U(\text{MCD})$ shall be started when the DLS-user issues the corresponding DL-DATA request. It shall be cancelled

- at a subscriber DLCEP, or at a CLASSICAL or DISORDERED peer DLCEP, or at an ORDERED peer or publisher DLCEP when $V_C(R)$ does not exist, when the DLE issues the corresponding DL-DATA confirm;
- at a CLASSICAL or DISORDERED publisher DLCEP, or at an ORDERED peer or publisher DLCEP when $V_C(R)$ exists, when the DLE finds it necessary to increment $V_C(R)$ and the resultant value of $V_C(R)$ equals the sequence number assigned to the corresponding DLSDU.

If the timer $T_U(\text{MCD})$ expires on a DL-DATA request, then the DLE shall

- a) increment $V_C(R)$;

NOTE After incrementation, $V_C(R)$ should have a value equal to the sequence number of the DLSDU associated with the expired request.

- b) remove the request from the sending DLCEP-address's user-request queue, $Q_A(\text{UR})$, and terminate processing of the request;
- c) maintain any appropriate DL-management statistics;
NOTE Determination of the minimum necessary statistics is for future study.
- d) if a DL-DATA confirm primitive for the request has not yet been issued, then
 - 1) initiate a DL-DATA confirm reporting "provider-originated failure — request timeout";
 - 2) if the choice of sending DLCEP data delivery features is DISORDERED or CLASSICAL, then initiate a reset of the DLC as specified in 10.2.2.19.

10.2.2.11 Starting, cancellation and expiration of the timer $T_{C,K}(\text{SS})$

NOTE This timer is used only by PEER DLCEPs whose sending data delivery features are DISORDERED or CLASSICAL.

The timer $T_{C,K}(\text{SS})$ shall be started whenever a DLPDU containing all or part of DLSDU $_K$ is transmitted and $V_{C,K}(\text{SS})$ is empty; it shall be cancelled whenever $V_C(A)$ is greater than or equal to K or whenever $V_{C,K}(\text{SS})$ becomes non-empty (see 10.2.2.5.2b)).

The duration of this timer shall be based on the local user-specified maximum confirm delay for DL-DATA primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be between 25 % and 50 % of $\frac{V_C(\text{NP}).\text{MCD}_D}{V(\text{NRC})+2}$; otherwise the duration should be between 25 % and 50 % of $\frac{60 \text{ s}}{V(\text{NRC})+2}$. DL-management may override these preferred durations.

NOTE See the permission at the start of clause 10 with regard to use of $V(\text{NRC})$ versus $V(\text{MRC})$.

If the timer $T_{C,K}(\text{SS})$ expires, then the DLE shall

- a) modify the variable $V_{C,K}(\text{SS})$, based on the length of the K 'th DLSDU, to indicate that the last segment of the K 'th DLSDU needs retransmission;
- b) append to the DLE's unscheduled-service queue, $Q(\text{US})$, a reference to the DLCEP's $Q_A(\text{UR})$, to schedule a retransmission of the unacknowledged DLSDU;
- c) maintain any appropriate DL-management statistics.

NOTE Determination of the minimum necessary statistics is for future study.

10.2.2.11.1 Use of the simplified timer $T_C(SS)$

When the permission of 6.7.4.9.1 is employed, the following rules apply:

The timer $T_C(SS)$ shall be started, but not restarted, whenever a DLPDU containing all or part of $DLSDU_K$ is transmitted and $V_{C,K}(SS)$ is empty. The timer shall be restarted whenever it is not running and $V_C(A)$ is less than $V_C(M)$; it shall be cancelled whenever $V_C(A)$ equals $V_C(M)$ or whenever $V_{C,K}(SS)$ becomes non-empty due to receipt of a request for retransmission (see 10.2.2.5.2b)).

The duration of this timer shall be based on the local user-specified maximum confirm delay for DL-DATA primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be between 25 % and 50 % of $\frac{V_C(NP).MCD\_D}{V(NRC)+2}$; otherwise the duration should be between 25 % and 50 % of $\frac{60\text{ s}}{V(NRC)+2}$. DL-management may override these preferred durations.

NOTE See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.

If the timer $T_C(SS)$ expires, then the DLE shall

- modify the variable $V_{C,K}(SS)$, based on the length of the unacknowledged $DLSDU_K$ with the lowest sequence number, to indicate that the last segment(s) of that $DLSDU$ need retransmission;
- append to the DLE's unscheduled-service queue, $Q(US)$, a reference to the DLCEP's $Q_A(UR)$, to schedule a retransmission of the unacknowledged $DLSDU$;
- maintain any appropriate DL-management statistics.

10.2.2.12 Starting, cancellation and expiration of the timer $T_{C,K}(RRS)$

NOTE 1 This timer is used by DISORDERED or CLASSICAL subscriber DLCEPs, and optionally by ORDERED peer or subscriber DLCEPs.

The timer $T_{C,K}(RRS)$ shall be started whenever it is not running and a DLPDU requesting retransmission of one or more segments of the K 'th $DLSDU$ is transmitted. It shall be cancelled whenever all segments of the K 'th $DLSDU$ are received.

The duration of this timer shall be based on the remote user-specified maximum confirm delay for DL-DATA primitives and conveyed in an EC DLPDU previously received from the sending DLCEP. If the specified value was other than UNLIMITED, then the duration of this timer should be between 25 % and 50 % of $\frac{V_C(NP).MCD\_D}{V(NRC)+2}$; otherwise the duration should be between 25 % and 50 % of $\frac{60\text{ s}}{V(NRC)+2}$. DL-management may override these preferred durations.

NOTE 2 See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.

If the timer $T_{C,K}(RRS)$ expires, then the DLE shall

- set $V_{C,K}(RRS)$ equal to the then-current value of $V_{C,K}(MRS)$, to indicate that the specified segments of the K 'th $DLSDU$ need retransmission;
- check for a reference to the DLCEP on the DLE's unscheduled-service queue, $Q(US)$, and if not found then append a reference to the DLCEP to the DLE's unscheduled-service queue, $Q(US)$, to schedule a retransmission of the missing $DLSDU$ segments;
- maintain any appropriate DL-management statistics.

NOTE Determination of the minimum necessary statistics is for future study.

10.2.2.13 Starting, cancellation and expiration of the timer $T_C(RAS)$

NOTE 1 This timer is used only by PUBLISHER or sending PEER DLEs whose sending data delivery features are ORDERED, DISORDERED or CLASSICAL, and is only required when DLC establishment requested residual activity on the DLC in this (sender-to-receiver) direction of data transfer.

When applicable (see 6.7.4.16 for the conditions of the timer's use), the timer $T_C(RAS)$ shall be started

- a) at a sending DISORDERED or CLASSICAL PEER DLCEP, whenever it is not running and when $V_C(A)$ equals $(V_C(N) - 1)$;
- b) at a PUBLISHER DLCEP, or sending ORDERED PEER DLCEP, whenever it is not running and when $V_C(M)$ equals $(V_C(N) - 1)$ and $V_C, V_C(M)(SS)$ is empty.

The duration of this timer shall be based on the user-specified maximum confirm delay for DL-CONNECT request or response primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be between 70 % and 95 % of $\frac{V_C(NP).MCD\_CRS}{V(NRC)+2}$; otherwise the duration should be between 70 % and 95 % of $\frac{60\text{ s}}{V(NRC)+2}$. DL-management may override these preferred durations.

NOTE 2 See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.

It shall be cancelled whenever $V_C(A)$ is not equal to $(V_C(N) - 1)$. If the timer $T_C(RAS)$ expires, then the DLE shall check for a reference to the DLCEP on the DLE's unscheduled-service queue, $Q(US)$, and if not found then append a reference to the DLCEP to the DLE's unscheduled-service queue, $Q(US)$, to schedule a transmission to the remote DLCEP(s).

NOTE 3 The resulting transmission may consist of a DT DLPDU with a null user data field, or a CA DLPDU.

10.2.2.14 Starting, cancellation and expiration of the timer $T_C(RAM)$

NOTE 1 This timer is used only by SUBSCRIBER or receiving PEER DLEs whose receiving data delivery features are ORDERED, DISORDERED or CLASSICAL, and is only required when DLC establishment requested residual activity on the DLC in this (receiver-from-sender) direction of data transfer.

When applicable (see 6.7.4.17 for the conditions of the timer's use), the timer $T_C(RAM)$ shall run continuously. It shall be restarted whenever any DLPDU is received on the DLCEP.

The duration of this timer shall be based on the remote user-specified maximum confirm delay for DL-CONNECT request or response primitives and conveyed in an EC DLPDU previously-received from the sending DLCEP. If the specified value was UNLIMITED, then the duration of this timer should be 60 s. Otherwise, the duration should be $V_C(NP).MCD\_CRS$. DL-management may override these preferred durations.

If the timer $T_C(RAM)$ expires, then the DLE shall initiate a reset of the DLCEP as specified in 10.2.2.19.

10.2.2.15 Receipt of a DL-RESET request primitive

When the DLE receives a DL-RESET request from a DLS-user for a DLCEP in the DATA-TRANSFER-READY state, then

- a) if the DLCEP class is not PEER, the DLE shall issue a confirm primitive with a status of "failure — reason unspecified" for each outstanding DL-DATA request primitive on the local DLCEP, and shall increment $V_C(R)$ for each such confirmed primitive;
- b) the DLE shall
 - 1) set $V_C(M)$ equal to $V_C(N) - 1$; set $V_C(MS)$ and $V_C(HS)$ equal to zero;
 - 2) release all DLE resources devoted to reception and reassembly of DLSDUs;
 - 3) cancel all timers $T_C, K(SS)$ and $T_C, K(RRS)$.
- c) if the DLCEP-class is PEER, then the DLE shall
 - 1) note that both
 - i) a DL-RESET confirm primitive;
 - ii) receipt of an RC DLPDU not requesting a reply RC DLPDU from the remote peer DLCEP

will be required;

- 2) encode an RC DLPDU, addressed to the remote peer DLCEP and requesting a reply RC DLPDU, with the reason specified by the DLS-user, using $V_C(L) + 1$ for the NDR sequence number and $V_C(M)$ for the NDS sequence number;
 - 3) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5;
 - 4) start a timer as specified in 10.2.1.2b);
 - 5) change the DLCEP state, $V_C(ST)$, to WAITING-FOR-RESET-COMPLETION, while waiting for an RC DLPDU.
- d) if the DLCEP-class is PUBLISHER, then the DLE shall
- 1) encode an RC DLPDU, addressed to all of the DLC's subscriber's DLCEPs, not requesting a reply RC DLPDU, with the reason specified by the DLS-user, using zero (0) for the NDR sequence number and $V_C(M)$ for the NDS sequence number,
 - 2) set $V_C(A)$ or $V_C(R)$, as appropriate, equal to $V_C(N) - 1$;
 - 3) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5;
 - 4) report a DL-RESET confirm to the local DLS-user with a status parameter of "success".
- e) If the DLCEP-class is SUBSCRIBER, then the DLE shall report a DL-RESET confirm to the local DLS-user with a status parameter of "success".

10.2.2.16 Receipt of a DL-RESET response primitive

When the DLE receives a DL-RESET response from a DLS-user for an active DLCEP whose state is WAITING-FOR-RESET-COMPLETION then

- a) if the DLCEP-class is PEER, and the set of events at the DLCEP for which the DLE is waiting is
- 1) only a DL-Reset response, then the DLE shall
 - i) encode a DT DLPDU (with or without data) from the DLCEP to notify its peer of the successful receipt of the confirming RC DLPDU;
 - ii) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5;
 - iii) again flush the DLCEP's internal queues;
 - iv) change the DLCEP state, $V_C(ST)$, to DATA-TRANSFER-READY;
 - v) report a DL-RESET-COMPLETED indication to the local DLS-user;
 - 2) both
 - i) a DL-Reset response;
 - ii) receipt of an RC DLPDU not requesting a reply from the peer DLCEP,
 then the DLE shall wait for that RC DLPDU;
 - 3) both
 - i) a DL-Reset response;
 - ii) subsequent receipt of a DT DLPDU from the peer DLCEP,

then the DLE shall

- A) encode an RC DLPDU, addressed to the remote peer DLCEP and not requesting a reply RC DLPDU, with a reason of "reset, unknown origin — reason unspecified", using $V_C(L) + 1$ for the NDR sequence number and $V_C(M)$ for the NDS sequence number;

- B) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5;
 - C) start a timer as specified in 10.2.1.2b);
 - D) note that a DL-RESET-COMPLETED indication to the local DLS-user is still required;
 - E) wait for a DT DLPDU.
- b) if the DLCEP-class is PUBLISHER or SUBSCRIBER, then the DLE shall
- 1) change the DLCEP state, $V_C(ST)$, to DATA-TRANSFER-READY;
 - 2) again flush the DLCEP's internal queues;
 - 3) report a DL-RESET-COMPLETED indication to the local DLS-user.

10.2.2.17 Receipt of an RC DLPDU

When the DLE receives an RC DLPDU, the DLE shall determine the version of the DL-protocol in use, as specified in the received RC DLPDU, and shall interpret the other RC-parameters of the DLPDU accordingly.

- a) If the DLCEP's DLCEP-class is PEER or SUBSCRIBER, when the DLE receives an RC DLPDU specifying that the DLCEP should be reset, then the DLE shall take action depending on the DLCEP state, $V_C(ST)$:

- 1) waiting-for-EC-DLPDU or waiting-for-connect-response or waiting-for-connect-completion:

The DLE shall

- i) disconnect the DLCEP as specified in 10.2.1.8 with a reason of "connection rejection — inconsistent DLCEP state, — permanent condition".

- 2) data-transfer-ready:

The DLE shall

- i) report a DL-RESET indication to the local DLS-user with a reason parameter equal to that in the received RC DLPDU;
- ii) if the DLCEP class is PEER and $V_C(A)$ is less than the sequence number implied by the NDR subfield of the RC parameters of the received RC DLPDU (which is the smallest sequence number $\geq V_C(A)$ whose modulus would be NDR), then the DLE shall process outstanding DL-DATA requests on the local DLCEP as specified in 10.2.2.5.2, with NDR as received in the RC DLPDU, $J=0$ and $K=1$;
- iii) if the DLCEP's DLCEP-class is PEER, the DLE shall issue a confirm primitive with a status of "failure — reset or disconnection" for each outstanding DL-DATA request primitive on the local DLCEP, and shall increment $V_C(R)$ for each such confirmed primitive; and set $V_C(A)$ or $V_C(R)$, as appropriate, equal to $V_C(N) - 1$;
- iv) set $V_C(L)$ and $V_C(H)$ equal to the sequence number implied by the NDS subfield of the RC-parameters of the received RC DLPDU (which is the smallest sequence number $\geq V_C(H)$ whose modulus would be NDS);
- v) set $V_C(M)$ equal to $V_C(N) - 1$; set $V_C(MS)$ and $V_C(HS)$ equal to zero;
- vi) release all DLE resources devoted to reception and reassembly of DLSDUs;
- vii) cancel all timers $T_{C,K(SS)}$ and $T_{C,K(RRS)}$;
- viii) change the DLCEP state, $V_C(ST)$, to WAITING-FOR-RESET-COMPLETION;
- ix) wait for a DL-RESET response plus, if the DLCEP class is PEER, subsequent receipt of a DT DLPDU indicating peer reset completion;

3) waiting-for-reset-completion:

- i) If the DLCEP-class is PEER and $V_C(A)$ is less than the sequence number implied by the NDR subfield of the RC-parameters of the received RC DLPDU (which is the smallest sequence number $\geq V_C(A)$ whose modulus would be NDR), then the DLE shall process outstanding DL-DATA requests on the local DLCEP as specified in 10.2.2.5.2, with NDR as received in the RC DLPDU, $J=0$ and $K=1$.
- ii) If the DLCEP's DLCEP-class is PEER, the DLE shall issue a confirm primitive with a status of "failure — reset or disconnection" for each outstanding DL-DATA request primitive on the local DLCEP, and shall increment $V_C(R)$ for each such confirmed primitive; and set $V_C(A)$ or $V_C(R)$, as appropriate, equal to $V_C(N) - 1$.
- iii) The DLE shall set $V_C(L)$ and $V_C(H)$ equal to the sequence number implied by the NDS subfield of the RC-parameters of the received RC DLPDU (which is the smallest sequence number $\geq V_C(H)$ whose modulus would be NDS).
- iv) If the DLCEP-class is SUBSCRIBER, then the DLE shall discard the received RC DLPDU.
- v) If the DLCEP-class is PEER, and the DLE is waiting for a DL-RESET response primitive at the DLCEP, then
 - if the received RC DLPDU specified that a reply RC DLPDU is required, then the DLE shall note that an RC DLPDU is required, else the DLE shall note that a DT DLPDU is required.
- vi) If the DLCEP-class is PEER, and the DLE is not waiting for a DL-RESET response primitive at the DLCEP, and the received RC DLPDU requested a reply, then the DLE shall
 - A) encode an RC DLPDU, addressed to the remote peer DLCEP and not requesting a reply RC DLPDU, with a reason of "reset, unknown origin — reason unspecified", using $V_C(L) + 1$ for the NDR sequence number and $V_C(M)$ for the NDS sequence number;
 - B) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5;
 - C) start a timer as specified in 10.2.1.2b);
 - D) if the reset-completion primitive expected by the DLS-user at the DLCEP is DL-RESET confirm, then the DLE shall note that an RC DLPDU is required, else the DLE shall note that a DT DLPDU is required.
- vii) If the DLCEP-class is PEER, and the DLE is not waiting for a DL-RESET response primitive at the DLCEP, and the received RC DLPDU did not request a reply, then the DLE shall
 - A) encode a DT DLPDU (with or without data) from the DLCEP to notify its peer of the successful receipt of the confirming RC DLPDU;
 - B) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5;
 - C) again flush the DLCEP's internal queues;

D) change the DLCEP state, $V_C(ST)$, to DATA-TRANSFER-READY;

E) if the reset-completion primitive expected by the DLS-user at the DLCEP is DL-RESET confirm, then the DLE shall report a DL-RESET confirm to the local DLS-user, else the DLE shall report a DL-RESET-COMPLETED indication to the local DLS-user.

viii) When none of ii) to v) applies, then

A) the DLE shall encode a DC DLPDU as specified in 8.2 and , with a reason of “provider-originated disconnection or connection rejection — reason unspecified”, and schedule the DC DLPDU for transmission at TIME-AVAILABLE priority as specified in 10.4.5;

B) the DC DLPDU shall have both destination and source addresses (see 8.2.1 formats 1L and 1S), the destination address shall be identical to the first source DLCEP-address of the received RC DLPDU, and the source address shall be identical to the destination DL-address of that received RC DLPDU.

4) In all other cases the DLE shall act as in a)3)v).

b) If the DLCEP's DLCEP-class is PUBLISHER, then the DLE shall act as in a)3)v).

10.2.2.18 Expiration of the timer $T_U(MCD)$ on a DL-RESET request or indication

If the timer $T_U(MCD)$ expires on a DL-RESET request or DLE-initiated reset, then

a) if this is the $(V(NRC)+1)$ 'th consecutive expiration without returning to the DATA-TRANSFER-READY state, then the DLE shall terminate DLCEP-reset processing and shall disconnect the DLCEP from the DLC as specified in 10.2.1.8 with a reason of “provider-originated disconnection — timeout, permanent condition”;

NOTE See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.

b) otherwise, if a) does not apply, then the DLE shall

1) restart the timer with the same period as the previous time;

2) reschedule the RC DLPDU (which was first sent just after exiting the DATA-TRANSFER-READY state) for retransmission at the DLC's priority as specified in 10.4.5.

10.2.2.19 DLE-initiated reset

When the DLE initiates a reset, then

a) the DLE shall change the DLCEP state, $V_C(ST)$, to WAITING-FOR-RESET-COMPLETION;

b) the DLE shall note the need for

1) receipt of a DL-RESET response from the DLS-user;

2) if the DLCEP class is PEER, subsequent receipt of an RC DLPDU indicating peer reset completion;

c) if the DLCEP's DLCEP-class is PEER, then the DLE shall start a user-request timer $T_U(MCD)$ with a duration based on the user-specified maximum confirm delay for DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be $\frac{V_C(NP).MCD\_CRS}{V(NRC)+1}$; otherwise the duration should be $\frac{60\text{ s}}{V(NRC)+1}$. DL-management may override these preferred durations;

NOTE See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.

d) if the DLCEP's DLCEP-class is not PEER, the DLE shall issue a confirm primitive with a status of “failure — reset or disconnection” for each outstanding DL-DATA request primitive on the local DLCEP, shall increment $V_C(R)$ for each such confirmed primitive, and shall set $V_C(A)$ or $V_C(R)$, as appropriate, equal to $V_C(N) - 1$;

- e) the DLE shall
 - 1) set $V_C(M)$ equal to $V_C(N) - 1$; set $V_C(MS)$ and $V_C(HS)$ equal to zero;
 - 2) release all DLE resources devoted to reception and reassembly of DLSDUs;
 - 3) cancel all timers $T_{C,K(SS)}$ and $T_{C,K(RRS)}$;
- f) the DLE shall issue a DL-RESET indication primitive to the local DLS-user, specifying the reason for the reset;
- g) if the DLCEP class is PEER, then the DLE shall
 - 1) encode an RC DLPDU, with the same reason as the reason given to the local DLS-user and requesting a reply RC DLPDU, using $V_C(L) + 1$ for the NDR sequence number and $V_C(M)$ for the NDS sequence number;
 - 2) queue that RC DLPDU for transmission at the DLC's priority as specified in 10.4.5;
- h) if the DLCEP-class is PUBLISHER, then the DLE shall
 - 1) encode an RC DLPDU, addressed to all of the DLC's subscriber's DLCEPs, not requesting a reply RC DLPDU, with the same reason as the reason given to the local DLS-user, using zero (0) for the NDR sequence number and $V_C(M)$ for the NDS sequence number, and
 - 2) schedule the DLPDU for transmission at the DLC's priority as specified in 10.4.5.

10.2.3 Operation of the DLC subscriber query service

The primitives of the DLC subscriber query service are DL-SUBSCRIBER-QUERY request and confirm.

10.2.3.1 Receipt of a DL-SUBSCRIBER-QUERY request primitive

If the request is accepted, as indicated by a returned status of "request accepted" for the DL-SUBSCRIBER-QUERY request, then upon completion of the request, either successfully or after failure, the DLE shall issue a DL-SUBSCRIBER-QUERY confirm at the DLCEP, conveying the status of the request to the DLS-user.

The DL-SUBSCRIBER-QUERY request shall be queued on the DLE's NODE DL-address user-request queue, $Q_N(UR)$, which is an implicit (DLE-controlled) queue.

- a) If that queue is full, then the DLE shall immediately return the corresponding DL-LISTENER-QUERY confirm indicating the reason for failure — "provider-originated failure — queue full".
- b) If a) does not apply, then the DLE shall start a user-request timer $T_U(MCD)$ with a duration based on the user-specified maximum confirm delay for DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY primitives. If the specified value was other than UNLIMITED, then the duration of this timer should be $\frac{V_C(NP).MCD\_CRS}{V(NRC)+1}$; otherwise, the duration should be $\frac{60\text{ s}}{V(NRC)+1}$. DL-management may override these preferred durations.

NOTE See the permission at the start of clause 10 with regard to use of $V(NRC)$ versus $V(MRC)$.

- c) If a) does not apply, then the DLE shall
 - append the request to the DLE's NODE user-request queue, $Q_N(UR)$, at TIME-AVAILABLE priority, where the request shall be placed in the second partition;
 - NOTE The third partition in the DLE's $Q_N(UR)$ is always empty.
 - append a reference to $Q_N(UR)$ to the DLE's unscheduled-service queue, $Q(US)$.

10.2.3.2 Transmission of a DL-address query SPDU

Upon receipt of a transmission opportunity for the queued DL-SUBSCRIBER-QUERY request, the DLE shall

- a) form a DL-address query SPDU (see 11.3.6.2) specifying the requested DL-address;
- b) include that SPDU as the DLSDU of a connectionless format-1s DT DLPDU of TIME-AVAILABLE priority with a SHORT destination DL-address for the DL-support functions of all DLE's on the local link, 0100₁₆ (see 6.3.3.2), and source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions.

10.2.3.3 Receipt of a DL-address query SPDU specifying a subscriber DLCEP-address

If a DLE receives a DL-address query SPDU/DLPDU for a DL-address and DL-address-class which is active within the DLE, then the DLE shall respond at a TIME-AVAILABLE priority with a corresponding DL-address reply SPDU/DLPDU, confirming the received DL-address query.

10.2.3.4 Receipt of a DL-address reply SPDU

If the DLE receives a DL-address reply SPDU/DLPDU in response to the DL-address query SPDU/DLPDU which was transmitted as a result of the DL-SUBSCRIBER-QUERY request, then the DLE shall

- a) cancel the timer $T_U(\text{MCD})$;
- b) initiate a DL-SUBSCRIBER-QUERY confirm reporting "success — a subscriber exists";
- c) release the request identifier, making it available for subsequent re-use.

10.2.3.5 Expiration of the timer $T_U(\text{MCD})$ on a DL-SUBSCRIBER-QUERY request

If the timer $T_U(\text{MCD})$ expires on a DL-SUBSCRIBER-QUERY request,

- a) if this is the $(V(\text{NRC})+1)$ 'th consecutive expiration, then the DLE shall

NOTE See the permission at the start of clause 10 with regard to use of V(NRC) versus V(MRC).

 - 1) terminate processing of the request;
 - 2) initiate a DL-SUBSCRIBER-QUERY confirm reporting "provider-originated failure — request timeout";
- b) otherwise, if a) does not apply, then the DLE shall
 - 1) restart the timer with the same period as the previous time;
 - 2) requeue the SPDU/DLPDU for retransmission as in 10.2.3.2b).

10.3 Operation of the connectionless-mode services

The connectionless-mode services are connectionless data transfer with local-DLE-confirmation, connectionless data transfer with remote-DLE-confirmation, connectionless data exchange, and listener query.

10.3.1 Operation of the connectionless data transfer with local-DLE-confirmation service

The primitives of the connectionless data transfer with local-DLE-confirmation service are DL-UNITDATA request, indication and confirm.

10.3.1.1 Receipt of a DL-UNITDATA request primitive not specifying remote-DLE-confirmation

When the DLE receives a DL-UNITDATA request primitive not specifying remote-DLE-confirmation, it shall associate the DLS-user-specified request identifier with the request. If the request is rejected, then the DLE shall issue a DL-UNITDATA confirm with the same

request identifier, conveying the status of the request to the DLS-user. If the request is accepted, then upon completion of the requested transmission, either successfully or after failure, the DLE shall issue a DL-UNITDATA confirm with the same request identifier, conveying the status of the request to the DLS-user.

- a) If the DL(SAP)-role of that source DL(SAP)-address is BASIC, then the calling-DL(SAP)-address specified in the DL-UNITDATA request is bound as a source at the specified priority to either an explicit (user-controlled) or implicit (DLE-controlled) queue.
- b) If the DL(SAP)-role of that source DL(SAP)-address is not BASIC, or if that queue is full, or if the specified DLSDU has an invalid length, then the DLE shall immediately return the reason for failure (in a DL-UNITDATA confirm primitive) as the status of the DL-UNITDATA request primitive.
- c) If the called-DL(SAP)-address specified in the DL-UNITDATA request is not a DL(SAP)-address whose DL(SAP)-role is BASIC or GROUP, then the request is erroneous. The DLE is permitted, but is not required, to detect such an error (for example, by detecting that the called address is not a DL(SAP)-address). If such an error is detected, then the DLE shall immediately return the reason for failure (in a DL-UNITDATA confirm primitive) as the status of the DL-UNITDATA request primitive.
- d) Otherwise, if neither b) nor c) applies, then
 - 1) the DLE shall create and start a user-request timer $T_U(\text{MCD})$ with a duration based on the user-specified maximum confirm delay for the DL-UNITDATA request primitive. If the specified value was other than UNLIMITED, then the duration of this timer should be equal to that user-specified maximum confirm delay, $P_U(\text{MCD})$; otherwise the duration should be 60 s. DL-management may override these preferred durations.
 - 2) the DLE shall append the request to the calling DLSAP-address's user-request queue, $Q_A(\text{UR})$, as follows:
 - i) If the DL-scheduling-policy attribute of the calling DLSAP-address is EXPLICIT, then the request shall await a DL-COMPEL-SERVICE request to release it for transmission and so shall be placed in the third partition of the user-request queue, $Q_A(\text{UR})$.
 - ii) Otherwise, if i) does not apply, then the request shall be placed in the second partition and the DLE shall append to the DLE's unscheduled-service queue, $Q(\text{US})$, a reference to $Q_A(\text{UR})$ of the same priority as the just-appended request.

10.3.1.2 Transmission of a unitdata DT DLSDU

Upon receipt of a transmission opportunity for the queued DLSDU, the DLE shall

- a) remove the request from the appropriate DLSAP-address user-request queue, $Q_A(\text{UR})$;
- b) form and send a DT DLPDU of the specified priority, with the specified called and calling DL(SAP)-addresses, with a null SD-parameters field (format P (see a))), and with a user data field whose length and contents equal the specified DLSDU;

NOTE With the exception of the final-token-use subfield of the DT DLPDU's first octet, the DT DLPDU may be formed when the request is queued, and need not be formed dynamically at the moment of transmission.
- c) issue a DL-UNITDATA confirm with the associated request identifier reporting "success";
- d) cancel and delete the associated timer $T_U(\text{MCD})$.

10.3.1.3 Receipt of a DT DLPDU, with an explicit source address, addressed to a DL(SAP)-address

When the DLE receives a DT DLPDU with an explicit source address addressed to a DL(SAP)-address bound to one or more of the DLE's DLSAPs, then if the DL(SAP)-role for that destination DL(SAP)-address is

- a) INITIATOR or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER, then the received DLPDU is erroneous and the DLE
 - 1) shall inform local DL-management of the event, possibly including the erroneous DLPDU's addresses;
 - 2) shall discard the DLPDU.
- b) BASIC or GROUP, then for each of those bindings
 - 1) the DLE shall attempt to append the received user data as a DLSDU, together with the called and calling DL(SAP)-addresses and DLL priority of the received DLPDU, to
 - the receiving queue which was bound at the received priority level, or
 - to the implicit OSI queue if no receiving buffer or queue was bound at that received priority level;
 - 2) if 1) is successful, then the DLE shall initiate a DL-UNITDATA indication primitive at that DLSAP;
 - 3) otherwise, if 1) is not successful, then the DLE shall inform local DL-management of the queue-full situation.

NOTE This DL-management notification may take the form of incrementing a counter of discarded DLSDUs.

10.3.1.4 Expiration of the timer $T_U(MCD)$ on a DL-UNITDATA request not specifying remote-DLE-confirmation

If the timer $T_U(MCD)$ expires on a DL-UNITDATA request not specifying remote-DLE-confirmation, then the DLE shall

- a) remove the request from the appropriate DLSAP-address user-request queue, $Q_A(UR)$, and the reference to the request from the DLE's unscheduled-service queue, $Q(US)$;
- b) initiate a DL-UNITDATA confirm with the associated request identifier reporting "failure — timeout before transmission".

10.3.2 Operation of the connectionless data transfer service with remote-DLE-confirmation

The primitives of the connectionless data transfer with remote-DLE-confirmation service are DL-UNITDATA request, indication and confirm.

10.3.2.1 Receipt of a DL-UNITDATA request primitive specifying remote-DLE-confirmation

When the DLE receives a DL-UNITDATA request primitive specifying remote-DLE-confirmation, it shall associate the DLS-user-specified request identifier with the request. Upon completion of the request, either successfully or after failure, the DLE shall issue a DL-UNITDATA confirm with the same request identifier, conveying the status of the request to the DLS-user.

The calling DL(SAP)-address specified in the DL-UNITDATA request is bound as a source at the specified priority to either an explicit (user-controlled) or implicit (DLE-controlled) queue. If the DL(SAP)-role of that source DL(SAP)-address is not BASIC, or if that queue is full, or if the specified DLSDU has an invalid length, then the DLE shall immediately return the reason for failure (in a DL-UNITDATA confirm primitive) as the status of the DL-UNITDATA request primitive.

If the called-DL(SAP)-address specified in the DL-UNITDATA request is not a DLSAP-address whose DL(SAP)-role is BASIC, then the request is erroneous. The DLE is permitted, but not required, to detect such an error (for example, by detecting that the called address is not a DLSAP-address). If such an error is detected during request processing, then the DLE shall return an appropriate error status in a DL-UNITDATA confirm primitive, indicating the reason for failure, and shall terminate processing of the request.

Otherwise

- a) The DLE shall create and start a user-request timer $T_U(\text{MCD})$ with a duration based on the user-specified maximum confirm delay for the DL-UNITDATA request primitive. If the specified value was other than UNLIMITED, then the duration of this timer should be equal to that user-specified maximum confirm delay, $P_U(\text{MCD})$; otherwise the duration should be 60 s. DL-management may override these preferred durations.
- b) The DLE shall append the request to the calling DLSAP-address's user-request queue, $Q_A(\text{UR})$, as follows:
 - 1) If the DL-scheduling-policy attribute of the calling DLSAP-address is EXPLICIT, then the request shall await a DL-COMPEL-SERVICE request to release it for transmission and so shall be placed in the third partition of the user-request queue, $Q_A(\text{UR})$.
 - 2) Otherwise the request shall be placed in the second partition of the user-request queue, $Q_A(\text{UR})$, and the DLE shall append to the DLE's unscheduled-service queue, $Q(\text{US})$, a reference to $Q_A(\text{UR})$ of the same priority as the just-appended request.

10.3.2.2 Transmission of a unitdata CA DLPDU

Upon receipt of a transmission opportunity for the queued DLSDU, the DLE shall form and send a CA DLPDU of the specified priority,

- a) with the specified called and calling DLSAP-addresses;
- b) with an SD-parameters field of format R (see b));
- c) with a user data field whose length and contents equal the specified DLSDU.

NOTE With the exception of the final-token-use subfield of the CA DLPDU's first octet, and possibly of the initiator's transaction index subfield, $N(\text{LTI})$, of the CA DLPDU's SD-parameters (see b)3)), the CA DLPDU may be formed when the request is queued, and need not be formed dynamically at the moment of transmission.

10.3.2.3 Receipt of a CA DLPDU, with an explicit source address, addressed to a DL(SAP)-address

When the DLE receives a CA DLPDU with an explicit source address addressed to a DL(SAP)-address bound to one or more of the DLE's DLSAPs, then if the DL(SAP)-role for that destination DL(SAP)-address is

- a) GROUP, then the received DLPDU is erroneous, and
 - 1) the DLE shall inform local DL-management of the event, possibly including the erroneous DLPDU's addresses;
 - 2) the DLE shall discard the received CA DLPDU.
- b) INITIATOR or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER, then the received DLPDU is erroneous and
 - 1) the DLE shall inform local DL-management of the event, possibly including the erroneous DLPDU's addresses;
 - 2) the DLE shall form and send, as an immediate reply, a reply DT DLPDU, as specified in 8.4.4.1 and 8.7, whose reported status indicates "failure — responder DL(SAP)-role incompatible with this DLPDU";
 - 3) the DLE shall discard the received CA DLPDU.
- c) BASIC, then
 - 1) the DLE shall attempt to append the received user data as a DLSDU, together with the called and calling DLSAP-addresses and DLL priority of the received DLPDU, to the receiving queue which was explicitly or implicitly bound at the received priority level;
 - 2) if 1) is successful, then the DLE shall initiate a DL-UNITDATA indication primitive at that DLSAP and shall indicate its success in 4);
 - 3) otherwise, if 1) is not successful, then the DLE shall indicate the reason for its failure in 4), and shall notify local DL-management of that failure;

NOTE This DL-management notification may take the form of incrementing a counter of discarded DLSDUs.

- 4) the DLE shall form and send, as an immediate reply, a reply DT DLPDU, as specified in 8.4.4.1 and 8.7, whose reported status indicates the success 2) or failure 3).

10.3.2.4 Receipt of a DT DLPDU, with an implicit source address, addressed to a DL(SAP)-address

When the DLE receives a DT DLPDU, with an implicit source address, addressed to a DL(SAP)-address bound to one or more of the DLE's DLSAPs, then if the DL(SAP)-role for that destination DL(SAP)-address is

- a) CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER or GROUP, then the received DLPDU is erroneous;
 - 1) the DLE shall inform local DL-management of the event, possibly including the erroneous DLPDU's addresses;
 - 2) the DLE shall discard the received DT DLPDU.
- b) INITIATOR, then the DLE shall act as specified in 10.3.3.3.
- c) BASIC, then
 - 1) If the transaction index subfield, N(LTI), of the SD-parameters of the received DLPDU specifies a transaction index for an outstanding incomplete request, then the DLE
 - i) shall release that transaction index, making it available for subsequent reuse;
 - ii) shall remove the request from the appropriate DLSAP-address user-request queue, $Q_A(UR)$;
 - iii) shall cancel and delete the timer $T_U(MCD)$ associated with the request;
 - iv) shall issue a DL-UNITDATA confirm to the requesting DLS-user, indicating the request's DLS-user-identifier and reporting the status conveyed by the received DT DLPDU.
 - 2) Otherwise, if 1) does not apply, then the DLE shall notify local DL-management of the receipt of an inappropriate DLPDU.

NOTE This DL-management notification may take the form of incrementing a counter of discarded DLSDUs.

10.3.2.5 Receipt of an SR DLPDU

When the DLE receives an SR DLPDU,

- which reports a status other than "BOK — bridge OK" (that is, has a value other than E_{16});
- which is received as a reply to an immediately prior CA, CD or ED DLPDU which was originated by the receiving DLE and which was addressed to a DLSAP-address;
- where the DL(SAP)-role for the DLSAP-address from which the transaction initiated (that is, the source DLSAP-address of the initiating CA, CD or ED DLPDU) is BASIC,

then the receiving DLE shall

- a) release the transaction index, N(LTI), used in the immediately prior CA, CD or ED DLPDU, making it available for subsequent reuse;
- b) remove the request from the appropriate DLSAP-address user-request queue, $Q_A(UR)$;
- c) cancel and delete the timer $T_U(MCD)$ associated with the request;
- d) issue a DL-UNITDATA confirm to the requesting DLS-user, indicating the request's DLS-user-identifier and reporting the status conveyed by the received SR DLPDU.

10.3.2.6 Expiration of the timer $T_U(\text{MCD})$ on a DL-UNITDATA request specifying remote-DLE-confirmation

If the timer $T_U(\text{MCD})$ expires on a DL-UNITDATA request specifying remote-DLE-confirmation, then the DLE

- a) shall release the corresponding transaction index and remove it from active use, making it available for subsequent reuse only after a period of twice the maximum network DLPDU lifetime, $V(\text{NDL})$, of the extended link;
- b) shall remove the request from the appropriate DLSAP-address user-request queue, $Q_A(\text{UR})$;
- c) shall delete the timer $T_U(\text{MCD})$;
- d) shall initiate a DL-UNITDATA confirm to the requesting DLS-user, indicating the request's DLS-user-identifier and reporting a status of "failure — timeout before transmission".

10.3.3 Operation of the connectionless data exchange service

10.3.3.1 Transmission of a unitdata CD or ED DLSDU

Upon receipt of a transmission opportunity for a reference to a DL-address user-request queue, $Q_A(\text{UR})$, for a DLSAP-address whose DL(SAP)-role is INITIATOR, the DLE shall examine the set of sending buffer bindings for that DLSAP-address in priority order as follows:

- a) If
 - there is a sending binding at URGENT priority to a buffer, and
 - the buffer is non-empty,
 then the DLS-user data contained in the URGENT buffer becomes the DLSDU selected for transmission, and the priority of that DLSDU is URGENT; or
- b) If a) does not apply, and
 - the transaction-priority specified in the invoking reference is normal or time-available;
 - there is a sending binding at normal priority to a buffer;
 - the buffer is non-empty,
 then the DLS-user data contained in the NORMAL buffer becomes the DLSDU selected for transmission, and the priority of that DLSDU is NORMAL; or
- c) If neither a) nor b) applies, and
 - the transaction-priority specified in the invoking reference is TIME-AVAILABLE;
 - there is a sending binding at TIME-AVAILABLE priority to a buffer;
 - the buffer is non-empty,
 then the DLS-user data contained in the TIME-AVAILABLE buffer becomes the DLSDU selected for transmission, and the priority of that DLSDU is TIME-AVAILABLE; or
- d) If none of a) to c) applies, then no DLSDU is selected for transmission, then the DLE shall form and send an appropriate DLPDU:
- e) If any of a) to c) applies, then a DLSDU was selected by this procedure, and
 - 1) the DLE shall form and send an ED DLPDU as specified in 8.6.3
 - i) with the destination address equal to the remote DLSAP-address specified in the invoking reference;
 - ii) with the source address equal to the DLSAP-address associated with the user-request queue, $Q_A(\text{UR})$, specified by that reference;
 - iii) with a priority equal to the transaction-priority specified in the invoking

- reference;
- iv) with an SD-parameters field of format R (see b)), specifying the actual priority of the selected DLSDU;
 - v) with a user data field whose length and contents equal the selected DLSDU;
- 2) the DLE shall await an immediate response. If
- i) the DLSDU which was sent to the responder in the ED DLPDU was obtained from a non-retentive buffer (BUFFER-NR);
 - ii) an appropriate DT or SR DLPDU is received as an immediate reply, as specified in 8.5.3, 8.5.4, 8.6.3, 8.6.4, 8.7.3 and 8.8.3;
 - iii) the explicit or implicit status conveyed by that reply is either “success” or “interim success”,
- then the buffer shall be set empty.
- f) If d) applies, then no DLSDU was selected by this procedure, and
- 1) the DLE shall form and send a CD DLPDU as specified in 8.5.3
 - i) with the destination address equal to the remote DLSAP-address specified in the invoking reference;
 - ii) with the source address equal to the DLSAP-address associated with the user-request queue, Q_A(UR), specified by that reference;
 - iii) with a priority equal to the transaction-priority specified in the invoking reference;
 - iv) with an SD-parameters field of format R (see b)), specifying that there is no accompanying DLSDU;
 - 2) the DLE shall await an immediate response.

10.3.3.2 Receipt of a CD or ED DLPDU, with an explicit source address, addressed to a DL(SAP)-address

When the DLE receives a CD or ED DLPDU with an explicit source address addressed to a DL(SAP)-address bound to one or more of the DLE’s DLSAPs, then if the DL(SAP)-role for that destination DL(SAP)-address is

- a) GROUP, then the received DLPDU is erroneous, and
 - 1) the DLE shall inform local DL-management of the event, possibly including the erroneous DLPDU’s addresses;
 - 2) the DLE shall discard the received CD or ED DLPDU.
- b) BASIC or INITIATOR, then the received DLPDU is erroneous, and
 - 1) the DLE shall inform local DL-management of the event, possibly including the erroneous DLPDU’s addresses;
 - 2) the DLE shall form and send, as an immediate reply, a reply DT DLPDU, as specified in 8.5.4.1, 8.6.4.1 and 8.7, whose reported status indicates “failure — responder DL(SAP)-role incompatible with this DLPDU”;
 - 3) the DLE shall discard the received CD or ED DLPDU.
- c) constrained responder, then
 - 1) if the source address of the received CD or ED DLPDU equals the value of the remote-DLSAP-address parameter which was specified in the most recent DL-BIND request primitive for the responding DLSAP-address, then the DLE shall act as in d);
 - 2) otherwise, when 1) does not apply, the received DLPDU is erroneous and the DLE shall act as in b) except that the status returned in b)2) shall be “failure — response restricted to a different peer DLSAP-address”.

d) unconstrained responder, then

1) if the received DLPDU is an ED DLPDU and thus conveys a non-null DLSDU, then

i) if

- the responding DLSAP-address has an explicit binding as receiver at the priority of the received DLPDU;
- the binding is to a buffer;
- the size of the buffer is at least as great as the size of the DLSDU,

then the DLE shall overwrite the buffer with the just-received DLSDU.

ii) If

- the responding DLSAP-address has an explicit binding as receiver at the priority of the received DLPDU;
- the binding is to a queue;
- the size of each queue record is at least as great as the size of the DLSDU;
- the queue is not full,

then the DLE shall append the just-received DLSDU to the queue.

iii) if either i) or ii) applies, then the DLE shall examine the set of sending buffer bindings for that DLSAP-address in priority order as follows:

A) if

- there is a sending binding at URGENT priority to a buffer;
- the buffer is non-empty,

then the DLS-user data contained in the URGENT buffer becomes the DLSDU selected for transmission, and the priority of that DLSDU is URGENT; or

B) if A) does not apply, and

- the transaction-priority specified in the received DLPDU is NORMAL or TIME-AVAILABLE;
- there is a sending binding at NORMAL priority to a buffer;
- the buffer is non-empty,

then the DLS-user data contained in the NORMAL buffer becomes the DLSDU selected for transmission, and the priority of that DLSDU is NORMAL; or

C) if neither A) nor B) applies, and

- the transaction-priority specified in the received DLPDU is TIME-AVAILABLE;
- there is a sending binding at TIME-AVAILABLE priority to a buffer;
- the buffer is non-empty,

then the DLS-user data contained in the TIME-AVAILABLE buffer becomes the DLSDU selected for transmission, and the priority of that DLSDU is TIME-AVAILABLE; or

D) if none of A) to C) applies, then no DLSDU is selected for transmission.

iv) if either i) or ii) applies, then iii) applies, and

A) the DLE shall form and send as an immediate reply an appropriate DT DLPDU as specified in 8.7;

B) if a DLSDU was sent to the initiator in a replying DT DLPDU, and that DLSDU was obtained from a non-retentive buffer (BUFFER-NR), then the buffer shall be set empty;

C) the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the responding DLS-user, reporting

- the responding DLSAP-address's DLS-user-/DL-identifier;

- the priority of the DLS-user data conveyed by the received ED DLPDU, or that no such DLS-user data was received (in a CD DLPDU);
 - the priority of the DLS-user data conveyed in the reply DT DLPDU, or that no such DLS-user data was sent;
 - a status of “success”.
- v) if neither i) nor ii) applies, then the received DLSDU was discarded, and
- A) the DLE shall form and send as an immediate reply an appropriate DT DLPDU as specified in 8.7;
 - B) the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the responding DLS-user, reporting
 - the responding DLSAP-address’s DLS-user-/DL-identifier;
 - the priority of the DLS-user data conveyed by the received ED DLPDU which was discarded;
 - a status of “failure”.
- 2) if the received DLPDU is a CD DLPDU and thus did not convey a DLSDU, then
- i) the DLE shall examine the set of sending buffer bindings for that DLSAP-address in priority order as in d)1)iii);
 - ii) the DLE shall act as in d)1)iv)A) and d)1)iv)B);
 - iii) if
 - a DLSDU was selected in ii), or
 - the indicate-null-UNITDATA-EXCHANGE-transactions parameter which was specified in the most recent DL-BIND request primitive for the responding DLSAP-address had the value TRUE,
 then the DLE shall act as in d)1)iv)C) except that the reported status shall be “failure — no DLS-user data exchanged”.

10.3.3.3 Receipt of a DT DLPDU, with an implicit source address, addressed to a DL(SAP)-address

When the DLE receives a DT DLPDU, with an implicit source address, addressed to a DL(SAP)-address bound to one or more of the DLE’s DLSAPs, then if the DL(SAP)-role for that destination DL(SAP)-address is

- a) BASIC or CONSTRAINED RESPONDER or UNCONSTRAINED RESPONDER or GROUP, then the DLE shall act as specified in 10.3.2.4.
- b) INITIATOR, and if the transaction index subfield, N(LTI), of the SD-parameters of the received DLPDU specifies a transaction index for an outstanding incomplete request, then
 - 1) the DLE shall release that transaction index, making it available for subsequent reuse;
 - 2) the DLE shall cancel and delete the timer $T_U(MCD)$ associated with the now-complete unitdata-exchange transaction;
 - 3) if the invoking reference was in the DLE’s unscheduled-service queue, Q(US), or had been dynamically appended to a specified scheduled sequence, then the DLE shall remove that reference from that queue or sequence appendage;
 - 4) if the received DT DLPDU contains DLS-user data and thus conveys a non-null DLSDU, then
 - i) if
 - the initiating DLSAP-address has an explicit binding as receiver at the priority of the received DLPDU;
 - the binding is to a buffer;
 - the size of the buffer is at least as great as the size of the DLSDU,

the DLE shall overwrite the buffer with the just-received DLSDU;

ii) if

- the initiating DLSAP-address has an explicit binding as receiver at the priority of the received DLPDU;
- the binding is to a queue;
- the size of each queue record is at least as great as the size of the DLSDU;
- the queue is not full,

the DLE shall append the just-received DLSDU to the queue;

iii) if either i) or ii) applies, then

A) if a DLSDU which was sent to the responder in an ED DLPDU, and that DLSDU was obtained from a non-retentive buffer (BUFFER-NR), then the buffer shall be set empty.

B) the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the initiating DLS-user, reporting

- the initiating DLSAP-address's DLS-user-/DL-identifier;
- the priority of the DLS-user data conveyed by the previously sent ED DLPDU, or that no such DLS-user data was sent (in a CD DLPDU);
- the priority of the DLS-user data conveyed by the received DLPDU;
- a status of "success";

iv) if neither i) nor ii) applies, then the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the initiating DLS-user, reporting

- the initiating DLSAP-address's DLS-user-/DL-identifier;
- the priority of the DLS-user data conveyed by the previously-sent ED DLPDU, or that no such DLS-user data was sent (in a CD DLPDU);
- the priority of the DLS-user data conveyed by the received DLPDU;
- a status of "failure — resource limitation in initiator";

5) if 4) does not apply, and the DLPDU sent to the responding DLE was an ED DLPDU, thus conveying a non-null DLSDU, then

i) If the DLSDU which was sent to the responder in the ED DLPDU was obtained from a non-retentive buffer (BUFFER-NR), then the buffer shall be set empty;

ii) the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the initiating DLS-user, reporting

- the initiating DLSAP-address's DLS-user/DL-identifier;
- the priority of the DLS-user data conveyed by the previously-sent ED DLPDU, or that no such DLS-user data was sent (in a CD DLPDU);
- the status conveyed by the received DT DLPDU;

6) if neither 4) nor 5) applies, and the DLPDU sent to the responding DLE was a CD DLPDU, then no DLSDUs were exchanged. In this case, if the indicate-null-UNITDATA-EXCHANGE-transactions parameter which was specified in the most recent DL-BIND request primitive for the initiating DLSAP-address had the value TRUE, then the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the initiating DLS-user, reporting the initiating DLSAP-address's DLS-user-/DL-identifier and a status of "failure — no DLS-user data exchanged";

c) INITIATOR, and if the transaction index subfield, N(LTI), of the SD-parameters of the received DLPDU does not specify a transaction index for an outstanding incomplete request, then the DLE shall notify local DL-management of the receipt of an inappropriate DLPDU.

NOTE This DL-management notification may take the form of incrementing a counter of discarded DLSDUs.

10.3.3.4 Receipt of an SR DLPDU

When the DLE receives an SR DLPDU,

- which reports a status other than “BOK — bridge OK” (that is, has a value other than E_{16});
- which was received as a reply to an immediately prior CA, CD or ED DLPDU which was originated by the receiving DLE and which was addressed to a DLSAP-address;
- where the DL(SAP)-role for the DLSAP-address from which the transaction initiated (that is, the source DLSAP-address of the initiating CA, CD or ED DLPDU) is INITIATOR,

then

- a) the receiving DLE shall release the transaction index, $N(LTI)$, used in the immediately prior CA, CD or ED DLPDU, making it available for subsequent reuse;
- b) the receiving DLE shall cancel and delete the timer $T_U(MCD)$ associated with the request;
- c) if the invoking reference was in the DLE’s unscheduled-service queue, $Q(US)$, or had been dynamically appended to a specified scheduled sequence, then the DLE shall remove that reference from that queue or sequence appendage;
- d) if the indicate-null-UNITDATA-EXCHANGE-transactions parameter which was specified in the most recent DL-BIND request primitive for the initiating DLSAP-address had the value TRUE, then the DLE shall issue a DL-UNITDATA-EXCHANGE indication to the initiating DLS-user, reporting the initiating DLSAP-address’s DLS-user-/DL-identifier and a status of “failure — no DLS-user data exchanged”.

10.3.3.5 Expiration of the timer $T_U(MCD)$ on an incomplete instance of the unitdata exchange service

If the timer $T_U(MCD)$ expires on an instance of the unitdata-exchange service, then the DLE

- a) shall release the corresponding transaction index and remove it from active use, making it available for subsequent re-use only after a period of twice the maximum DLPDU lifetime, $V(MDL)$, of the extended link;
- b) shall delete the timer $T_U(MCD)$;
- c) shall issue a DL-UNITDATA-EXCHANGE indication to the initiating DLS-user, specifying the initiating DLSAP-address’s DLS-user-/DL-identifier and reporting a status of “provider-originated failure — request timeout”.

10.3.4 Operation of the listener query service

The primitives of the listener query service are DL-LISTENER-QUERY request and confirm.

10.3.4.1 Receipt of a DL-LISTENER-QUERY request primitive

When the DLE receives a DL-LISTENER-QUERY request, it shall associate the DLS-user-specified request identifier with the request. If the request is accepted, as indicated by a returned status of “request accepted” for the DL-LISTENER-QUERY request, then upon completion of the request, either successfully or after failure, the DLE shall issue a DL-LISTENER-QUERY confirm with the same request identifier, conveying the status of the request to the DLS-user.

The DL-LISTENER-QUERY request shall be queued on the DLE’s NODE DL-address user-request queue, $Q_N(UR)$, which is an implicit (DLE-controlled) queue.

- a) If that queue is full, then the DLE shall immediately return the corresponding DL-LISTENER-QUERY confirm indicating the reason for failure — “provider-originated failure — queue full”.
- b) If a) does not apply, then the DLE shall start a user-request timer $T_U(MCD)$ with a duration based on the user-specified maximum confirm delay for the DL-LISTENER-QUERY request

primitive. If the specified value was other than UNLIMITED, then the duration of this timer should be $\frac{P_U(MCD)}{V(NRC)+1}$; otherwise the duration should be $\frac{60\text{ s}}{V(NRC)+1}$. DL-management may override these preferred durations.

NOTE See the permission at the start of clause 10 with regard to use of V(NRC) versus V(MRC).

c) If a) does not apply, then the DLE shall

— append the request to the DLE's NODE user-request queue, $Q_N(UR)$, at TIME-AVAILABLE priority, where the request shall be placed in the second partition;

NOTE The third partition in the DLE's $Q_N(UR)$ is always empty.

— append a reference to $Q_N(UR)$ to the DLE's unscheduled-service queue, $Q(US)$.

10.3.4.2 Transmission of a DL-address query SPDU

Upon receipt of a transmission opportunity for the queued DL-LISTENER-QUERY request, the DLE shall form and send a DL-address query SPDU as specified in 10.2.3.2.

10.3.4.3 Receipt of a DL-address query SPDU specifying a group DL-address

If a DLE receives a DL-address query SPDU (conveyed by a DT DLPDU) for a DL-address and DL-address -class which is active within the DLE, then the DLE shall

- a) form a replying DT DLPDU containing a corresponding DL-address reply SPDU, confirming the received DL-address query;
- b) append the DT DLPDU to the DLE's NODE user-request queue, $Q_N(UR)$, at the relevant priority;
- c) append a reference to $Q_N(UR)$ to the DLE's unscheduled-service queue, $Q(US)$.

10.3.4.4 Receipt of a DL-address reply SPDU

If the DLE receives a DL-address reply SPDU (conveyed by a DT DLPDU) as an apparent response to the DL-address query SPDU/DLPDU which was transmitted as a result of the DL-LISTENER-QUERY request, then the DLE shall

- a) cancel the timer $T_U(MCD)$;
- b) initiate a DL-LISTENER-QUERY confirm with the corresponding request identifier reporting "success — a listener exists".

10.3.4.5 Expiration of the timer $T_U(MCD)$ on a DL-LISTENER-QUERY request

If the timer $T_U(MCD)$ expires on a DL-LISTENER-QUERY request, then

- a) If this is the $(V(NRC)+1)$ 'th consecutive expiration, then the DLE shall

NOTE See the permission at the start of clause 10 with regard to use of V(NRC) versus V(MRC).

- 1) terminate processing of the request;
- 2) initiate a DL-LISTENER-QUERY confirm with the corresponding request identifier reporting "provider-originated failure — request timeout".

- b) Otherwise, if a) does not apply, then the DLE shall

- 1) restart the timer with the same period as the previous time;
- 2) requeue the same DT DLPDU for retransmission as specified in 10.3.4.3b).

10.4 Operation of the scheduling guidance services

The scheduling guidance services are the DL-time service, the compel-service service, the sequence scheduling service, and the sequence subsetting service.

10.4.1 Operation of the DL-time service

The only primitive of the DL-time service is DL-TIME request.

10.4.1.1 Receipt of a DL-TIME request primitive

When the DLE receives a DL-TIME request, it shall respond with a five-component record as shown in Table 78, whose values are

- a) the current value of the time source's link-id, V(TSL);
- b) the quality of the current DL-time, V(TQ), as defined in b);
- c) the current value of the local DL-time-offset, V(DLTO);
- d) the current value of the local-link-scheduling-time-offset, V(LSTO);
- e) the current value of the local node-time, C(NT).

The DLS-user may then sum the last three values to determine the node's current DL-time, or use the values separately, as appropriate.

Table 78 – Components of returned DL-time

Octet index	Contents of field
1 2	V(TSL)
3	V(TQ)
4 5 6 7 8 9 10	V(DLTO)
11 12 13 14 15 16 17	V(LSTO)
18 19 20 21 22 23 24	C(NT)

If the DLE's time-synchronization class (see 12.6.3) is NONE and the DLE does not maintain even an estimated C(NT), then it may respond with the values 0...0 for V(TQ), V(DLTO), V(LSTO) and C(NT), and FFFF₁₆ for V(TSL); this latter, which is an invalid link-id, indicates to the DLS-user the DLE's incapability.

10.4.1.2 Transmission of a TD DLPDU

After sending a TD DLPDU, the LAS DLE shall restart its local timer T(TDP), to expire at a time no greater than $0,95 \times V(TDP)$ in the future (see 6.7.1.26 and 6.7.1.26).

10.4.1.3 Receipt of a TD DLPDU

If the DLE's time-synchronization class (see 12.6.3) is NONE, then

- a) If the DLE which is not the LAS DLE supports the variables, timers and counters defined in 6.7.1.19 to 6.7.1.27, then the DLE shall set V(TSL) equal to $N_S(TSL)$, set V(DLTO) equal to $N_S(DLTO)$, and set V(LSTO) equal to

$$(N_S(DLT) + N_S(DLTA) - N_S(DLTO) - N_R(NT) + \frac{V(MD)}{2}) \text{ modulo } 2^{56}. \quad (20)$$

- b) Otherwise, when a) does not apply, then the DLE shall ignore receipt of the TD DLPDU.

When a DLE other than the LAS DLE, with a time-synchronization class other than NONE, receives a TD DLPDU, then

- c) the DLE shall restart its local timer T(TDP), to expire at a time such that if it does not receive a TD DLPDU until after the expiration of this timer, it can still maintain time synchronization within the limits implied by the local link's time-synchronization-class (see 6.7.1.25). The value of this timer shall be larger than the minimum specified for the time-synchronization-class in 12.6.3, but the exact value may be determined locally by the DLE.

- d) the DLE shall note that the DLE does not have to send a CT.

NOTE Variance in the expiration periods of the timers T(TDP) among DLEs, and cancellation upon receipt of a TD DLPDU of such requests for transmission of a CT DLPDU, both serve to reduce the likelihood that a single link-wide noise event, inhibiting correct reception of a TD DLPDU, will result in multiple transmissions of CT DLPDUs.

- e) the DLE shall update

- its locally-maintained sense of the link-id of that link, within the extended-link, which is the time source, V(TSL);
- its locally-maintained sense of the local LAS node, V(LN);
- its locally-maintained sense of the quality of DL-time, V(TQ);
- its DL-time-offset, V(DLTO);
- its local-link-scheduling-time-offset, V(LSTO);
- its node-timer frequency

by comparing the local variables V(TQ), V(TSL) and V(LN) with the corresponding fields of the received TD DLPDU — $N_S(TQ)$ and $N_S(TSL)$ and the DLPDU's source NODE DL-address — to detect changes in either the reference source for the time or in the time distribution path.

- 1) If the TD DLPDU's source NODE DL-address differs from the receiving DLE's V(LN), then

i) the DLE shall set V(LN) equal to the source NODE DL-address from the received TD DLPDU;

ii) the DLE shall inform local DL-management that a change of time source on the local link has occurred;

iii) if the DLE requires a measurement of round-trip-delay between itself and the local LAS, to maintain the required accuracy for its time-synchronization class, then the DLE shall invalidate V(MD), the prior round-trip-delay-measurement with the current LAS.

- 2) If the DLE requires a measurement of round-trip-delay between itself and the local LAS, and if the current value for that round-trip delay, $V(MD)$, is known to be invalid, then the DLE shall form an RQ DLPDU addressed to $V(LN).0$, and append the DLPDU as specified in 10.4.5, at NORMAL priority to the DLE's NODE DL-address user-request queue, $Q_N(UR)$.

NOTE This is not the queue normally used with the procedures of 10.4.5.

- 3) If the measurement of round-trip-delay to the current LAS, $V(MD)$, is valid, then the DLE shall set its $V(TQ)$ from the node-time-quality value, $N_S(TQ)$, in the received DLPDU, after adjustment of the number-of-intervening-links subfield for the one extra intervening link between the sending and receiving DLEs (by incrementing $N_S(TQ).LLL$ when forming $V(TQ)$ to account for the receiving DLE).

NOTE Therefore a DLE with its own time source, which is not also serving as the source of DL-time to the local link, must ignore its own time source and synchronize with the other DLEs on the local link.

- 4) If the time-source-link value, $N_S(TSL)$, in the received DLPDU differs from the receiving DLE's $V(TSL)$, and the round-trip-delay-measurement with the current LAS, $V(MD)$, is valid, then the DLE shall

- i) set $V(TSL)$ equal to $N_S(TSL)$;
- ii) set all other local references to the time-source-link, such as the link-id of the source of time distribution of periodic scheduled activities in 12.3.7), equal to $N_S(TSL)$;
- iii) inform local DL-management that a change of time source on the extended link has occurred.

- 5) If the DL-time-offset, $N_S(DLTO)$, in the received DLPDU differs from the receiving DLE's $V(DLTO)$, then the DLE shall

- i) set $V(DLTO)$ equal to $N_S(DLTO)$;
- ii) adjust all other local references based on DL-time, such as the periodic schedule DL-time base (T_0) in 12.3.7), by the amount that $V(DLTO)$ just changed, to reflect the new DL-time base;
- iii) inform local DL-management that a change in $V(DLTO)$ has occurred.

- 6) If the measurement of round-trip-delay to the current LAS, $V(MD)$, is valid, then the DLE shall compute

$$TEMP = (N_S(DLT) + N_S(DLTA) - N_S(DLTO) - V(LSTO) - N_R(NT)) \text{ modulo } 2^{56} \quad (21)$$

NOTE The above equation is equivalent in form to the computations in a).

In the following computations, the DLE shall use the value of K specified in Table 79.

Table 79 – Time synchronization computation

Time-synchronization class	Value of K for use in e)4)
1 μ s	4
10 μ s	41
100 μ s	410
1 ms	4 096
10 ms	40 960
100 ms	409 600
1 s	4 096 000

i) If $\text{abs}(TEMP)$, the absolute value of $TEMP$, is greater than K , and this has not been the case for the prior two received TD DLPDUs, then the DLE shall

A) set $V(LSTO)$ equal to $(TEMP + V(LSTO))$ modulo 2^{56} ;

B) inform local DL-management that a discontinuous change in $V(LSTO)$ has occurred.

ii) If $\text{abs}(TEMP)$, the absolute value of $TEMP$, is less than or equal to K , or if it was greater than K for the prior two received TD DLPDUs, then the DLE shall use the computed value, $TEMP$, together with prior computed values as appropriate, to adjust the frequency of incrementation of the DLE's $C(NT)$

A) to reduce the long-term precession of $V(LSTO)$, relative to $N_S(DLT) + N_S(DLTA) - N_S(DLTO)$, to zero;

B) to keep the long-term counting rate of $C(NT)$ to increment by approximately 8 192 000 ($2^{13} \times 10^3$) counts/s.

This frequency adjustment shall be maintained until a newer adjustment is computed.

NOTE The details of the filtering algorithm which determines this frequency adjustment are intentionally left to the implementor.

10.4.1.3.1 Additional actions required of a bridge

If the TD DLPDU was received at an active port in the forwarding state (see clause 11) which is the bridge DLE's root port (that is, the active port toward the root of the bridge spanning tree), then the DLE shall forward the DLPDU reception event to the bridge operation level (see 6.1.2). The bridge operation level shall share the received sense of time with

- those other active ports of the bridge which are currently serving as the LAS on their own local links;
- all bridge ports which are not in the forwarding or listening states;
- any local DLS-user.

The following also apply.

a) If the bridge has any port other than its root port which is acting as LAS, then the bridge shall

- 1) update the DL-time-offset, $V(DLTO)$, for each of those other ports, such that the sum of $C(NT)$, $V(LSTO)$ and $V(DLTO)$ for that non-root port equals the sum of $C(NT)$, $V(LSTO)$ and $V(DLTO)$ for the root port;

NOTE The DL-time-offset, $V(DLTO)$, and local-link-scheduling-time-offset, $V(LSTO)$, are port-specific (as are most of the variables of 6.7.5, and thus are maintained independently for each port).

- 2) if 1) caused a change in the DL-time-offset, $V(DLTO)$, then adjust the periodic schedule DL-time base, T_0 , in 12.3.7) by the same amount that $V(DLTO)$ just changed, to reflect the new DL-time base;
- 3) if there was a change in the time-source-link, $V(TSL)$, then set the link-id of the source of time distribution of periodic scheduled activities in 12.3.7), equal to the new value of $V(TSL)$.

b) The bridge operation level may schedule the sending of additional TD DLPDUs on those other ports which are acting as LAS.

c) The contents of the time subfields of those later DLPDUs shall be based on the then-current values for the sending port's internally-maintained node-time variables, $C(NT)$, $V(LSTO)$ and $V(DLTO)$, rather than the value of the DL-time parameters from the just-received TD DLPDU.

- d) The contents of the time-source and time-quality subfields — V(TSL) and N(TQ) — of those later DLPDUs shall be based on the corresponding fields from the just-received TD DLPDU, except that the sent N(TQ) shall reflect both
- any impact of the DLE's timeliness class on the resolution of the forwarded time;
 - the fact that there is one additional bridge in the forwarding path.

10.4.1.4 Receipt of an RQ DLPDU

When the DLE receives an RQ DLPDU, it shall form and send as an immediate reply a ROUND-TRIP-DELAY REPLY (RR) DLPDU

- a) whose destination DL-address equals the source address of the received RQ DLPDU;
- b) whose source address is formed by concatenating the sender's node-id, V_S(TN), and an octet of zero;
- c) whose first two parameter subfields equal the parameter subfields, respectively, of the received and appended-to RQ DLPDU.

10.4.1.5 Receipt of an RR DLPDU

When the DLE receives an RR DLPDU, if the DLE maintains C(NT), then it shall, within some reasonable period not exceeding 60 s, compute the round-trip delay, RTD, as a function of the four parameter subfields of the received and appended-to RR DLPDU, as:

$$\text{RTD} = (\text{short-time}_4 - \text{short-time}_3 + \text{short-time}_2 - \text{short-time}_1) \text{ modulo } 2^{24} \quad (22)$$

where the fields are numbered in their order of transmission (or appending) within the DLPDU. If the remote DLE in the measurement was the current LAS DLE, V(LN), then the computed delay shall be saved as the measured-delay

$$V(\text{MD}) = \text{RTD} \quad (23)$$

which thereby becomes valid for use in C(NT) frequency-adjustment computations, as specified in equation (15) (see 10.4.1.3).

The results of several round-trip measurements with the same remote node may be combined to reduce the mean error in V(MD).

10.4.1.6 Expiration of the timer T(TDP)

If the timer T(TDP) expires, then

- a) if the DLE is the local link's LAS, then the DLE shall note the requirement to transmit a TD DLPDU at the first opportunity;
- b) otherwise the DLE shall note the requirement to transmit a CT DLPDU at the first opportunity.

10.4.2 Operation of the compel-service service

The only primitive of the compel-service service is DL-COMPEL-SERVICE request.

10.4.2.1 Receipt of a DL-COMPEL-SERVICE request primitive

When the DLE receives a DL-COMPEL-SERVICE request, it shall classify the request and take the appropriate corresponding action. If the request is for

- a) a local (to the DLE) DLSAP-address whose DL(SAP)-role is BASIC, or a peer or publisher or subscriber DLCEP, for which the DL-scheduling-policy is EXPLICIT, then

- 1) if the DL-address is bound to a sending queue, and the number of DLSDUs in the third partition of that DL-address's user-request queue, $Q_A(UR)$, which are waiting for a DL-COMPEL-SERVICE request primitive is non-zero, then the DLE shall

- i) adjust the local DL-address's user-request queue, $Q_A(UR)$, by reducing that number (of DLSDUs in the third partition) by one;

- ii) reassign the highest-priority request, from that third partition to the second partition, unless such movement would cause the sending window size of the addressed DLCEP to be exceeded;

NOTE 1 The restriction in ii) above applies only to peer and publisher DLCEPs, and never to DLSAP-addresses.

NOTE 2 Implementations are permitted to defer the actual reassignment until the occurrence of a transmit opportunity for the DLSDU being reassigned. Such deferral may permit a subsequently-requested higher-priority DLSDU to be reassigned in lieu of the DLSDU that would have been reassigned at the moment when the DL-COMPEL-SERVICE request was made. However, in no case may an implementation accumulate a "quota of DL-COMPELLED requests" which extends beyond the next time that the third partition of the queue $Q_A(UR)$ becomes empty.

- iii) form a **reference** to that DL-address's $Q_A(UR)$ at the priority specified in the DL-COMPEL-SERVICE request, where the **reference** indicates the need to send a DLSDU from the sending queue identified in 1), and append the **reference** to either

- A) the specified scheduled sequence, if the DL-COMPEL-SERVICE request specified a sequence-identifier, or

- B) the DLE's unscheduled-service queue, $Q(US)$, if no Sequence-identifier was specified in the request;

- iv) return an immediate status of "success".

Otherwise, when no reassignment occurred, the DLE shall return an immediate status of "user failure — no DLSDU to release".

- 2) If the DL-address is bound to a sending buffer, then the DLE shall

- i) modify the variable $V_{C,K(SS)}$, for the appropriate K corresponding to the DLSDU currently associated with the buffer, to indicate that all segments of the DLSDU require transmission;

- ii) form a **reference** to the $Q_A(UR)$ of the specified local peer or publisher DLCEP, at the DLCEP's priority, where the **reference** indicates the need to send a DLSDU from the sending buffer identified in 2), and append the **reference** to either

- A) the specified scheduled sequence, if the DL-COMPEL-SERVICE request specified a sequence-identifier, or

- B) the DLE's unscheduled-service queue, $Q(US)$, if no Sequence-identifier was specified in the request;

- iv) return an immediate status of "success".

- b) the remote peer or publisher DLCEP of a local peer or subscriber DLCEP, respectively, then the DLE shall

- 1) form a **reference** to the $Q_A(UR)$ of the specified local peer or subscriber DLCEP, at the DLCEP's priority, where the **reference** indicates the need to compel the transmission of a DLSDU from the remote correspondent peer or publisher DLCEP identified in b), and append the **reference** to either

- i) the specified scheduled sequence, if the DL-COMPEL-SERVICE request specified a sequence-identifier, or

- ii) the DLE's unscheduled-service queue, $Q(US)$, if no sequence-identifier was specified in the request;

NOTE The above reference is distinguishable from references which result in local release of DLSDUs.

- 2) return an immediate status of "success";

- c) a local (to the DLE) DLSAP-address whose DL(SAP)-role is INITIATOR, then the DLE shall
 - 1) form a **reference** to that DL-address's $Q_A(UR)$ at the priority specified in the DL-COMPEL-SERVICE request, where the **reference** indicates the need to compel an instance of the unitdata-exchange service with the specified remote DLSAP-address, and append the **reference** to either
 - i) the specified scheduled sequence, if the DL-COMPEL-SERVICE request specified a sequence-identifier, or
 - ii) the DLE's unscheduled-service queue, $Q(US)$, if no sequence-identifier was specified in the request;
 - 2) return an immediate status of "success".
- d) Some other DL-address, then the DLE shall return an immediate status of "user failure — invalid DL-address".

10.4.3 Operation of the sequence scheduling service

The primitives of the sequence scheduling service are DL-SCHEDULE-SEQUENCE request and confirm, and DL-CANCEL-SCHEDULE request and confirm.

The ability of a DLE to respond positively to a DL-SCHEDULE-SEQUENCE request is determined by both the DLE's own abilities to service a dynamic (that is, not pre-configured by DL-management) request, and the current LAS DLE's ability to support that scheduling process. The conformance claimed in clause 11 gives some information on the DLE's capabilities.

10.4.3.1 Receipt of a DL-SCHEDULE-SEQUENCE request primitive

When the DLE receives a DL-SCHEDULE-SEQUENCE request, it shall assign a schedule identifier to the request and return that identifier as part of the DL-SCHEDULE-SEQUENCE request primitive.

Upon completion of the scheduling request, either successfully or after failure, the DLE shall issue a DL-SCHEDULE-SEQUENCE confirm with the same schedule identifier, conveying the status of the scheduling request to the DLS-user. If the scheduling request failed, then the DLE shall release the schedule identifier, making it available for subsequent reuse.

NOTE 1 L-SCHEDULE-SEQUENCE confirm indicates completion of the *scheduling* of the associated sequence, either locally or in conjunction with the local LAS; it does not indicate completion of the *execution* of that sequence. An implementation which defers issuing DL-SCHEDULE-SEQUENCE confirm primitives until after the completion of their scheduled sequences is erroneous.

The sequence itself is defined in terms of primitive sequence components specified in IEC 61158-3:2003⁸, 10.5.3.2.2, and in clause 11 to this standard.

NOTE 2 The specific representations of these primitive sequence components within an end-system is a local-view issue, outside the scope of open system standardization. However, clause 11 does include a standardized representation for some of these components when they are communicated between end-systems and an LAS DLE, and implementations may choose to use that same representation locally. Formal DL-programming-interface specifications, which would include the details of such an encoding, could standardize this or a similar local representation.

The DLE shall determine the class of scheduling operation requested, based on the sequence definition and the type of schedule requested.

⁸ Expected date of concurrent publication.

If the Schedule Type parameter has the value ONE-TIME or PERIODIC, then

- a) if the sequence consists of a single element, and the Schedule Type parameter has the value ONE-TIME, and the Desired Starting Time parameter specifies IMMEDIATE, and the sequence-priority parameter specifies a priority less than or equal to the intrinsic priority of the single element, then the DLE shall append a reference to the requested sequence at the Sequence Priority to the DLE's unscheduled-service queue, Q(US);
- b) else if a) does not apply and the sequence consists of one or more elements, all of which are references to DLCEPs remote from the DLE, none of which are for DLCs specifying EXTRA DLPDU-authentication, then the DLE shall
 - 1) form an SPDU equivalent to the sequence, of the same priority as the sequence;
 - 2) include that SPDU as the DLSDU of a connectionless DT DLPDU of TIME-AVAILABLE priority with a SHORT destination DL-address of the local LAS, 0400₁₆ (see 6.3.3.2), and source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions;
 - 3) append the DT DLPDU to the DLE's NODE user-request queue, Q_N(UR), at the relevant priority;
 - 4) append a reference to Q_N(UR) to the DLE's unscheduled-service queue, Q(US);
- c) else if a) and b) do not apply then the DLE shall
 - 1) use the DLS-user-provided DLSEP-address, or allocate a DLSEP-address if one was not provided;
 - 2) associate the requested sequence to the DLSEP-address, so that receipt of an ES DLPDU specifying that DLSEP-address will cause execution of the associated sequence;
 - 3) compute the maximum uninterrupted duration of the request;
 - 4) form a scheduling-request SPDU with priority equal to the Sequence-Priority of the DLS-user request; where the SPDU requests that the token be delegated to the sequence at the specified DLSEP-address, for the computed duration;
 - 5) include that SPDU in a DT DLPDU and queue that DLPDU for transmission as specified in b)2) to b)4).

If the Schedule Type parameter has the value REPETITIVE, then

- d) the DLE shall append a reference to the requested sequence at the Sequence Priority to the DLE's unscheduled-service queue, Q(US).

It is permitted, but not desirable, for a sequence that meets the requirements of b) to be executed locally as in a), c) or d). It is also permitted, but not desirable, for a sequence that meets the requirements of a) or c) to be sent as an SPDU as in b) to the LAS for remote execution; this permission requires that the SPDU syntax (see clause 11) be capable of conveying the necessary information.

NOTE Application of a DLS-request confirm delay to a DL-SCHEDULE-SEQUENCE request primitive is for future study.

10.4.3.2 Receipt of a DL-CANCEL-SCHEDULE request primitive

When the DLE receives a DL-CANCEL-SCHEDULE request, it shall determine the class of scheduled sequence being cancelled:

- a) If the sequence is locally scheduled (see 10.4.3.1a) or d)), then the DLE shall
 - 1) remove the reference to the schedule from the unscheduled-service queue, Q(US);
 - 2) issue a DL-CANCEL-SCHEDULE confirm with the identifier of the cancelled schedule;
 - 3) release the schedule identifier, making it available for subsequent reuse;

- b) If the sequence is centrally scheduled but locally executed (see 10.4.3.1c)), then the DLE shall
 - 1) remove the reference to the schedule from the DLSEP-address associated with the schedule;
 - 2) form a cancel-schedule SPDU (see 11.3.4.4);
 - 3) include that SPDU as the DLSDU of a connectionless DT DLPDU of TIME-AVAILABLE priority with a SHORT destination DL-address of the local LAS, 0400₁₆ (see 6.3.3.2), and source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions;
 - 4) append the DT DLPDU to the DLE's NODE user-request queue, Q_N(UR), at the relevant priority;
 - 5) append a reference to Q_N(UR) to the DLE's unscheduled-service queue, Q(US);
- c) If the sequence is centrally scheduled and centrally executed (see 10.4.3.1b)), then the DLE shall form and send a cancel-schedule SPDU as specified in b)2) to b)5).

10.4.3.3 Receipt of a schedule-cancelled SPDU

If the DLE receives a schedule-cancelled SPDU in response to a cancel-schedule SPDU, then the DLE shall

- a) issue a DL-CANCEL-SCHEDULE confirm with the identifier of the cancelled schedule;
- b) release the schedule identifier and the associated DLSEP-address, if any, making it (them) available for subsequent reuse.

10.4.3.4 LAS-initiated schedule cancellation

If the DLE receives a schedule-cancelled SPDU from the LAS, then the DLE shall determine the class of scheduled sequence being cancelled:

- a) If the sequence is centrally scheduled but locally executed (see 10.4.3.1c)), then the DLE shall
 - 1) remove the reference to the schedule from the DLSEP-address associated with the schedule;
 - 2) issue a DL-CANCEL-SCHEDULE confirm with the identifier of the cancelled schedule;
 - 3) release the schedule identifier and the associated DLSEP-address, making them available for subsequent reuse.
- b) If the sequence is centrally scheduled and centrally executed (see 10.4.3.1b)), then the DLE shall
 - 1) issue a DL-CANCEL-SCHEDULE confirm with the identifier of the cancelled schedule;
 - 2) release the schedule identifier, making it available for subsequent reuse.

10.4.4 Operation of the sequence subsetting service

The primitives of the sequence subsetting service are DL-SUBSET-SEQUENCE request and confirm.

10.4.4.1 Receipt of a DL-SUBSET-SEQUENCE request primitive

When the DLE receives a DL-SUBSET-SEQUENCE request sequence,

- a) if the specified sequence is not defined or not subsettable, then the DLE shall return an appropriate error status and reject the request;
- b) otherwise, when a) does not apply, then the DLE shall subset the sequence as requested and issue a DL-SUBSET-SEQUENCE confirm primitive with the request identifier provided by the DL-SUBSET-SEQUENCE request.

10.4.5 Implicit scheduling of DLS-user requests

When the DLE receives a DLS-user request or response primitive, and the applicable DL-scheduling-policy (see 6.2.6) is IMPLICIT, then the DLE shall execute that request or response as soon as possible, subject to the implicit, or explicit, priority of the request or response. If the request or response requires transmission of one or more DLPDUs, then the DLE shall

- a) select the appropriate queue, $Q_A(UR)$, to which the request should be appended:
 - 1) for connectionless-mode procedures (see 10.3), the associated sending DLSAP-address;
 - 2) for connection-oriented procedures (see 10.2),
 - i) the associated local DLCEP-address for DT DLPDUs when the sending DLCEP's DLCEP-class is PEER or PUBLISHER, or
 - ii) the associated sending (calling or responding) DLSAP-address for EC, DC and RC DLPDUs, or
 - iii) either i) or ii) for DT DLPDUs when the sending DLCEP's DLCEP-class is SUBSCRIBER;
 - 3) for other procedures (see 10.4), the queue designated by the procedure;
- b) append the request, at the appropriate priority, to that $Q_A(UR)$, and adjust that queue, if necessary, so as not to increase the total number of queue members which are in the third partition of the queue, if any;
- c) append a reference to that $Q_A(UR)$, at the appropriate priority, to the DLE's scheduled-service queue, $Q(US)$.

11 Type 1: DL-support subprotocol

11.1 Scope

This subclause

- a) defines the encodings for the higher-level (see 6.1.3) support PDUs (SPDUs) required to support the LAS operation and scheduling subprotocol;
- b) defines the components of a defined sequence (see IEC 61158-3:2003⁹, 10.5.3.2.2;
- c) defines the encodings for the higher-level (see 6.1.3) support PDUs (SPDUs) required to support various configuration and capability queries used by
 - the DL-SUBSCRIBER-QUERY request and DL-LISTENER-QUERY request elements of procedure;
 - bridges for forwarding-database maintenance;
 - specialized DLL monitoring, analysis and management tools;
- d) specifies elements of procedure for processing the received SPDUs.

11.2 Overview of LAS operation

An LAS is required to perform the following functions:

- a) Link maintenance — the LAS
 - 1) detects the presence of a newly added DLE on the local link, after which it activates that DLE;
 - 2) detects that a previously participating DLE is no longer active on the local link;
 - 3) maintains a list of all DLEs operational on the local link;
 - 4) distributes the list of operational DLEs to all other link-master DLEs on the local link.
- b) Time distribution — the LAS acts as the time master for the local link and sends time distribution DLPDUs to all other DLEs on the local link to provide a common sense of DL-time for the local link.
- c) Token circulation — the LAS sends a PT DLPDU to each active DLE on the local link. By doing this repeatedly, the LAS DLE provides a delegated token which “circulates” successively, usually in NODE DL-address order, to all active DLEs on the local link.
- d) Schedule execution — the LAS sends DLPDUs on the local link as dictated by a schedule which specifies the initial DL-time and repetition period for each of those transmissions. An initial schedule (possibly null) is provided by DL-management, and may be subsequently modified by the active LAS.
- e) LAS transfer — the LAS transfers its role to another link master DLE, if requested by that link master or by DL-management.

An LAS may also perform the following functions:

- f) Schedule construction — the LAS may process scheduling requests from other DLEs on the local link and modify the existing schedule, if any, to satisfy those requests. The schedule construction is required to take into account the maximum-scheduled-traffic, V(MST), (see 6.7.5.5) and maximum-scheduling-overhead, V(MSO), (see 6.7.5.6).
- g) Schedule transfer — the LAS may send the current state of the link’s schedule to another DLE on the local link, either as a backup measure or preceding an intentional transfer of the LAS role. This subclause specifies the SPDUs required to transfer the pre-constructed schedule from the LAS DLE to another LM DLE on the local link. The format of these SPDUs does not support the updating of such a transferred schedule. The procedures for schedule transfer are specified in clause 11.

⁹ Expected date of concurrent publication.

The procedures for LAS operation and DL-support operation are specified in 12.2; procedures for schedule construction requests are specified in ; procedures for time distribution are specified in 10.4.1; procedures for token circulation are specified in 8.15; procedures for local schedule construction requests are specified in . The SPDUs for transferring the schedule construction requests to the LAS DLE are specified in this subclause. The procedures for schedule construction depend upon the implementation and thus are not specified.

This subclause also specifies SPDUs to support bridges, protocol-analyzers and other diagnostic tools, (see 11.3.2.7, 11.3.2.8, 11.3.2.9, 11.3.2.10, 11.3.6.4 and 11.3.6.5). The procedures for sending and receiving these SPDUs depend upon the implementation and are thus not specified.

11.3 DL-support subprotocol definition

The DL-support subprotocol defines Support Protocol Data Unit (SPDU) encodings to support the needs of LAS operation, including scheduling and other DLE functions. Other DLE support functions presently defined include mechanisms to share address and limited configuration information in support of

- DL-SUBSCRIBER-QUERY (see 10.2.3) and DL-LISTENER-QUERY (see 10.3.4) requests;
- filtering-database-maintenance within bridges (see clause 11));
- network analysis (link analyzer) support tools.

Any DLPDU sent to, or by, the DL-support functions within a DLE, including any DLPDU addressed to a NODE DL-address, which has a non-null user data field, shall contain as “user data” a single SPDU whose encoding and interpretation is as described in this subclause. This requirement includes any DLPDU addressed to a DLSAP-address designating LAS functionality, such as link-local DL-address 0400₁₆. It also includes any PR or TL DLPDU, both of which always have a user data field.

11.3.1 Common definitions

11.3.1.1 SPDU header

The first octet of each SPDU shall specify a header common to all SPDUs, as depicted in Table 80. It shall contain:

- a) an SPDU classification;
- b) for some SPDUs, the protocol version, whose value is specified in clause 9.

Table 80 – SPDU 1st octet: SPDU class, and protocol version or subclass

SPDU class					Protocol version or subclass		
CCCCC ≤ 11011					VVV		
CCCCC ≥ 11100					SSS		
7	6	5	4	3	2	1	0

When the SPDU class is in the range 00 to 1B₁₆, then the protocol version, VVV, whose value is specified at the beginning of clause 9, shall be present and shall be encoded as in 9.1a)4).

NOTE 1 The resulting range of values for the octet is 00 to DF₁₆.

When the SPDU class is in the range 1C₁₆ to 1F₁₆, then the subclass field, SSS, shall be present.

NOTE 2 The resulting range of values for the octet is E0₁₆ to FF₁₆.

Some of the SPDUs contain only the header octet. The SPDUs fall into five classes:

- a) SPDUs supporting normal link initialization, DLE activation, and other link maintenance functions,
- b) SPDUs supporting LAS transfer,
- c) SPDUs supporting schedule construction,
- d) SPDUs supporting schedule transfer,
- e) SPDUs supporting non-LAS features of the DL-protocol, such as DL-SUBSCRIBER-QUERY and DL-LISTENER-QUERY requests, bridges, remote DL-management, and special-purpose DL-protocol analyzers.

11.3.1.2 Booleans

All Boolean elements share a common encoding:

- 0 FALSE;
- 1 TRUE.

11.3.2 Link-maintenance SPDUs

11.3.2.1 Probe-response SPPDU

The probe-response SPPDU is sent in a Probe Response (PR) DLPDU (see 8.14.1). The probe-response SPPDU shall be formatted as specified in Table 81. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 81 – Probe-response SPPDU

Octet index	Contents of field							
1	0000 0VVV							
2	DL-protocol versions supported							
3	N	F	CC	V(MICD)				
4								
5	V(MRD) × V(ST)							
6								
7	V(MID)							
8	V(RID)							
	7	6	5	4	3	2	1	0

- a) The SPPDU header octet, octet 1, shall be encoded as 0000 0VVV, which identifies this as a DL-probe-response SPPDU.

Table 82 – DL-protocol versions supported

DL-protocol versions supported							
V7	V6	V5	V4	V3	V2	V1	V0
7	6	5	4	3	2	1	0

- b) Octet 2 shall specify the version(s) of the DL-protocol supported, as depicted in Table 82, where each V_K shall be encoded as a Boolean indicating whether protocol version K is supported.

Table 83 – PR-SPDU: 3rd and 4th octets

Token circulation not needed	Fractional duty cycle operation	Expected DLE functional class		Maximum-inactivity-to-claim-LAS-delay (most significant bits)			
N	F	CC		V(MICD) / 256			
15	14	13	12	11	10	9	8

Maximum-inactivity-to-claim-LAS-delay (least significant bits)							
V(MICD) modulo 256							
7	6	5	4	3	2	1	0

c) Octets 3 and 4 shall specify, as depicted in Table 83:

- 1) the DLE's lack of need for token circulation without an explicit request, encoded as a Boolean, N: 0 (no, token circulation is needed) or 1 (yes, token circulation is not needed);
- 2) whether the DLE will function as an FDC DLE which can be expected to be non-responsive to some live-list link-maintenance queries, and whether that DLE should be included in the expected-non-response list, V(ENRL) (see 6.7.5.4), encoded as a Boolean, F;
- 3) the expected functional class of DLE operation: Basic, Link Master or Bridge (see 6.6), to enable the current LAS to determine whether it should query the new DLE's capabilities further and potentially initiate a transfer of the LAS role to the new DLE, encoded as:

00 not reported;

NOTE This alternative exists solely for backward compatability with the earlier editions of IEC 61158-4.

01 Basic;

10 Link Master;

11 Bridge;

NOTE The latter three values are the same coding as specified in 11.3.2.8f)1).

- 4) if the responding DLE is a link master then that DLE's required maximum-inactivity-to-claim-LAS-delay, V(MICD) defined in 6.7.5.19, else zero.

NOTE If this value is larger than the link's configured value of V(MICD), then the responding DLE is not capable of operating as LAS on the link as specified in clause 11.

- d) Octets 5 and 6 shall specify the responding DLE's required value of maximum-response-delay-in-octets, $V(MRD) \times V(ST)$, defined in 6.7.1.3 and 6.7.1.1. If this value, which reflects the device's capability, is larger than the link's configured value of maximum-response-delay-in-octets, $V(MRD) \times V(ST)$, then the responding node is not capable of operating on the local link.
- e) Octet 7 shall specify the responding DLE's required value of minimum-inter-DLPDU-delay, V(MID) defined in 6.7.1.12. If this value, which reflects the device's capability, is larger than the link's configured value of V(MID), then the responding DLE is not capable of operating on the local link.
- f) Octet 8 shall specify the value for the sending DLE's random identifier variable, V(RID) (see 6.7.1.18), which is re-randomized upon receipt of a PN DLPDU addressed to the DLE.

11.3.2.2 Node-activation SPDU

The node-activation SPDU specifies the DL-configuration parameters for the receiving DLE's link. Its primary purpose is to permit the early initiation of DL-services and thus permit higher-layer protocol stacks, including any OSI management protocol stack, to become fully operational.

The node-activation SPDU shall be formatted as specified in Table 84. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 84 – Node-activation SPDU

Octet index	Contents of field							
1	0000 1VVV							
2	value for V(MRC)				value for V(NRC)			
3	value for V(PhLO)							
4	reserved				value for V(TSC)			
5	value for V(MEP)							
6								
7	value for V(TL)							
8								
9	value for V(NDL)							
10								
11	V(RID)							
	7	6	5	4	3	2	1	0

- The SPDU header octet, octet 1, shall be encoded as 0000 1VVV, which identifies this as a node-activation SPDU.
- Octet 2 shall specify in two quartets the value for the receiving DLE's maximum-retry-count variable, V(MRC), as specified in 6.7.1.5, and the value for the receiving DLE's network-retry-count variable, V(NRC), as specified in 6.7.1.6. Both values are properties of the local link.
- Octet 3 shall specify the value for the per-DLPDU-PhL-overhead, as specified in 6.7.1.2. When the value for this variable is not configured, a value corresponding to the LAS DLE's own associated PhE, as then configured, shall be used.

Table 85 – Node-activation SPDU: 4th octet

reserved					V(TSC)		
7	6	5	4	3	2	1	0

- Octet 4 shall specify, as depicted in Table 85:
 - reserved for future use, encoded as 00000;
 - value to be used by the receiving DLE as the local link's time-synchronization-class, V(TSC), as specified in 6.7.1.25, using the coding specified in 9.6 b)1).
- Octets 5 and 6 shall specify the value for the receiving DLE's DL-MAC-address-embedding-prefix variable, V(MEP), as specified in 6.7.1.10.
- Octets 7 and 8 shall specify the value for the receiving DLE's this-link variable, V(TL), as specified in 6.7.1.9. When the appropriate value for this variable is unknown, the LAS DLE shall use a value of zero.
- Octets 9 and 10 shall specify the value for the receiving DLE's network-DLPDU-lifetime variable, V(NDL), as specified in 6.7.1.7.

- h) Octet 11 shall specify the value of the addressed DLE's random identifier variable, V(RID) (see 6.7.1.18), as recorded by the LAS from a prior-received probe-response SPDU.

11.3.2.3 LAS-data-base-status SPDU

The LAS-data-base-status SPDU is sent by the LAS to all LM DLEs on the local link to publish the latest version number for the active schedule and the latest revision number for the local-link live-list, V(LL) (see 6.7.5.2). It is also sent in each TL DLPDU. The LAS-data-base-status SPDU shall be encoded as specified in Table 86. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 86 – LAS-data-base-status SPDU

Octet index	Contents of field
1	1111 0000
2	schedule-type
3 4	schedule version-number
5 6	live-list revision-number

- a) The SPDU header octet, octet 1, shall be encoded as 1111 0000, specifying an LAS-data-base-status SPDU.

Table 87 – LAS-data-base-status SPDU: 2nd octet

Schedule-type							
T	reserved					D	S
7	6	5	4	3	2	1	0

- b) Octet 2 shall specify, as depicted in Table 87:
- 1) the LAS's capability to transfer its schedule, T, encoded as a Boolean: 0 (no, LAS is not capable) or 1 (yes, LAS is capable of transferring its schedule to other LM DLEs);
 - 2) reserved for future use, encoded as 00000;
 - 3) whether all or part of the active schedule has been dynamically constructed by the LAS, D, encoded as a Boolean;
 - 4) whether all or part of the active schedule has been statically constructed by DL-management, S, encoded as a Boolean;
- NOTE If the LAS has an active schedule, the two subfields D and S cannot both be encoded as 0.
- c) Octets 3 and 4 shall specify the version-number of the currently active schedule. If there is no active schedule, then its value shall be zero; otherwise, it shall have a non-zero value.
- d) Octets 5 and 6 shall specify the current live-list revision-number. It is always a non-zero value, which is incremented (modulo 2^{16}) every time the LAS detects a change in the live-list.

11.3.2.4 Live-list-change SPDU

The Live-list-change SPDU is sent by the LAS to all LM DLEs on the local link, whenever the LAS has detected a change in the status of one or more DLEs. This SPDU shall be encoded as specified in Table 88. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field.

Table 88 – Live-list- change SPDU

Octet index	Contents of field
1	1111 0001
2 3	live-list revision-number
4 5	DLE-status
...	DLE-status

- a) The SPDU header octet, octet 1, shall be encoded as 1111 0001, specifying a Live-list-change SPDU.
- b) Octets 2 and 3 shall specify the live-list revision-number. It is incremented every time the LAS detects a change in the live-list and prepares the live-list-SPDU, but it is not incremented if the change is added to an SPDU which has not yet been published by the LAS. The value in this SPDU is the revision-number which applies to the live-list immediately after the changes specified in this SPDU. When this number reaches its maximum value, the next time it is incremented, it is incremented to 1 (thus this number is always non-zero).
- c) The remainder of the SPDU is an array of two octet members specifying DLE-status and structured as shown in Table 89:
 - 1) The first octet of each member shall specify the NODE DL-address of the DLE which is affected by the change.
 - 2) The second octet of each member shall specify the status of the DLE, encoded as.
 - i) N, the DLE's non-need for token circulation, as returned by that DLE in a previously sent PR SPDU (see 11.3.2.1c)1));
 - ii) F, whether the DLE is an FDC DLE, as returned by that DLE in a previously sent PR SPDU (see 11.3.2.1c)2));
 - iii) reserved for future use, encoded as 0000;
 - iv) SS, the last-observed status of that DLE, encoded as
 - 01: not present,
 - 10: present but presumed now asleep, which is only possible for FDC DLEs, or
 - 11: present and awake.

Table 89 – DLE-status structure

Octet sub-index	Contents of subfield							
1	NODE DL-address of the described DLE							
2	N	F	reserved				SS	
	7	6	5	4	3	2	1	0

The length of this SPDU may approach 128 octets.

11.3.2.5 Live-list-request SPDU

This SPDU requests that the current LAS DLE publish the entire live-list. The Live-list-request SPDU shall be encoded with the format specified in Table 80, as follows:

- a) The SPDU header octet, octet 1, shall be encoded as 1111 0010, specifying a Live-list-request SPDU.

11.3.2.6 Live-list-detail SPDU

The Live-list-detail SPDU is sent by the LAS to all LM DLEs on the local link. This SPDU shall be encoded as specified in Table 90. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field

Table 90 – Live-list-detail SPDU

Octet index	Contents of field
1	1111 0011
2	NODE DL-address of LAS's LM DLE
3 4	live-list revision-number
5	V(FUN)
6	V(NUN)
7 ...	Liveness-status
... ...	DLE-type

- a) The SPDU header octet, octet 1, shall be encoded as 1111 0011, specifying a Live-list-detail SPDU.
- b) Octet 2 shall specify the NODE DL-address of the link-master DLE which is currently providing the LAS functionality.
- c) Octets 3 and 4 shall specify the current live-list revision-number.
- d) Octet 5 shall specify the current value of first-unpolled-node, V(FUN) (see 6.7.5.15).
- e) Octet 6 shall specify the current value of number-of-consecutive-unpolled-nodes, V(NUN), (see 6.7.5.16).
- f) Octet 7 and following shall specify the liveness-status of up to 240 DLEs, one bit per DLE, in order of descending NODE DL-address, encoded as an array of Booleans: 0 (DLE is not communicating and presumed “dead”), 1 (DLE is communicating and thus “alive”).

The DLEs not expected to be connected to the link, as specified by the variables V(FUN) and V(NUN), are not included in this status. The array of Booleans shall be ordered so that bit 7 of octet 7 specifies the status of any DLE associated with the highest address (FF₁₆), bit 6 shall specify the status of any DLE associated with the next highest address (FE₁₆), and so on. The unused bits of the last octet shall be set to 0.

- g) This field specifies the FDC status for up to 240 DLEs, one bit per DLE, in order of descending NODE address, encoded as an array of Booleans: 0 (DLE is not an FDC DLE) or 1 (DLE is an FDC DLE). The DLEs not expected to be connected to the link, as specified by the variables V(FUN) and V(NUN), are not included in this status. The array of Booleans shall be ordered so that bit 7 of first octet of this field specifies the status of any DLE associated with the highest address (FF₁₆), bit 6 shall specify the status of any DLE associated with the next highest address (FE₁₆), and so on. The DLE-type for any DLE with a liveness-status of 0, as indicated by the corresponding Boolean encoded in f), shall be encoded as 0.

Trailing octets with the value zero shall be omitted from this field and from the SPDU. Therefore, when the local link has no FDC DLEs, then this entire field shall be omitted from the SPDU.

11.3.2.7 DL-conformance-query SPDU

The DL-conformance-query SPDU requests the receiving DLE to send a DL-conformance SPDU specifying the DL-conformance classes of the replying DLE to the requesting DLE. Its purposes are

- 1) to permit a new LAS DLE's DL-management to assess the impact of DLEs which are already-connected to the local link, facilitating recovery of the LAS role after failure of the previous LAS DLE, and to facilitate potential transfer of the LAS role to a more-appropriate DLE;
- 2) to permit remote DL-management to interrogate DLEs on the extended link, by providing a query (and reply) which can be forwarded through a bridge;
- 3) to facilitate DL-protocol analyzer initialization after attachment to an operating local link.

The DL-conformance-query SPDU shall be encoded with the format specified in Table 80, as follows:

- a) The SPDU header octet, octet 1, shall be encoded as 1110 1100, specifying a DL-conformance-query SPDU.

11.3.2.8 DL-conformance-reply SPDU

The DL-conformance-reply SPDU shall be formatted as specified in Table 91. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field.

Table 91 – DL-conformance-reply SPDU

Octet index	Contents of field
1	0010 0VVV
2	DL-protocol versions supported
3 ... 6	DL-conformance encoding

- a) The SPDU header octet, octet 1, shall be encoded as 0010 0VVV, which identifies this as a DL-conformance-reply SPDU.

Table 92 – DL-protocol versions supported

DL-protocol versions supported							
V ₇	V ₆	V ₅	V ₄	V ₃	V ₂	V ₁	V ₀
7	6	5	4	3	2	1	0

- b) Octet 2 shall specify the version(s) of the DL-protocol supported, as depicted in Table 92, where each V_K shall be encoded as a Boolean indicating whether protocol version K is supported.

Table 93 – DL-conformance encoding (portion 1)

Supported connectionless extended data delivery features		Supported DL-priorities	Timeliness time-stamp support	Supported types of timeliness			
remote-DLE-confirmed unitdata transfer	unitdata exchange	multi-priority	DL-time-stamped DLSDU support	TRANSPARENT	RESIDENCE	UPDATE	SYNCHRONIZED
C	X	M	D	T	R	U	S
7	6	5	4	3	2	1	0

c) Octet 3 shall specify, as depicted in Table 93.

- 1) the DLE's support for connectionless-mode extended data-delivery features, encoded as a set of two Booleans;
- 2) the DLE's support for multiple DL-priorities, encoded as a Boolean;
- 3) the DLE's support for buffer time-stamps on DLCs, encoded as a Boolean;
- 4) the DLE's supported types of DLC timeliness computation, encoded as a set of four Booleans.

Table 94 – DL-conformance encoding (portion 2)

Subscribers-to-publisher support	Supported DLCEP data-delivery features		Supported maximum DLSDU : DLPDU data ratio				
X	FF		RRRRR				
7	6	5	4	3	2	1	0

d) Octet 4 shall specify, as depicted in Table 94.

- 1) the DLE's support for the subscribers-to-publisher direction of multi-peer DLSDU transfer, X, as specified in clause 11 and 13.9.2.3, which shall be encoded as a Boolean;
- 2) the DLE's supported DLCEP data-delivery features, FF, as specified in clause 11 and 13.9.2.2, which shall be encoded as one less than the relevant item number of 13.9.2.2, with a value of 0 to 3;
- 3) the DLE's maximum supported DLSDU : DLPDU data ratio for ORDERED DLCEPs, RRRRR, as specified in clause 11 and 13.9.2.4, which shall be encoded as the relevant supported value from 13.9.2.4, with a value of 0 to 16.

Table 95 – DL-conformance encoding (portion 3)

reserved				Supported time-based scheduling services	Supported DL-time synchronism class		
000				S	TTT		
7	6	5	4	3	2	1	0

e) Octet 5 shall specify, as depicted in Table 95.

- 1) a four-bit field, coded as zero, reserved for future standards use;
- 2) the DLE's support for time-based scheduling services, S, as specified in clause 11 and 13.9.3.2, which shall be encoded as a Boolean;

- 3) the DLE's DL-time synchronism class, TTT, as specified in 12.6.3 and 13.9.3.1, which shall be encoded as specified in b)1).

All other values for these fields are invalid and shall not be used.

Table 96 – DL-conformance encoding (portion 4)

Supported DLE class		Support for LONG DL-addresses	Fractional duty cycle operation	Token circulation desired	Supported LAS time-based scheduling activity class		
CC		A	F	D	SSS		
7	6	5	4	3	2	1	0

- f) Octet 6 shall specify, as depicted in Table 96.
 - 1) the DLE's current class of operation, CC, as specified in 6.6 and 13.10.1, which shall be encoded as the relevant item number of 13.10.1, with a value of 1 to 3;
 - 2) the DLE's support for LONG DL-addresses, encoded as a Boolean;
 - 3) whether the DLE is an FDC DLE, F, which can be expected to be non-responsive to some live-list link-maintenance queries, encoded as 0 (not an FDC DLE) or 1 (an FDC DLE);
 - 4) the DLE's need for token circulation without an explicit request, D, encoded as 0 (token circulation desired) or 1 (token circulation not desired);
 - 5) the DLE's support for LAS time-based scheduling activities, SSS, as specified in clause 11 and 13.9.3.3, which shall be encoded as the relevant item number of 13.9.3.3, with a value of 1 to 5;

All other values for these fields are invalid and shall not be used.

11.3.2.9 Link-basic-parameters-request SPDU

The link-basic-parameters-request SPDU requests the link-basic-parameters SPDU from the addressed LAS. The link-basic-parameters-request SPDU shall be encoded as specified in Table 80:

- The SPDU header octet, octet 1, shall be encoded as 1110 1101, specifying a link-basic-parameters-request SPDU.

11.3.2.10 Link-basic-parameters-reply SPDU

The link-basic-parameters-reply SPDU specifies the DL-configuration parameters for basic-class DLEs on the receiving DLE's link. The link-basic-parameters-reply SPDU shall be formatted as specified in Table 97. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field.

Table 97 – Link-basic-parameters-reply SPDU

Octet index	Contents of field
1	0010 1VVV
2 ... 10	same as octets 2...10 of node-activation SPDU
11 12	value for V(ST)
13	value for V(MRD)
14	value for V(MID)

- a) The SPDU header octet, octet 1, shall be encoded as 0010 1VVV, which identifies this as a link-basic-parameters-reply SPDU.
- b) Octets 2 to 10 shall be identical to the octets 2 to 10 of the node-activation SPDU (see 11.3.2.2).
- c) Octets 11 and 12 shall specify the value for the receiving DLE's slot-time variable, V(ST), as specified in 6.7.1.1.
- d) Octet 13 shall specify the value for the receiving DLE's maximum-reply-delay variable, V(MRD), as specified in 6.7.1.3.
- e) Octet 14 shall specify the value for the receiving DLE's minimum-inter-DLPDU-delay variable, V(MID), as specified in 6.7.1.12. When the appropriate value for this variable is unknown, a value of zero shall be used.

11.3.2.11 Link-master-parameters-request SPDU

The link-master-parameters-request SPDU requests the link-parameters SPDU from the addressed LAS. The link-master-parameters-request SPDU shall be encoded as specified in Table 80.

- a) The SPDU header octet, octet 1, shall be encoded as 1110 1110, specifying a link-master-parameters-request SPDU.

11.3.2.12 Link-master-parameters-reply SPDU

The link-master-parameters-reply SPDU specifies the DL-configuration parameters for link-master-class DLEs on the receiving DLE's link. Its primary purpose is to permit link-master DLEs to prepare to assume the role of LAS on the local link. The link-master-parameters-reply SPDU shall be formatted as specified in Table 98. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 98 – Link-master-parameters-reply SPDU

Octet index	Contents of field
1	0011 0VVV
2	value for V(MST)
3 4	value for V(DMDT)
5 6	value for V(DTHT)
7 8	value for V(LTHT)
9 10	value for V(TTRT)
11	value for V(FUN)
12	value for V(NUN)
13 14	value for V(TDP)
15 16	value for V(MICD)
17 18	value for V(LDDP)
19 20	value for V(ML)

- a) The SPDU header octet, octet 1, shall be encoded as 0011 0VVV, which identifies this as a link-master-parameters-reply SPDU.
- b) Octet 2 shall specify the value for the receiving DLE's local link's maximum-scheduled-traffic variable, V(MST), as specified in 6.7.5.5.
- c) Octets 3 and 4 shall specify the value for the receiving DLE's local link's default-minimum-token-delegation-time variable, V(DMDT), as specified in 6.7.5.7.
- d) Octets 5 and 6 shall specify the value for the receiving DLE's local link's default-token-holding-time variable, V(DTHT), as specified in 6.7.5.8.
- e) Octets 7 and 8 shall specify the value for the receiving DLE's local link's link-maintenance-token-holding-time variable, V(LTHT), as specified in 6.7.5.9.
- f) Octets 9 and 10 shall specify the value for the receiving DLE's local link's target-token-rotation-time variable, V(TTRT), as specified in 6.7.5.11.
- g) Octet 11 shall specify the value for the receiving DLE's local link's first-unpolled-node id variable, V(FUN), as specified in 6.7.5.15.
- h) Octet 12 shall specify the value for the receiving DLE's local link's number-of-consecutive-unpolled-nodes ids variable, V(NUN), as specified in 6.7.5.16.
- j) Octets 13 and 14 shall specify the value for the receiving DLE's local link's time-distribution-period variable, V(TDP), as specified in 6.7.1.18. When the appropriate value for this variable is unknown, the minimum required value for the LAS's time-synchronism class (see 12.6.3) shall be used.
- k) Octets 15 and 16 shall specify the value for the receiving DLE's local link's maximum-inactivity-to-claim-LAS-delay variable, V(MICD), as specified in 6.7.5.19.
- l) Octets 17 and 18 shall specify the value for the receiving DLE's local link's LAS-data-base-distribution-period variable, V(LDDP), as specified in 6.7.1.20.
- m) Octets 19 and 20 shall specify the value for a receiving bridge DLE's maximum-link variable, V(ML), as specified in 6.7.6.1. Non-bridge DLEs which are functioning as an LAS DLE may specify a value of zero for this field of the SPDU when they do not maintain the proper value of V(ML) for the local link.

11.3.2.13 Token-hold-time-request SPDU

The token-hold-time-request SPDU requests the token-hold-time-array SPDU from the addressed LAS. The token-hold-time-request SPDU shall be encoded as specified in Table 99:

Table 99 – Token-hold-time-request SPDU

Octet index	Contents of field
1	1110 1111
2	starting NODE DL-address

- a) The SPDU header octet, octet 1, shall be encoded as 1110 1111, specifying a token-hold-time-request SPDU.
- b) Octet 2 shall specify the NODE DL-addresses for which maximum-token-hold-time is requested. The lower quartet of this octet shall always be 0. If maximum-token-hold-time for all DLEs is requested, then the value of this field shall be 00. If the value of this field is $N0_{16}$, where $N = 1$ to F_{16} , then the request is for DLEs with NODE DL-addresses from $N0_{16}$ to NF_{16} .

11.3.2.14 Token-hold-time-array SPDU

The token-hold-time-array SPDU specifies the maximum-token-hold-time-array parameters (see 6.7.5.10) for DLEs on the receiving DLE's link. Its primary purpose is to permit link-master DLEs to prepare to assume the role of LAS on the local link. This SPDU is sent by the LAS either to all link master DLEs to update their data base, or to a specific link master DLE as a response to token-hold-time-request from that link master DLE. The token-hold-time-array SPDU shall be formatted as specified in Table 100. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 100 – Token-hold-time-array SPDU

Octet index	Contents of field
1	0011 1VVV
2	starting NODE DL-address
3 4	maximum-token-hold-time
33 34	maximum-token-hold-time

- a) The SPDU header octet, octet 1, shall be encoded as 0011 1VVV, which identifies this as a token-hold-time-array SPDU.
- b) Octet 2 shall specify the NODE DL-addresses for which maximum-token-hold-time is included in this SPDU. The lower quartet of this octet shall always be 0. If the value of this field is $N0_{16}$, where $N = 1$ to F_{16} , then this SPDU includes the maximum-token-hold-time for DLEs with NODE DL-addresses from $N0_{16}$ to NF_{16} . If the value of this field is 00, then this SPDU does not include the maximum-token-hold-time for any DLE, and it is used to indicate that the LAS has sent all of token-hold-time-array SPDUs, so that the receiving DLE can do error recovery, if it had not received all of such SPDUs.
- c) Octets 3 to 34 shall specify the value of maximum-token-hold-time parameter for DLEs as an array of two octet values in the ascending sequence of NODE DL-addresses starting with $N0_{16}$ and ending with NF_{16} . If any DLE in this array is constructed or configured so that it does not need to receive a token delegated by a PT DLPDU, so that the value of the N subfield in 11.3.2.1c)1) was 1, then the maximum-token-hold-time for that DLE shall be set to zero.

11.3.2.15 FDC-DLE-has-“awakened” SPDU

The FDC-DLE-has-“awakened” SPDU notifies the local LAS DLE that the specified DLE is now online and ready to receive any DLPDUs which the LAS DLE has stored, awaiting the FDC DLE's return to active participation on the local link.

The FDC-DLE-has-“awakened” SPDU shall be encoded with the format specified in Table 80, as follows:

- The SPDU header octet, octet 1, shall be encoded as 1111 1000, specifying an FDC-DLE-has-“awakened” SPDU.

11.3.2.16 FDC-DLE-may-“go-to-sleep”-notification SPDU

The FDC-DLE-may-“go-to-sleep”-notification SPDU notifies the local LAS DLE that the specified DLE is now ready to go temporarily offline, and that the LAS DLE should act as a bridge with respect to any non-WK DLPDUs addressed to the FDC DLE until the FDC DLE next sends an FDC-DLE-has-“awakened” SPDU.

The FDC-DLE-may-“go-to-sleep”-notification SPDU shall be encoded with the format specified in Table 80, as follows:

- The SPDU header octet, octet 1, shall be encoded as 1111 1001, specifying an FDC-DLE-may-“go-to-sleep”-notification SPDU.

11.3.2.17 FDC-DLE-may-“go-to-sleep”-acknowledge SPDU

The FDC-DLE-may-“go-to-sleep”-acknowledge SPDU notifies the requesting DLE that the local LAS DLE is prepared to act as a bridge with respect to any non-WK DLPDUs addressed to the FDC DLE until the FDC DLE next sends an FDC-DLE-has-“awakened” SPDU.

The FDC-DLE-may-“go-to-sleep”-acknowledge SPDU shall be encoded with the format specified in Table 80, as follows:

- The SPDU header octet, octet 1, shall be encoded as 1111 1010, specifying an FDC-DLE-may-“go-to-sleep”-acknowledge SPDU.

11.3.3 LAS-transfer SPDUs

LAS-transfer SPDUs are used to coordinate link schedule information on a local link, and to transfer the LAS role from one DLE to another DLE on the same link.

11.3.3.1 Relinquish-LAS-role-request SPDU

This SPDU requests that the current LAS DLE transfer its role as LAS to the requesting DLE. If the current LAS DLE accepts the request, it shall transfer the LAS role by sending a TL DLPDU addressed to that requesting DLE as specified in 8.20.3. The relinquish-LAS-role-request SPDU shall be encoded with the format specified in Table 80, as follows:

- The SPDU header octet, octet 1, shall be encoded as 1111 0100, specifying a relinquish-LAS-role-request SPDU.

11.3.3.2 Accept-LAS-role-request SPDU

This SPDU requests that the receiving DLE prepare to accept the role as LAS from the requesting LAS DLE. The accept-LAS-role-request SPDU shall be encoded with the format specified in Table 80, as follows:

- The SPDU header octet, octet 1, shall be encoded as 1111 0101, specifying an accept-LAS-role-request SPDU.

11.3.3.3 Accept-LAS-role-reply SPDU

This SPDU is sent in response to an accept-LAS-role-request SPDU (see 11.3.3.2). By sending the response, the sending DLE accepts the request to become the LAS in the near future and agrees to take whatever actions are necessary (such as LAS-database synchronization) to prepare for the transfer of the LAS role. When the sending DLE is ready to accept the LAS role, it shall initiate the transfer by sending a relinquish-LAS-role-request SPDU (see 11.3.3.1) to the LAS DLE. The accept-LAS-role-reply SPDU shall be encoded with the format specified in Table 80, as follows:

- The SPDU header octet, octet 1, shall be encoded as 1111 0110, specifying an accept-LAS-role-reply SPDU.

11.3.4 Schedule-construction SPDUs

Schedule-construction SPDUs convey requests from a non-LAS DLE to the LAS DLE on the same link, and replies from that LAS DLE to the requesting DLE. They can also be used to cancel previously requested scheduling actions.

11.3.4.1 Sequence description encoding

Sequence descriptions are used in schedule construction (see 11.3.4.2) SPDUs to define desired elements of LAS activity. Each sequence description consists of a series of encoded sequence elements, with the order of sequence elements the same as that in the defining DLS-user-specified sequence.

Each encoded sequence element consists of a header, as specified in Table 101, followed by two or more octets of associated information.

Table 101 – Sequence element header encoding

Item type and format						Priority	
TTTTTT						PP	
7	6	5	4	3	2	1	0

The sequence-element header shall encode the type and format, and priority if relevant, of the sequence element. When present, the priority field shall be encoded as specified in 7.2.1.3.

11.3.4.1.1 SHORT DL-address and duration sequence element

Table 102 – SHORT DL-address and duration sequence element

Octet index	Contents of field
1	TTTT TTPP
2 3	SHORT DL-address
4 5	duration

A SHORT DL-address and duration sequence element shall be encoded as specified in Table 102:

- a) The sequence element header octet, octet 1, shall be encoded as TTTT TTPP, in accordance with Table 101, as follows:

01₁₆ — CD-request, SHORT DL-address and associated duration, URGENT priority;

02₁₆ — CD-request, SHORT DL-address and associated duration, NORMAL priority;

03₁₆ — CD-request, SHORT DL-address and associated duration, TIME-AVAILABLE priority;

04₁₆ — ES-request, SHORT DL-address and associated duration;

- b) Octets 2 and 3 of the sequence element shall specify a SHORT DLCEP-address (CD-request) or SHORT DLSEP-address (ES-request).
- c) Octets 4 and 5 shall specify a two-octet maximum duration of reply-token (CD-request) or delegated-token (ES-request) usage, measured in octets of link transmission capacity.

This is the maximum duration of a single instance of usage of the specified token by the addressed DLE, and does not include the link capacity required to send CD or ES DLPDU, immediate-response-recovery-delay slot-times, $V(IRR D) \times V(ST)$ octet-durations and the retry considerations of the LAS DLE in sending the token through transmission of the CD or ES DLPDU.

11.3.4.1.2 LONG DL-address and duration sequence element

Table 103 – LONG DL-address and duration sequence element

Octet index	Contents of field
1	TTTT TTPP
2 3 4 5	LONG DL-address
6 7	duration

A LONG DL-address and duration sequence element shall be encoded as specified in Table 103.

- a) The sequence element header octet, octet 1, shall be encoded as TTTT TTPP, in accordance with Table 101, as follows:

09₁₆ — CD-request, LONG DL-address and associated duration, URGENT priority;

0A₁₆ — CD-request, LONG DL-address and associated duration, NORMAL priority;

0B₁₆ — CD-request, LONG DL-address and associated duration, TIME-AVAILABLE priority;

0C₁₆ — ES-request, LONG DL-address and associated duration;

- b) Octets 2 to 5 of the sequence element shall specify a delocalized LONG DLCEP-address (CD-request) or LONG DLSEP-address (ES-request).
- c) Octets 6 and 7 shall specify a two-octet maximum duration of reply-token (CD-request) or delegated-token (ES-request) usage, measured in octets of link transmission capacity.

This is the maximum duration of a single instance of usage of the specified token by the addressed DLE, and does not include the link capacity required to send CD or ES DLPDU, immediate-response-recovery-delay slot-times, $V(IRR) \times V(ST)$ octet-durations and the retry considerations of the LAS DLE in sending the token through transmission of the CD or ES DLPDU.

11.3.4.1.3 Wakeup request sequence element

Table 104 – Wakeup request sequence element

Octet index	Contents of field
1	TTTT TT00
2	NODE DL-address

A wakeup request sequence element shall be encoded as specified in Table 104:

- a) The sequence element header octet, octet 1, shall be encoded as TTTT TT00, in accordance with Table 101, as follows:

00₁₆ — WK-request, NODE DL-address;

- b) Octet 2 of the sequence element shall specify a NODE DL-address.

NOTE While a Wakeup sequence element cannot be included in a DLS-user-originated schedule request, it can be included in a scheduling request made by a fractional-duty-cycle (FDC) DLE wishing to schedule its own "reawakening".

11.3.4.2 Schedule-request SPDU

Scheduling requests consist of requests for either one-time or periodic (that is, cyclic with a fixed period) execution of a sequence, subject to specified scheduling constraints. These requests differ only in the parameters specifying the required period and permissible jitter of cyclic execution of the sequence, which are not relevant to one-time execution of the sequence.

Table 105 – Schedule-request SPDU

Octet index	Contents of field
1	0110 0VVV
2	Sequence type, schedule type, and schedule priority
3 4	Schedule identifier
5 6	Value for $V_S(\text{TSL})$
7 8 9 10 11 12	Desired starting time
13 14 15 16	Earliest starting time, relative to desired starting time
17 18 19 20	Latest ending time, relative to desired starting time
21 22 23 24	Period of cyclic sequence
25 26 27 28	Maximum permissible jitter
29 ...	Sequence description

Scheduling-request SPDUs shall be formatted as specified in Table 105.

a) The SPDU header octet, octet 1, shall be encoded as 0110 0VVV.

Table 106 – Sequence type, schedule type and priority encoding

Sequence type			Schedule type			Priority	
SSS			TTT			PP	
7	6	5	4	3	2	1	0

b) Octet 2 shall identify the sequence type, the schedule type, and the priority of the scheduling request. It shall be encoded as specified in Table 106.

1) The sequence type SSS shall specify whether the entire sequence is to be executed contiguously or whether discontinuous (interrupted) execution is permitted, and in the latter case whether the sequence as a whole is to be considered ordered or unordered:

— A **contiguous** sequence is one in which the DLS-provider must schedule enough consecutive segments of local-link capacity to execute the entire sequence.

— A **discontiguous** sequence is one in which the DLS-provider may intersperse the execution of the sequence with other link activity.

— An **ordered** sequence is one in which the DLS-provider must execute the sequence elements in the order specified.

— An **unordered** sequence is one in which the DLS-provider may rearrange the order of execution of the elements of the sequence, facilitating DLL-internal optimizations.

NOTE Contiguous sequences are always treated as ordered, since unordered permits no additional optimization.

This subfield shall be encoded in three bits as.

000) unordered discontiguous;

001) ordered discontiguous;

011) (ordered) contiguous.

All other values are reserved for future standards use.

2) The schedule type TTT shall be encoded in three bits as.

001) one-time;

100) periodic.

All other values are reserved for future standards use.

3) The priority PP shall be encoded in two bits as specified in 7.2.1.3.

c) Octets 3 and 4 shall encode a non-zero schedule-identifier provided by the requesting DLE.

NOTE This identifier enables the DLE which requests scheduling to associate a reply with its corresponding scheduling request.

d) Octets 5 and 6 shall encode the current value of the DL-time source link-id variable, $V_S(TSL)$, of the sending DLE.

e) Octets 7 to 12 shall encode the desired starting DL-time for the first instance of execution of the scheduled sequence, in units of 2^{-5} ms, where that time is referenced to the DL-time source link-id specified in d).

f) Octets 13 to 16 shall encode the offset of the earliest acceptable starting DL-time for the first instance of execution of the scheduled sequence, in units of 2^{-5} ms, where that time is referenced to the DL-time specified in octets 7 to 12.

g) Octets 17 to 20 shall encode the offset of the latest acceptable ending DL-time for the first instance of execution of the scheduled sequence, in units of 2^{-5} ms, where that time is referenced to the DL-time specified in octets 7 to 12.

h) In periodic-scheduling SPDUs, octets 21 to 24 shall encode the repetition period for execution of a cyclic scheduled sequence, in units of 2^{-5} ms. In one-time-scheduling SPDUs, octets 21 to 24 shall be encoded as zero.

i) In periodic-scheduling SPDUs, octets 25 to 28 shall encode the maximum permissible cycle-to-nominal jitter, in units of 2^{-5} ms, in the repetitive starting times of a cyclic scheduled sequence. In one-time-scheduling SPDUs, octets 25 to 28 shall be encoded as zero.

j) The remaining octets, through the last octet of the SPDU, shall encode a description of the sequence to be scheduled, as specified in 11.3.4.1.

The length of this SPDU may approach 256 octets.

11.3.4.3 Scheduling-completed SPDU

The LAS DLE shall acknowledge receipt and completion of a schedule-request SPDU by replying with a scheduling-completed SPDU, which shall be encoded as specified in Table 107:

- a) The SPDU header octet, octet 1, shall be encoded as 0110 1VVV, specifying a scheduling-completed SPDU.

Table 107 – Scheduling-completed SPDU

Octet index	Contents of field
1	0110 1VVV
2 3	Schedule identifier
4	Status or reason
5 6	value for $V_S(\text{TSL})$
7 8 9 10 11 12	Scheduled starting time

- b) Octets 2 and 3 shall encode the schedule-identifier provided by the requesting DLE in the schedule-request SPDU to which this SPDU is a reply.

NOTE This identifier enables the DLE which requested the scheduling to associate the reply with its corresponding scheduling request.

- c) Octet 4 shall encode the status of the corresponding request — either success or the reason for failure, as specified in Table 108.

Table 108 – Status and reason codes

Status	Code
SUCCESS	00 ₁₆
SERVICE-UNAVAILABLE — TEMPORARY	02 ₁₆
RESOURCE-UNAVAILABLE — TEMPORARY	03 ₁₆
TIME-BASE MISMATCH — TEMPORARY	04 ₁₆
SERVICE-UNAVAILABLE — PERMANENT	82 ₁₆
RESOURCE-UNAVAILABLE — PERMANENT	83 ₁₆

- d) If octet 4 encodes success (00), then octets 5 and 6 shall encode the current value of the DL-time source link-id variable, $V_S(\text{TSL})$, of the LAS DLE.
- e) If octet 4 encodes success (00), then octets 7 to 12 shall encode the scheduled starting DL-time for execution of the sequence, in units of 2^{-5} ms, where that time is referenced to the DL-time source link-id specified in d).
- f) If octet 4 encodes failure (any value other than 00), then this SPDU shall have only 4 octets and the next two fields shall be omitted.

11.3.4.4 Cancel-schedule SPDU**Table 109 – Cancel-schedule SPDU**

Octet index	Contents of field
1	1111 1100
2 3	Schedule identifier

The DLE which requested a scheduling action of the LAS DLE may also request the cancellation of that previous scheduling action. The SPDU by which this request is made shall be encoded as specified in Table 109.

- a) The SPDU header octet, octet 1, shall be encoded as 1111 1100, specifying a cancel-schedule SPDU.
- b) Octets 2 and 3 shall encode the schedule-identifier provided by the requesting DLE in the schedule-request SPDU which this SPDU is canceling.

NOTE This identifier enables the LAS DLE to associate the cancellation request with the corresponding scheduling request.

11.3.4.5 Schedule-cancelled SPDU**Table 110 – Schedule-cancelled SPDU**

Octet index	Contents of field
1	1111 1101
2 3	Schedule identifier
4	Status or reason

The LAS DLE shall acknowledge cancellation of a scheduled sequence by sending a schedule-cancelled SPDU to the DLE which originally made the schedule request. The schedule-cancelled SPDU shall be encoded as specified in Table 110:

- a) The SPDU header octet, octet 1, shall be encoded as 1111 1101, specifying a schedule-cancelled SPDU.
- b) Octets 2 and 3 shall encode the schedule-identifier provided by the requesting DLE in the schedule-request SPDU which this SPDU is canceling.

NOTE This identifier enables the DLE which requested the scheduling to associate the reply with its corresponding scheduling request.

- c) Octet 4 shall encode the reason for cancellation, as specified in Table 108.

11.3.5 Schedule-transfer SPDUs

Schedule-transfer SPDUs convey a link schedule from an LAS DLE to a non-LAS DLE on the same link. This transfer allows another link master DLE to assume the LAS role and continue the execution of the link schedule. The link schedule transferred via these SPDUs could have been constructed by the current LAS or could have been originally transferred to the current LAS from another previous LAS DLE or by DL-management. The format of these SPDUs does not save the original schedule-request SPDU. Therefore, an LAS DLE which receives such a schedule from another LAS DLE or from DL-management cannot modify this schedule.

The link schedule consists of one or more periodic subschedule SPDUs and one schedule-summary SPDU as shown in Table 111. Each subschedule contains one or more sequences, all of the same execution period. An LAS may have a limit on the number of different subschedules it can execute simultaneously. The schedule can be transferred one SPDU at a time, or all SPDUs can be included in one DLPDU. If the schedule is transferred in one DLPDU, then the SPDUs should be in one contiguous sequence as shown in the Table 111. The format of each SPDU is such that the end of each SPDU can be determined without any ambiguity.

The 3-bit version number, NNN, of the schedule transfer encoding which is used in schedule transfer SPDUs shall be encoded as 000.

Table 111 – Link-schedule

Contents
Schedule-summary SPDU
Subschedule SPDU
Subschedule SPDU

11.3.5.1 Schedule-summary SPDU

The schedule-summary SPDU is a record of attributes which are common to all subschedules. This SPDU shall be encoded as specified in Table 112. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field. If this SPDU is for a null schedule, then it shall consist of only the first four octets, and the remaining octets shown shall be omitted from the SPDU.

- a) The SPDU header octet, octet 1, shall be encoded as 1111 1111, specifying a schedule transfer SPDU.
- b) Octet 2 shall specify the type of schedule transfer SPDU — schedule-summary, and shall be encoded as 0000 0NNN, where NNN is the version number of the schedule transfer encoding.

Table 112 – Schedule-summary SPDU

Octet index	Contents of field
1	1111 1111
2	0000 0NNN
3 4	schedule version-number
5 6	schedule-builder identifier (0100...0FFF)
7	number of subschedules
8	V(MSO)
9 10	storage required for total schedule
11 12	timing resolution required to meet schedule-jitter commitments
13 14	$V(MRD) \times V(ST)$
15 16	V(TSL)
17 18 19 20 21 22	periodic schedule DL-time base (T0)
23 24 25 26	macro-cycle duration
27 28	subschedule-SPDU reference
...	subschedule-SPDU reference
...	subschedule-SPDU reference
...	end-of-schedule-summary

- c) Octets 3 and 4 shall specify the non-zero value of the schedule version-number as a two-octet integer, or shall specify the value zero. If the value of this field is zero, which is used to indicate a null schedule, then this field shall be the last field actually present in the SPDU.
- d) Octets 5 and 6 shall specify the source of the schedule as an identifier with a value in the range of 0100₁₆ ... 0FFF₁₆.
- NOTE This field is only for reference and is not used by the LAS.
- e) Octet 7 shall specify the number of subschedules in the schedule.
- f) Octet 8 shall specify the value of V(MSO), maximum-scheduling-overhead (see 6.7.5.6), which was used in the construction of the schedule. Its unit of measurement and range of values is specified in 6.7.5.6.
- g) Octets 9 and 10 shall specify the total storage space required for the schedule-summary and all of the subschedules in number of octets. This is the sum of number of octets in each SPDU of the schedule.
- h) Octets 11 and 12 shall specify the minimum resolution of the time clock required to execute this schedule and still meet the jitter specifications to which the schedule was constructed. Its units are 2⁻⁵ ms.

- i) Octets 13 and 14 shall specify the value of maximum-response-delay slot-times, $V(\text{MRD}) \times V(\text{ST})$ octet-durations, which was used in the construction of the schedule. The unit of measurement and range of values are specified in 6.7.1.1 and 6.7.1.3.
- j) Octets 15 and 16 shall specify the link-id of the source of time distribution, which was used to specify the DL-time used in this schedule.
- k) Octets 17 to 22 shall specify the starting DL-time (T_0), referenced to the DL-time source link-id specified in j), of all periodic subschedules in this schedule. The start time of all sequences in all subschedules is shown as offset with respect to this periodic schedule DL-time base. Its units are 2^{-5} ms.
- l) Octets 23 to 26 shall specify the duration of macrocycle, which is equal to the least common multiple of periods of all subschedules. Its units are 2^{-5} ms.
- m) Octets 27 and up to the end of the SPDU shall reference the subschedule-SPDUs by its identifier, as shown in Table 113. Each identifier shall be 2 octets long and shall have a non-zero value.

NOTE This could be a null (zero-length) sequence of identifiers.

- n) The end-of-schedule-summary shall be encoded as a multiple of two octets whose values are all zero.

NOTE The schedule builder can fill the end of this SPDU with more than the minimum number of octets in the anticipation of future addition of subschedule-SPDU references.

Table 113 – Subschedule-SPDU reference

Octet index	Contents of subfield
1 2	subschedule identifier

The length of this SPDU may approach 256 octets.

11.3.5.2 Subschedule SPDU

The subschedule SPDU is a record of attributes which are common to one subschedule and specification of all sequences for that subschedule. This SPDU shall be encoded as specified in Table 114. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 114 – Subschedule SPDU

Octet index	Contents of subfield
1	1111 1111
2	0000 1NNN
3 4	schedule version-number
5 6	subschedule identifier
7 8 9 10	subschedule period
11 ...	sequence sub-SPDU
...	sequence sub-SPDU
...	sequence sub-SPDU
...	end-of-subschedule

- a) The SPDU header octet, octet 1, shall be encoded as 1111 1111, specifying a schedule transfer SPDU.
- b) Octet 2 shall specify the type of schedule transfer SPDU - subschedule, and shall be encoded as 0000 1NNN, where NNN is the version number of the schedule transfer encoding.
- c) Octets 3 and 4 shall specify the two octet integer value of the version-number of the schedule which includes this subschedule; all values except zero are permitted. The value of this field shall be identical to the same field of schedule-summary SPDU.
- d) Octets 5 and 6 shall either specify the non-zero value of a subschedule identifier as referenced in the schedule-summary SPDU (see 11.3.5.1d)) or it shall be zero. If the value of this field is zero, then this shall be the last octet of this SPDU, and it is used to indicate that the LAS has sent all of subschedule SPDUs, so that the receiving DLE can do error recovery, if it had not received all of such SPDUs. If the value of this octet is non-zero, then this octet shall be followed by the other fields of this SPDU as specified in e), f) and g).
- e) Octets 7 to 10 shall specify a non-zero execution period of all sequences in the subschedule. Its units are 2^{-5} ms.
- f) Octets 11 and up to the end of the SPDU shall specify one or more sequences as specified in 11.3.5.2.1.
- g) The end-of-subschedule shall be encoded as a multiple of four octets whose values are all zero.

NOTE The schedule builder can fill the end of this SPDU with more than the minimum number of octets in the anticipation of future addition of sequence sub-SPDUs.

The length of this SPDU may approach 256 octets.

11.3.5.2.1 Sequence sub-SPDU

The sequence sub-SPDU is a sub-record of attributes which are common to one sequence and specification of all elements of that sequence. This sub-SPDU shall be encoded as specified in Table 115. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field:

Table 115 – Sequence Sub-SPDU

Octet index	Contents of subfield
1 2 3 4	scheduled-starting-time-offset
5 6	Maximum computed duration of sequence execution
7 ...	element-description
...	element-description
...	element-description
...	end-of-sequence

- a) Octets 1 to 4 shall specify a non-zero starting time of the sequence as offset from periodic schedule DL-time base (T_0). Its units are 2^{-5} ms. The LAS is required to execute the entire sequence continuously starting from the first to the last element. The starting time of the sequence is

$T_0 + N \times (\text{subschedule period}) + \text{scheduled-starting-time-offset}$,
where N is a non-negative integer.

NOTE Whenever there is a change in the DL-time-offset, $V(DLTO)$, the periodic schedule time-base, $(T0)$, is adjusted by the same amount. Therefore, once the execution of a schedule has begun, the start time of each sequence is given by

$$V(LSTO) + C(NT) =$$

$$V_{start}(LSTO) + C_{start}(NT) + N \times (\text{subschedule-period}) + \text{scheduled-starting-time-offset},$$

where the sum $V_{start}(LSTO) + C_{start}(NT)$ is the value of $V(LSTO) + C(NT)$ at the start of the schedule execution.

- b) Octets 5 and 6 shall specify the worst-case duration of executing the sequence. This field is only for reference and LAS does not use this to execute the sequence.
- c) Octet 7 and following shall specify one or more elements as specified in 11.3.5.2.2.
- d) The end-of-sequence shall be encoded as two octets whose values are both zero.

11.3.5.2.2 Element

The element is the lowest level component of the schedule, and it represents a transaction (see 3.4.21). The element shall be encoded as specified in Table 116. Multi-octet values shall be encoded with the most significant octet of the value encoded in the lowest-index octet of the multi-octet field.

Table 116 – Element-description

Octet index	Contents of subfield
1	element-type
2 ...	element-parameter

- a) Octets 1 shall specify the type of transaction and shall be encoded as:
 - 1011 00PP — CD-request, SHORT DL-address, PP = priority (see 7.2.1.3);
 - 1011 10PP — CD-request, LONG DL-address, PP = priority (see 7.2.1.3);
 - 1000 0000 — ES-request, SHORT DL-address, first and last ES of one sequence;
 - 1000 0010 — ES-request, SHORT DL-address, first ES of one sequence;
 - 1000 0110 — ES-request, SHORT DL-address, subsequent, but not last ES of one sequence;
 - 1000 0111 — ES-request, SHORT DL-address, last ES of one sequence;
 - 1000 1000 — ES-request, LONG DL-address, first and last ES of one sequence;
 - 1000 1010 — ES-request, LONG DL-address, first ES of one sequence;
 - 1000 1110 — ES-request, LONG DL-address, subsequent, but not last ES of one sequence;
 - 1000 1111 — ES-request, LONG DL-address, last ES of one sequence;
 - 0000 0000 — WK-request, NODE DL-address;
- b) Octet 2 and up shall specify the parameters for the element. The length and encoding depends upon the element-type.
 - 1) If element type is CD-request, SHORT DL-address, then the element parameter shall have two octets and these two octets, octets 2 and 3, shall specify a SHORT DLCEP-address.
 - 2) If element type is CD-request, LONG DL-address, then the element parameter shall have four octets and these four octets, octets 2 to 5, shall specify a LONG DLCEP-address.
 - 3) If the element type is ES-request, SHORT DL-address, then the element parameter shall have four octets. The first two octets of these four octets, octets 2 and 3, shall specify a SHORT DLSEP-address. The next two octets of these four octets, octets 4 and 5, shall specify duration measured in octets of link transmission capacity, DD-parameter of ES DLPDU.

- 4) If the element type is ES-request, NODE DL-address, then the element parameter shall have six octets. The first four octets of these four octets, octet 2 to 5 shall specify a NODE DLSEP-address. The next two octets of these six octets, octet 6 and 7 shall specify duration measured in octets of link transmission capacity, DD-parameter of ES DLPDU.
- 5) If the element type is WK-request, NODE DL-address, then the element parameter shall have one octet and this octet, octet 2 shall specify a non-zero NODE DL-address.

11.3.5.3 Schedule-summary-request SPDU

The schedule-summary-request SPDU requests the schedule-summary SPDU for the currently active schedule, from the addressed LAS. The schedule-summary-request SPDU shall be encoded as specified in Table 117.

Table 117 – Schedule-summary-request SPDU

Octet index	Contents of field
1	1111 1110
2	0000 0000

- a) The SPDU header octet, octet 1, shall be encoded as 1111 1110, specifying a schedule transfer request SPDU.
- b) Octet 2 shall be encoded as 0000 0000, specifying a schedule-summary-request SPDU.

11.3.5.4 Subschedule-request SPDU

The subschedule-request SPDU requests the subschedule SPDU from the addressed LAS. The subschedule-request SPDU shall be encoded as specified in Table 118:

Table 118 – Subschedule-request SPDU

Octet index	Contents of field
1	1111 1110
2	0000 1000
3 4	schedule version-number
5 6	subschedule identifier

- a) The SPDU header octet, octet 1, shall be encoded as 1111 1110, specifying a schedule transfer request SPDU.
- b) Octet 2 shall be encoded as 0000 1000, specifying a subschedule-request SPDU.
- c) Octets 3 and 4 shall specify the two octet integer value of the version-number of the schedule; all values except zero are permitted.
- d) Octets 5 and 6 shall either specify the non-zero value of a subschedule identifier as referenced in the schedule-summary SPDU (see 11.3.5.1m)) or it shall be zero. If the value of this field is zero, then this is a request to the LAS to send all subschedule SPDUs for the specified schedule, otherwise it is a request to the LAS to send the specified subschedule SPDU.

11.3.6 Non-LAS SPDUs

Non-LAS SPDUs are used to support DL-services other than scheduling, such as DL-SUBSCRIBER-QUERY requests and DL-LISTENER-QUERY requests, to support bridge filtering database maintenance operations, and to support protocol-analyzers. These SPDUs may be sent between any two DLEs on the extended link, and may be broadcast to all DLEs on a local link where appropriate.

11.3.6.1 Parameter lists in SPDUs

Non-LAS SPDUs are used to query or transfer DLS-provider database information from one DLE to another. These SPDUs use a common parameter-list structure to encode the transferred database items, and to specify the beginning, continuation, or ending of such a list.

Each encoded element of parameter-list consists of a header, as specified in Table 119, followed by zero or more octets of associated information.

Table 119 – Parameter-list element-header encoding

List element type				List element subtype			
TTTT				SSSS			
7	6	5	4	3	2	1	0

The list element-header shall encode the type and subtype, if relevant, of the list element. The format and primary function of the list element shall be determined by the list element type, TTTT, whose encodings shall be.

- 0 begin/end-of-list element;
- 1 continuation-of-list element;
- 2 SHORT DL-address element;
- 3 LONG DL-address element;
- 4 DLSAP-address-characteristics element;
- 5 DLCEP-characteristics element.

11.3.6.1.1 Begin/end-of-list element

Table 120 – Begin/end-of-list element

Octet index	Contents of field
1	0000 0000

A begin/end-of-list element shall be encoded as specified in Table 120:

- The list element header octet, octet 1, shall be encoded as 0000 0000, in accordance with Table 119.

11.3.6.1.2 Continuation-of-list element

Table 121 – Continuation-of-list element

Octet index	Contents of field
1	0001 0000
2 3 4 5	Continuation identifier

A continuation-of-list element shall be encoded as specified in Table 121:

- a) The list element header octet, octet 1, shall be encoded as 0001 0000, in accordance with Table 119.
- b) Octets 2 to 5 shall encode a continuation-identifier provided by the DLE whose DLS-provider database is being read. There is no constraint on its value or interpretation.

NOTE This identifier enables the DLE whose DLS-provider database is being read to pass a *handle* to its peer DLE, for later return in another database-accessing request, such that the two DLEs can continue the database access at later time using only the connectionless lower-level DL-protocol.

11.3.6.1.3 SHORT DL-address element

Table 122 – SHORT DL-address list element

Octet index	Contents of field
1	0010 SSSS
2 3	SHORT DL-address

Each SHORT DL-address list element shall be encoded as specified in Table 122:

- a) The list element header octet, octet 1, shall be encoded as 0010 SSSS, in accordance with Table 119, and the sub-type field, SSSS, shall specify the DL-address type of the accompanying DL-address parameter, as follows:

20₁₆ — SHORT group DL-address;

21₁₆ — SHORT DLSAP-address whose DL(SAP)-role is BASIC;

24₁₆ — SHORT DLSAP-address whose DL(SAP)-role is INITIATOR;

25₁₆ — SHORT DLSAP-address whose DL(SAP)-role is UNCONSTRAINED RESPONDER;

26₁₆ — SHORT DLSAP-address whose DL(SAP)-role is CONSTRAINED RESPONDER;

28₁₆ — SHORT DLCEP-address of a peer DLCEP;

29₁₆ — SHORT DLCEP-address of a publisher DLCEP;

2A₁₆ — SHORT DLCEP-address of a subscriber DLCEP's publisher;

2B₁₆ — SHORT DLCEP-address of a subscriber DLCEP (which exists during DLCEP-establishment);

2C₁₆ — SHORT DLSEP-address;

- b) Octets 2 and 3 of the list element shall specify a SHORT DL-address.

11.3.6.1.4 LONG DL-address element**Table 123 – LONG DL-address element**

Octet index	Contents of field
1	0011 SSSS
2 3 4 5	LONG DL-address

Each LONG DL-address list element shall be encoded as specified in Table 123:

- a) The list element header octet, octet 1, shall be encoded as 0011 SSSS, in accordance with Table 119, and the subtype field, SSSS, shall specify the DL-address type of the accompanying DL-address parameter, as follows:

30₁₆ — LONG group DL-address;

31₁₆ — LONG DLSAP-address whose DL(SAP)-role is BASIC;

34₁₆ — LONG DLSAP-address whose DL(SAP)-role is INITIATOR;

35₁₆ — LONG DLSAP-address whose DL(SAP)-role is UNCONSTRAINED RESPONDER;

36₁₆ — LONG DLSAP-address whose DL(SAP)-role is CONSTRAINED RESPONDER;

38₁₆ — LONG DLCEP-address of a peer DLCEP;

39₁₆ — LONG DLCEP-address of a publisher DLCEP;

3A₁₆ — LONG DLCEP-address of a subscriber DLCEP's publisher;

3B₁₆ — LONG DLCEP-address of a subscriber DLCEP (which exists during DLCEP-establishment);

3C₁₆ — LONG DLSEP-address;

- b) Octets 2 to 5 of the list element shall specify a delocalized LONG DL-address.

11.3.6.1.5 DLSAP-address-characteristics element**Table 124 – DLSAP-address-characteristics element**

Octet index	Contents of field
1	0100 0000
2 ... 5	constrained responder's remote-DLSAP-address as a LONG DL-address

Each DLSAP-address-characteristics list element shall be encoded as specified in Table 124:

- a) The list element header octet, octet 1, shall be encoded as 0100 0000, in accordance with Table 119.
- b) Octets 2 to 5 shall encode the remote DLSAP-address parameter of a DLSAP-address whose DL(SAP)-role binding is CONSTRAINED RESPONDER. This encoding shall be in the form of a delocalized LONG DL-address.

11.3.6.1.6 DLCEP-characteristics element

Table 125 – DLCEP-characteristics element

Octet index	Contents of field
1	0101 0000
2 3	basic DLCEP parameters
4 ... 7	sender DLCEP parameters
8 ... 11	receiver DLCEP parameters

Each DLCEP-characteristics list element shall be encoded as specified in Table 125:

- The list element header octet, octet 1, shall be encoded as 0101 0000, in accordance with Table 119.
- Octets 2 and 3 shall encode the basic parameters of the DLCEP, as specified for the first and second octets of the parameters of an EC DLPDU in Table 37 and Table 38 and in 9.1a)2) to a)8), except that reply-request subfield (R) (see 9.1a)1)) shall be coded as 0.
- Octets 4 to 7 shall encode the sender parameters of the DLCEP, as specified for the seventh to tenth octets, respectively, of an EC DLPDU in Table 41 through Table 43 and in 9.1c)1) to c)10).
- Octets 8 and 11 shall encode the receiver parameters of the DLCEP, as specified for the eleventh to fourteenth octets, respectively, of an EC DLPDU in Table 44 through Table 46 and in 9.1d)1) to d)10).

11.3.6.2 Address-query SPDU

Table 126 – Address-query SPDU

Octet index	Contents of field
1	0100 0VVV
2	query modifiers
3 4 5 6	request identifier
7 ...	DL-address parameter-list element

The address-query SPDU requests the receiving DLE to search its forwarding database (of DL-addresses recognized for either local reception or forwarding) for the DL-address specified by the associated single DL-address parameter list element, and to compose and send an address-report SPDU if that address is found. Its primary purpose is to support DL-SUBSCRIBER-QUERY and DL-LISTENER-QUERY requests and link protocol analyzers.

The address-query SPDU shall be formatted as specified in Table 126:

- The SPDU header octet, octet 1, shall be encoded as 0100 0VVV, specifying an address-query SPDU.

- b) Octet 2 specifies whether additional information on the DLSAP-address-binding or DLCEP should be contained within a responder's address-report SPDU. Its encoding shall be:

00₁₆ — No accompanying peer DL-address or DLCEP information is requested.

01₁₆ — Responders should include list elements for peer DL-address, and, where appropriate, for DLCEP-characteristics, in the replying address-report SPDU when known.

Other values are reserved.

- c) Octets 3 to 6 shall encode a non-zero request-identifier provided by the requesting DLE.

NOTE This identifier enables the requesting DLE to associate a reply with its corresponding request.

- d) Octets 7 and following shall encode either one SHORT DL-address element or one LONG DL-address element, as specified in 11.3.6.1.3 and 11.3.6.1.4. The encoded DL-address shall specify the DL-address which is the subject of the query.

11.3.6.3 Address-report SPDU

Table 127 – Address-report SPDU

Octet index	Contents of field
1	0100 1VVV
2	reason for report
3 4 5 6	request identifier
7 ...	parameter-list elements

The address-report SPDU is sent in response to an address-query SPDU when a receiving DLE determines that the queried DL-address is contained within the receiving DLE's forwarding database, for either local reception or for forwarding to another DLE.

It is also sent to all local bridges to report the activation or deactivation of a DL-address

- 1) by a non-FDC DLE when the DL-address is non-local (that is, has a link component with a value other than zero or V(TL));
- 2) by an FDC DLE for all of its addresses.

NOTE DL-address activation occurs upon executing a DL-BIND or DL-CONNECT request; DL-address deactivation occurs upon executing a DL-UNBIND or DL-DISCONNECT request.

Its primary purposes are to support DL-SUBSCRIBER-QUERY and DL-LISTENER-QUERY requests, and to facilitate bridges and link protocol analyzers.

The address-report SPDU may be addressed to a DLE on another link. It shall be formatted as specified in Table 127:

- a) The SPDU header octet, octet 1, shall be encoded as 0100 1VVV, specifying an address-report SPDU.
- b) Octet 2 specifies the reason for the address-report SPDU; it shall be encoded as:

00₁₆ — The SPDU is a reply to a prior address-query SPDU;

10₁₆ — A non-FDC DLE has activated the specified DL-address;

11₁₆ — An FDC DLE has activated the specified DL-address;

12₁₆ — A non-FDC DLE has deactivated the specified DL-address;

13₁₆ — An FDC DLE has deactivated the specified DL-address.

All other values are reserved for future standard use.

c) Octets 3 to 6 shall encode

- 1) the non-zero request-identifier provided by the requesting DLE in the address-query SPDU to which this SPDU is a reply; or

NOTE This identifier enables the querying DLE to associate the reply with its corresponding query.

- 2) a value of zero indicating that this is a spontaneous report of the activation or deactivation of a DL-address, and not a reply to a prior SPDU.

d) Octets 7 and following shall encode either one SHORT DL-address element or one LONG DL-address element, as specified in 11.3.6.1.3 and 11.3.6.1.4.

e) If this SPDU is a reply to an address-query SPDU, and that associated query SPDU specified that responders should include peer DL-address and DLCEP-characteristics elements in the reply SPDU when known.

- 1) the responding DLE shall append a DLSAP-address-characteristics or DLCEP-address-characteristics element specifying the characteristics of the queried DL-address to the reply SPDU, as specified in 11.3.6.1.5 and 11.3.6.1.6; and then
- 2) if the queried DL-address specified a peer DLCEP-address, then the responding DLE shall append a DL-address element specifying the remote peer DL-address of the queried DL-address to the reply DLPDU, as specified in 11.3.6.1.3 and 11.3.6.1.4.

NOTE These last two elements provide necessary functionality to support link protocol analyzers.

11.3.6.4 Address-list-query SPDU

The address-list-query SPDU requests the receiving DLE to search its forwarding database (of DL-addresses recognized for either local reception or forwarding) for DL-addresses which meet the query criteria, starting at the beginning-of-list or continuation-point specified in the query, and to include as many of those DL-addresses which meet the criteria as possible in an address-list-reply SPDU, concluding the SPDU with an end-of-list or continuation-point list element.

Table 128 – Address-list-query SPDU

Octet index	Contents of field
1	0101 0VVV
2 3	selection criteria
4 5 6 7	request identifier
8 ...	parameter-list element

The primary purpose of this SPDU is to provide bridges with a means of building and maintaining their forwarding databases.

The address-list-query SPDU shall be formatted as specified in Table 128:

- a) The SPDU header octet, octet 1, shall be encoded as 0101 0VVV, specifying a address-list-query SPDU.

Table 129 – DL-address selection criteria

Include standard addresses	reserved			DLSEP-address	Subscriber DLCEP	Publisher DLCEP	Peer DLCEP
A	B (=0)	C (=0)	D (=0)	E	F	G	H
15	14	13	12	11	10	9	8

reserved			Constrained responder DLSAP-address	Unconstrained responder DLSAP-address	Initiator DLSAP-address	Basic DLSAP-address	Group DL-address
J (=0)	K (=0)	L (=0)	M	N	P	Q	R
7	6	5	4	3	2	1	0

b) Octets 2 and 3 specify the selection criteria for DL-addresses to be included within a responder's address-list-reply SPDU. They shall be encoded as two octets of individual selection criteria, as specified in Table 129, with octet 2 consisting of fields A to H, and octet 3 of fields J to R.

- 1) If field A=1, then the standard DL-addresses specified in Table 5 through Table 7 which meet the other selection criteria shall be included in the reply list; otherwise (field A=0) they shall be excluded from the reply list.
- 2) Fields B, C, D, J, K and L are reserved for future standards use and shall be encoded as zero.
- 3) If field E=1, then active DLSEP-addresses of the receiving DLE shall be included in the reply list; otherwise (field E=0) they shall be excluded from the reply list.
- 4) If field F=1, then (publisher) DLCEP-addresses of subscriber DLCEPs within the receiving DLE, or forwarded by the receiving DLE, shall be included in the reply list; otherwise (field F=0) they shall be excluded from the reply list.
- 5) If field G=1, then DLCEP-addresses of publisher DLCEPs within the receiving DLE, or forwarded by the receiving DLE, shall be included in the reply list; otherwise (field G=0) they shall be excluded from the reply list.
- 6) If field H=1, then DLCEP-addresses of peer DLCEPs within the receiving DLE, or forwarded by the receiving DLE, shall be included in the reply list; otherwise (field H=0) they shall be excluded from the reply list.
- 7) If field M=1, then DLSAP-addresses of DLSAPs within the receiving DLE, or forwarded by the receiving DLE, whose DL(SAP)-role is **CONSTRAINED-RESPONDER**, shall be included in the reply list; otherwise (field M=0) they shall be excluded from the reply list.
- 8) If field N=1, then DLSAP-addresses of DLSAPs within the receiving DLE, or forwarded by the receiving DLE, whose DL(SAP)-role is **UNCONSTRAINED-RESPONDER**, shall be included in the reply list; otherwise (field N=0) they shall be excluded from the reply list.
- 9) If field P=1, then DLSAP-addresses of DLSAPs within the receiving DLE, or forwarded by the receiving DLE, whose DL(SAP)-role is **INITIATOR**, shall be included in the reply list; otherwise (field P=0) they shall be excluded from the reply list.
- 10) If field Q=1, then DLSAP-addresses of DLSAPs within the receiving DLE, or forwarded by the receiving DLE, whose DL(SAP)-role is **BASIC**, shall be included in the reply list; otherwise (field Q=0) they shall be excluded from the reply list.
- 11) If field R=1, then group DL-addresses of DLSAPs within the receiving DLE, or forwarded by the receiving DLE, shall be included in the reply list; otherwise (field R=0) they shall be excluded from the reply list.

- c) Octets 4 to 7 shall encode a non-zero request-identifier provided by the requesting DLE.

NOTE This identifier enables the requesting DLE to associate a reply with its corresponding request.

- d) Octets 8 and following shall encode either one begin/end-of-list element or one continuation-of-list element, as specified in 11.3.6.1.1 and 11.3.6.1.2. The encoded element shall specify to the receiving DLE where, within its forwarding database, the search should begin:
- 1) If a begin/end-of-list element is specified, it indicates that the query is interrogating the entire forwarding database of the receiving DLE.
 - 2) If a continuation-of-list element is specified, it indicates that the query is interrogating the forwarding database of the receiving DLE continuing from the specified point. For this reason, the continuation-of-list element shall be identical to that returned by a prior address-list-reply SPDU received from the same correspondent DLE.

11.3.6.5 Address-list-reply SPDU

Table 130 – Address-list-reply SPDU

Octet index	Contents of field
1	0101 1VVV
2 3 4 5	request identifier
6 ...	parameter-list elements

The address-list-reply SPDU is sent in response to an address-list-query SPDU. Its primary purpose is to provide bridges with a means of building and maintaining their forwarding databases.

The address-list-reply SPDU may be addressed to a DLE on another link. It shall be formatted as specified in Table 130.

- a) The SPDU header octet, octet 1, shall be encoded as 0101 1VVV, specifying an address-list-reply SPDU.
- b) Octets 2 to 5 shall encode the non-zero request-identifier provided by the requesting DLE in the address-list-query SPDU to which this SPDU is a reply.

NOTE This identifier enables the querying DLE to associate the reply with its corresponding query.

- c) Octets 6 and following shall encode zero or more SHORT DL-address and LONG DL-address elements, as specified in 11.3.6.1.3 and 11.3.6.1.4, specifying DL-addresses which meet the selection criteria specified in the corresponding address-list-query SPDU to which this is a reply.

The last element in the parameter-list shall encode either one begin/end-of-list element or one continuation-of-list element, as specified in 11.3.6.1.1 and 11.3.6.1.2. If the encoded element is a begin/end-of-list element, it shall specify to the querying DLE that the requested search of the replying DLE's forwarding database is complete; if it is a continuation-of-list element, then the querying DLE can request that the search be continued at the point indicated in that continuation-of-list element.

The length of this SPDU may approach 256 octets.

11.4 Elements of Procedures for receiving SPDUs

11.4.1 Procedures for SPDUs received by the DLE acting as LAS

11.4.1.1 Receipt of a Link-basic-parameters-request SPDU

The LAS DLE shall

- a) form a Link-basic-parameters-reply SPDU (11.3.2.10);
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the Link-basic-parameters-request SPDU;
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required Link-basic-parameters-reply SPDU as the conveyed DLSDU.

11.4.1.2 Receipt of an FDC-DLE-has-“awakened” SPDU

The LAS DLE shall forward all those DLPDUs addressed to the notifying DLE which were stored by the LAS DLE while the notifying DLE had “gone-to-sleep”, and the LAS DLE shall stop storing any new received DLPDUs addressed to the notifying DLE.

11.4.1.3 Receipt of a FDC-DLE-may-“go-to-sleep”-notification SPDU

The LAS DLE shall send an FDC-DLE-may-“go-to-sleep”-acknowledge SPDU to the notifying DLE, and shall start storing any new received DLPDUs addressed to the notifying DLE for subsequent forwarding when that notifying DLE next notifies the LAS DLE that it has “awakened”.

11.4.1.4 Receipt of an Accept-LAS-role-reply SPDU

The LAS DLE shall note that the Accept-LAS-role-request SPDU has been received by the replying DLE, and shall wait for the Relinquish-LAS-role-request SPDU from that DLE.

11.4.1.5 Receipt of a Schedule-request SPDU

The LAS DLE shall attempt to update the schedule to satisfy this request. After completing the processing of this request, the LAS DLE shall send the Scheduling-completed SPDU as specified in 11.3.4.3.

11.4.1.6 Receipt of a Cancel-schedule SPDU

The LAS DLE shall attempt to update the schedule to satisfy this request. After completing the processing of this request, the LAS DLE shall send the Schedule-cancelled SPDU as specified in 11.3.4.5.

11.4.2 Procedures for SPDUs received by an LM DLE

11.4.2.1 Receipt of an Accept-LAS-role-request SPDU

If the addressed DLE accepts the request, it shall respond with an accept-LAS-role-reply SPDU, and shall take whatever actions are necessary to prepare for the transfer of the LAS role. When the addressed DLE is ready to accept the LAS role, it shall initiate the transfer by sending a relinquish-LAS-role-request SPDU (11.3.3.1) to the LAS DLE.

11.4.3 Procedures for SPDUs received by all DLEs

11.4.3.1 Receipt of a DL-conformance-query SPDU

The addressed DLE shall schedule the transmission of a connectionless DT DLPDU of NORMAL priority with the required DL-conformance-reply SPDU (11.3.2.8) as the conveyed DLSDU. The destination DL-address of this DLPDU shall be equal to the source DL-address of the DLPDU which conveyed the DL-conformance-query SPDU.

11.4.3.2 Receipt of a FDC-DLE-may-“go-to-sleep”-acknowledge SPDU

If the receiving DLE is an FDC DLE then it may become inattentive to all non-WK DLPDUs received on the local link.

12 Type 1: Other DLE elements of procedure

Throughout this clause, whenever a conditional action is specified and the specified enabling condition does not occur, then the corresponding action also does not occur. The net effect is that events which do not meet any of the enabling conditions specified in a service procedure will have no consequential actions with respect to that specific service procedure.

12.1 DLE initialization

12.1.1 Hardware or host-system initialization

Upon power-up or after being reset, a DLE shall enter the OFFLINE DL-state, in which it is incapable of transmission on the local link, and shall issue a DLM-EVENT indication notifying DL-management of the event. The DLE shall remain in the OFFLINE state until instructed by DL-management to commence normal DL-operation.

DL-reception in the OFFLINE state is permitted but is not required.

The PhE reports its characteristics to the DLL during its own initialization, as specified in 6.4.1.1. If the reported framing-overhead is more than the DLE's configured per-DLPDU-PhL-overhead, $V(\text{PhLO})$, then the DLE shall remain in the OFFLINE state and shall notify local D-management of the problem.

12.1.2 Receipt of a DLM-ACTION request primitive

When the DLE receives a DLM-ACTION request, it shall associate the DLMS-user-specified request identifier with the request. If the request is accepted, as indicated by a returned status of “request accepted” for the DLM-ACTION request, then upon completion of the request, either successfully or after failure, the DLE shall issue a DLM-ACTION confirm with the same request identifier, conveying the status of the request to the DLMS-user.

12.1.2.1 Receipt of a DLM-ACTION request primitive requesting a change in DLE state

Figure 15 shows the state transitions between a DLE's offline and online states.

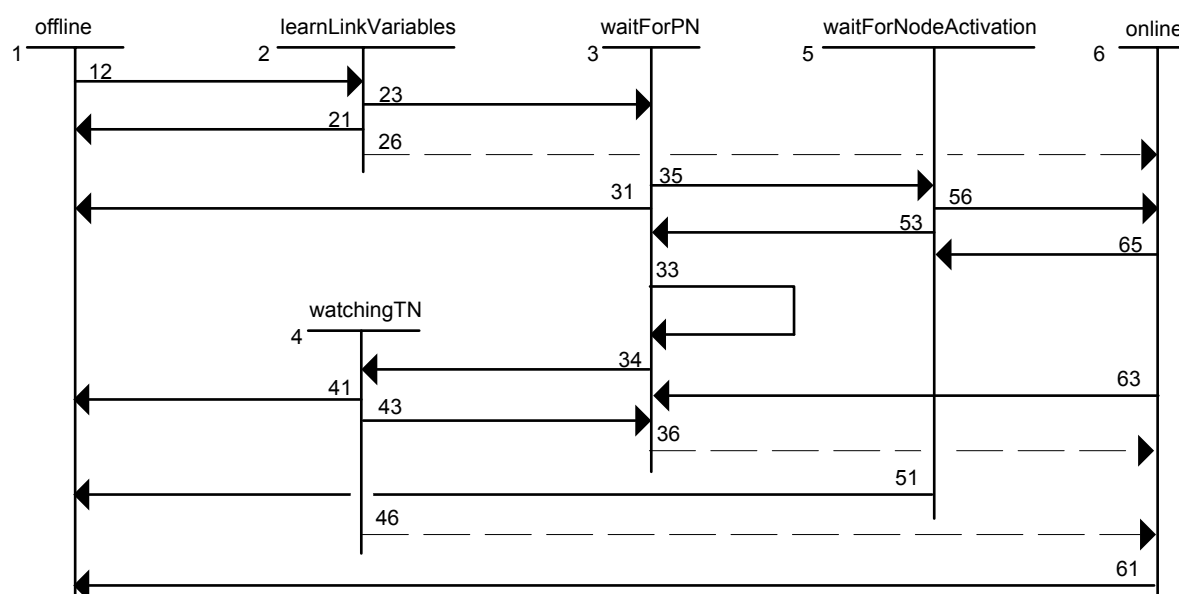


Figure 15 – State transitions of a DLE

If the DLM-ACTION request specifies that the DLE should change its DL-state to ONLINE, then the DLE shall initiate this process by changing its state to learnLinkVariables.

- a) If the DLE's currently-assigned NODE DL-address is in the range $F8_{16}..FF_{16}$ (see 6.3.2.2), then the DLE shall randomly choose, with an approximately uniform distribution, the low-order two bits of that NODE DL-address, adopt that modified address as its currently assigned address, and use the resulting address until a new NODE DL-address is assigned to (or by) the DLE. The actual random choice shall be statistically independent of similar choices made by other DLEs.

NOTE This requirement for statistical independence minimizes the probability of repeatedly identical choices by identically constructed real devices, and thus provides the basis for eventual discrimination among multiple DLEs which happen to choose or respond to the same NODE DL-address, either through fault or misconfiguration.

- b) If the DLE is capable of, and enabled for, link-master-class or bridge-class operation, and the DLE has knowledge of

- the slot-time, $V(ST)$;
- the maximum-inactivity-to-claim-LAS-delay, $V(MICD)$;
- the Ph-parameters required for forming the PN DLPDU (see 8.13.2);
- the DL-parameters of the local link enumerated in the node-activation SPDU (see 11.3.2.2],

then the DLE shall enable the link-activity-monitoring and CL DLPDU transmission activities of 8.19.3, and if successful in claiming the scheduler token, shall

- change its DL-state to ONLINE;
- issue a DLM-EVENT indication notifying DL-management of the event.

- c) The DLE shall enable the NODE DL-address monitoring activities of 8.13.4 and 8.15.4 and PR DLPDU transmission activities of 8.14.3:

- 1) The DLE shall learn the DLL protocol version and some of the PhL and DLL configuration parameters of the local link from the PN-parameters of any received PN DLPDU and enter the waitForPN state.
- 2) If the DLE receives a PT DLPDU addressed to its currently-assigned NODE DL-address, then the DLE shall monitor the apparent usage of its currently-assigned NODE DL-address (by another DLE on the local link) as follows:

- i) If the currently-assigned NODE DL-address is in the range $10_{16}..F7_{16}$ (see 6.3.2.2), then the DLE shall enter the watchingTN state, and if the DLE detects that its currently-assigned NODE DL-address is in use by another DLE (by detecting a subsequent immediate-response DLPDU, using the same procedure that the LAS uses to monitor the response to a transmitted PT DLPDU as specified in 8.15.3.2), then the DLE shall first change its NODE DL-address to $F8_{16}$ and then restart this procedure with step a);

- ii) If the currently-assigned NODE DL-address is in the range $F8_{16}..FF_{16}$ (see 6.3.2.2), then the DLE shall restart this procedure with step a);

NOTE Restarting this procedure with step a) does not cause a state change.

- 3) If the DLE receives a PN DLPDU addressed to its currently assigned NODE DL-address, and

- the DLE is not in the waitForNodeActivation state;
- the DLE supports the protocol version specified in that PN DLPDU;
- the DLE and its associated PhE are configured as specified in that PN DLPDU;
- the DLE is able to respond within the maximum-reply-delay, $V(MRD)$, specified by the received PN DLPDU,

NOTE The DLE may have learned the link parameters from an earlier PN DLPDU, but it is necessary to check the parameters of the just received PN DLPDU, and reconfigure if necessary, because the parameters may have changed.

then

- i) the DLE shall respond as in 8.13.4 and 8.14.3;
- ii) if
 - the DLE's required minimum-inter-DLPDU-delay is less than or equal to the value of V(MID) specified in the received PN DLPDU;
 - the DLE's required maximum-response-delay is less than or equal to the value of V(MRD) specified in the received PN DLPDU,

NOTE The above two comparisons are made between the local link's configured value of V(MID) and V(MRD) as received in a PN DLPDU and the DLE's minimum requirements (that is, its device capabilities) for these two parameters. The DLE sends its minimum required values in a PR SPDU. Any DLPDU sent by any DLE, including any PR DLPDU, is required to use the link's configured value of V(MID) for delaying its transmission, as specified in 6)2)5).

then the DLE shall change its state to waitForNodeActivation;

- iii) if ii) does not apply, then the DLE shall restart this procedure with step a).

4) If the DLE is in the waitForNodeActivation state, and

- i) if a PN or PT DLPDU is received then the DLE shall restart this procedure with step a).
- ii) if a DT DLPDU is received, and the DT DLPDU contains a node-activation SPDU (see 11.3.2.2), and the N(RID) field of that SPDU equals the value of V(RID) (see 6.7.1.18) which was used in the probe-response SPDU conveyed in the PR DLPDU last sent by this DLE, then the DLE shall
 - change its configuration variables to reflect the contents of that received node-activation SPDU;
 - change its DL-state to ONLINE;
 - issue a DLM-EVENT indication notifying DL-management of the event.

iii) if

- A) a DT DLPDU with two addresses is received, where the destination address of the DT DLPDU is the NODE DL-address currently assigned to the DLE;
- B) the source address of the DT DLPDU is the default address for the local-link's LAS;
- C) either the DT DLPDU does not contain a node-activation SPDU, or the N(RID) field of the received node-activation SPDU is not equal to the value of V(RID) (see 6.7.1.18) which was used in the probe-response SPDU conveyed in the PR DLPDU last sent by this DLE,

then the DLE shall infer that its currently assigned NODE DL-address is in use by another DLE, at which point it shall first change its NODE DL-address to F8₁₆ and then restart this procedure with step a).

If the DLM-ACTION request specifies that the DLE should change its DL-state to OFFLINE, then the DLE shall

- change its DL-state to OFFLINE;
- issue a DLM-EVENT indication notifying DL-management of the event.

12.1.2.2 Receipt of a DLM-ACTION request primitive requesting a change in link configuration parameters

If a DLE which is acting as the link's LAS receives a DLM-ACTION request which would change any of the link parameters conveyed by a Node-activation SPDU, a Link-basic-parameters-reply SPDU or a Link-master-parameters-reply SPDU, then the LAS DLE shall convey the changed parameter(s) to the other affected DLEs on the local link by sending the appropriate SPDU(s) to those other affected DLEs at the appropriate group DL-address(es).

If a DLE which is not acting as the link's LAS receives a DLM-ACTION request which would change any of the link parameters conveyed by a Node-activation SPDU, a Link-basic-parameters-reply SPDU or a Link-master-parameters-reply SPDU, then that non-LAS DLE shall not change its operational values for those variables, but shall update the values which it would use should it initialize the link after detecting the non-existence of any other DLE acting as LAS.

12.1.2.3 Receipt of a DLM-ACTION request primitive requesting assumption of the LAS role

If an LM DLE which is not acting as the link's LAS receives a request from local DL-management to assume the LAS role, then

- if the DLE does not have the necessary parameters required to operate as the LAS, then it shall follow the procedure of clause 11 to obtain those parameters.

After receiving the necessary parameters to operate as LAS, the LM DLE shall

- a) form a Relinquish-LAS-role-request SPDU;
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400₁₆ (see 6.3.3.2);
 - a SHORT source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions;
 - a null SD-parameter field;
 - a Relinquish-LAS-role-request SPDU as the conveyed DLSDU;
- c) wait for reception of a TL DLPDU;
- d) if a TL DLPDU is not received within 15 receptions of a PT DLPDU with a token-use-subfield equal to RESTART, then the LM DLE shall retry this process up to a maximum of two times commencing with step a);
- e) if a TL DLPDU is received, but is rejected as specified in 8.20.4.2, then the LM DLE shall assume that it does not have the necessary parameters required to operate as the LAS and shall repeat the procedures of clause 11 up to a maximum of two times;
- f) if a TL DLPDU is still not received after all retry attempts (as in d) or e)), then the LM DLE shall inform local DL-management of this failure via the corresponding DLM-ACTION confirm.

12.1.3 Receipt of a PN DLPDU while in the ONLINE state

If the DLE is not in any of the initialization states of clause 11, and the DLE receives a PN DLPDU addressed to its currently assigned NODE DL-address, and the DLE supports the protocol version specified in that PN DLPDU, then

- the DLE and its associated PhE shall be configured as specified in that PN DLPDU;
- if the DLE is able to respond within the maximum-reply-delay, V(MRD), specified by the received PN DLPDU, then

- i) The DLE shall respond as in 8.13.4 and 8.14.3.
- ii) if
 - the DLE's required minimum-inter-DLPDU-delay is less than the value of $V(MID)$ specified in the received PN DLPDU;
 - the DLE's maximum-response-delay is less than the value of $V(MRD)$ specified in the received PN DLPDU,
 the DLE shall change its state to `waitForNodeActivation` and behave as specified in clause 11.

NOTE The DLE should not disconnect any DLCEPs or do any other initialization and continue to operate normally after it changes its state from `waitForNodeActivation` to `ONLINE`.

- iii) If
 - the DLE does not support the protocol version specified in the received PN DLPDU,
 - the DLE is not able to respond within the maximum-reply-delay, $V(MRD)$, specified by the received PN DLPDU, or
 - ii) does not apply,
 then the DLE shall
 - issue a DL-Disconnect indication to all non-idle DLCEPs;
 - initialize the state of all DLCEPs to `Idle`;
 - reset all queues to empty;
 - issue a confirm primitive with an appropriate status for any outstanding DLS-user request;
 - perform any other initialization necessary to transition the DLE to the `OFFLINE` state;
 - start with the procedure of 12.1.2.1b).

12.2 LAS behavior and operation

12.2.1 LAS operation when holding a scheduler token

The LAS DLE shall initiate each scheduled transaction at its scheduled time within the jitter limits permitted by the schedule for that transaction.

The LAS DLE shall repetitively use the following procedure to determine the next (unscheduled) transaction, subject to the constraint that the transaction can be completed before the next scheduled activity.

- a) If the LAS DLE is required to send a TD DLPDU, then it shall send a TD DLPDU.
- b) If the LAS DLE has not completed the current cycle of "circulating the token", as indicated by the fact that
 - the next DLE to receive the PT DLPDU is not the lowest-numbered NODE DL-address represented in $V(LL)$, or
 - the value of the final-use subfield of that next PT DLPDU will be `CONTINUE`,

then

- 1) The LAS DLE may use the available bus capacity for link maintenance as specified in clause 11.
- 2) If the LAS DLE is required to send an ES DLPDU as specified in clause 11, then it may use the available bus capacity for sending that ES DLPDU as specified in 8.16.3.2, provided this use is limited to sending one ES DLPDU to the DL-address specified in the ES DLPDU per cycle of "circulating the token".

- 3) If the available link capacity permits, then the LAS DLE shall send a PT DLPDU as specified in 8.15.3.2;
- 4) Otherwise, when 3) does not apply and the available link capacity is not sufficient to send a PT DLPDU, the LAS DLE shall repetitively use the following criteria to determine the next DLPDU to be transmitted:
 - i) If a node-activation SPDU is scheduled for transmission, and there is sufficient available link capacity to send the SPDU in a DT DLPDU as specified in 8.7, then the LAS DLE shall transmit the SPDU within the envelope of a DT DLPDU addressed the intended recipient.
 - ii) If i) does not apply, and there is sufficient available link capacity to probe a new node address as specified in 8.13, then the LAS DLE shall perform the procedure of 8.13.3.
 - iii) If neither i) nor ii) applies, then the LAS DLE may transmit any other DLPDU
 - from the generic source address for the local LAS, 0400₁₆ (see 6.3.3.2);
 - which is already queued for transmission,
 provided that there is sufficient available link capacity to complete the transaction.
 - iv) If none of i) to iii) applies, and the duration until the next scheduled transaction exceeds token-recovery-delay slot-times, $P(\text{TRD}) \times V(\text{ST})$ octet-durations, then the LAS DLE shall send one or more IDLE DLPDUs of appropriate length to keep the local link from experiencing an excessive period of inactivity.

Any such link capacity which is not adequate for the procedure of 8.15.3 need not be attributed to the duration of token usage allocated for the LAS DLE's link-maintenance activities.
- c) Otherwise, when b) does not apply, so that the LAS DLE has just completed a cycle of "circulating the token", then
 - 1) If the LAS DLE has a need to transfer its role to another DLE, then
 - i) the LAS DLE shall not start the next cycle of "circulating the token"; but instead
 - ii) the LAS DLE shall send all DLPDUs which it has already scheduled for transmission from the generic source address for the local LAS, 0400₁₆ (see 6.3.3.2), after which
 - iii) the LAS DLE shall send a TL DLPDU as specified in 8.20.3.
 - 2) Otherwise, when 1) does not apply, then the LAS DLE shall send all link-maintenance DLPDUs (see 6.7.5.9) which are queued for transmission, if necessary over an interval of one or more cycles of "circulating the token", so that the link capacity used for link maintenance during one cycle of "circulating the token" does not exceed the configured link-maintenance-token-holding-time, $V(\text{LTHT})$.

The LAS DLE shall repetitively use the following criteria to determine the next transaction.

- i) If there is a need to send a node-activation SPDU, then the DLE shall send such an SPDU (in a DT DLPDU).
- ii) If i) does not apply, and the LAS DLE has not sent any PN DLPDUs during the just completed cycle of "circulating the token", then the LAS DLE shall send a PN DLPDU to the next NODE DL-address to be probed, as specified in 8.13.3.
- iii) If neither i) nor ii) applies, then the LAS DLE may send any DLPDU, other than a TD, PN, PT or ES DLPDU, which it needs to send in its role as LAS.

After completing any such permitted transactions, the LAS DLE shall start a new cycle of "circulating the token".

NOTE The LAS DLE will perform as many transactions as possible, as specified by a), b) and c), provided that the transactions can be completed before the next scheduled activity.

12.2.2 Return of a delegated token; assumption of a scheduler token

When a delegated token which was delegated by a PT DLPDU is returned by an RI DLPDU, then the LAS DLE shall associate the value of the DD-parameter of that received RI DLPDU with the DLE returning the token, and use it as the minimum token delegation time for sending the next PT DLPDU to that DLE as required in 8.15.3.1d)3).

When a delegated token which was delegated by an ES DLPDU is returned by an RI DLPDU, then the LAS DLE shall check whether the delegating ES DLPDU was the last delegation of a scheduled sequence, as indicated by the encoding specified in 11.3.5.2.2a):

- a) If it was not the last delegation within the current cycle of that scheduled sequence execution, then the LAS DLE shall ignore the request for an additional delegation which is requested by the just-received RI DLPDU;
- b) Otherwise, when it was the last delegation of a scheduled sequence, then the DLE may, but need not,
 - 1) associate the value of the DD-parameter of that received RI DLPDU with the destination DL-address of that ES DLPDU;
 - 2) note the need to send another ES DLPDU with a final token use subfield equal to CONTINUE to the requesting DL-address, during an interval of otherwise-unscheduled bus capacity.

When the LAS DLE assumes a scheduler token due to link inactivity, on expiration of a link monitoring interval, then the LAS DLE shall initiate transmission of its next DLPDU such that there is no more than 14 slot-times, $14 \times V(ST)$ octet-durations, of inactivity on the local link.

The LAS DLE may measure the duration for which the token has been delegated, and if that duration expires before the token is returned, then the LAS DLE may assume the scheduler token immediately upon that expiration.

If the delegated token has not been returned, and it is time for the next scheduled activity, then the LAS DLE may assume the scheduler token immediately.

12.2.3 Receipt of a probe-response (PR) SPDU

If one or more PR DLPDUs are received in response to a PN DLPDU sent by the LAS DLE, then the LAS DLE shall process the first received PR DLPDU (see 11.3.2.1) as specified in this clause, and shall discard all of the other PR DLPDUs received during that response window.

The LAS DLE shall compare the values specified in the minimum-inter-DLPDU-delay, $V(MID)$, and maximum-response-delay-in-octets, $V(MRD) \times V(ST)$, fields of a received PR SPDU with the same parameters as configured for the local link.

If

- the responding DLE's $V(MID)$ is less than or equal to the link's configured value for $V(MID)$;
- the responding DLE's $V(MRD)$ is less than or equal to the link's configured value for $V(MRD)$,

then

- a) the LAS DLE shall form a node-activation SPDU with its $V(RID)$ field equal to the $V(RID)$ field of the received PR SPDU;
- b) the LAS DLE shall note the need to send a node-activation SPDU to the responding DLE, where that node-activation SPDU is conveyed in a connectionless DT DLPDU with
 - format 1S;

- NORMAL priority;
 - a SHORT destination DL-address equal to the NODE.0 DL-address of the responding DLE's DL-support functions;
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required node-activation SPDU as the conveyed DLSDU.
- c) after sending this DT DLPDU, the LAS DLE shall include the NODE DL-address of the responding DLE in its local-link live-list, V(LL).
- d) If the token-circulation-not-needed field, N, of the responding PR SPDU was set to 0 (no, token circulation is needed), then the LAS DLE shall include the NODE DL-address of the responding DLE in its local-link-token-circulation-list, V(TCL).
- e) The LAS DLE shall publish the live-list change in one of the following two ways:
- 1) the LAS DLE shall
 - i) increment the live-list revision-number;
 - ii) form a live-list-change SPDU (see 11.3.2.4) indicating the NODE DL-address of the affected DLE and its status;
 - iii) schedule the transmission of a connectionless DT DLPDU with
 - format 1S,
 - NORMAL priority,
 - a SHORT destination DL-address equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2),
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2),
 - a null SD-parameter field,
 - the required live-list-change SPDU as the conveyed DLSDU; or
 - 2) the LAS DLE shall append the NODE DL-address of the affected DLE and its status to an existing DT DLPDU conveying a live-list-change SPDU, formed as in 1) and 2), which has previously been scheduled for transmission but has not yet been sent.
- f) If the PR DLPDU specified a DLE class of Link Master or Bridge for the responding DLE, the LAS may consider that information in determining whether to request the new DLE to assume the LAS role after synchronizing with the existing schedule (for example, because the current LAS is a Link Master DLE and the responding device is a Bridge DLE).

Otherwise, when either

- the responding DLE's V(MID) is greater than the link's configured value for V(MID), or
- the responding DLE's V(MRD) is greater than the link's configured value for V(MRD),

then the DLE shall notify its local DL-management.

12.2.4 Lack of response to a PT DLPDU

If a DLE does not send any DLPDU in response to a PT DLPDU addressed to the DLE, as specified in 8.15.4.1, then the LAS DLE shall note the event. After three such non-responses for any specific NODE DL-address within three consecutive cycles of "circulating the token", the LAS DLE shall

- a) remove that NODE DL-address from the LAS DLE's local-link token-circulation-list, V(TCL), and local-link live-list, V(LL);
- b) publish the live-list change as specified in clause 11.

12.2.5 Receipt of a live-list-request SPDU

The LAS shall

- a) form a live-list-detail SPDU (see clause 11);
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the live-list-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required live-list-detail SPDU as the conveyed DLSDU.

12.2.6 Receipt of a relinquish-LAS-role-request SPDU

If the LAS had already noted the need to transfer the LAS role, then the LAS shall ignore the received SPDU, else the LAS shall note the need to transfer the LAS role.

12.2.7 Other link-maintenance requirements

Whenever there is a change in the LAS DLE's schedule, or whenever an LM DLE other than the most recent LAS DLE assumes the scheduler role, or periodically such that the time between two successive distributions of the LAS database by means of LAS-data-base-status SPDUs is equal to or more than the LAS-data-base-distribution-period, $V(LDDP)$ (see 6.7.5.20) and less than $1,5 \times V(LDDP)$, then the LAS shall

- a) form an LAS-database-status SPDU (see clause 11);
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required LAS-database-status SPDU as the conveyed DLSDU.

12.2.8 Receipt of a link-master-parameters-request SPDU

The LAS shall

- a) form a link-master-parameters-reply SPDU (see 11.3.2.12);
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the link-master-parameters-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required link-master-parameters-reply SPDU as the conveyed DLSDU.

12.2.9 Receipt of a token-hold-time-request SPDU

If the starting NODE DL-address field of the received SPDU is 00, then the LAS DLE shall

- a) form as many token-hold-time-array SPDUs (see 11.3.2.14) as necessary to send the maximum-token-hold-time for all DLEs represented in the local-link-live-list, V(LL);
- b) form one token-hold-time-array SPDU with a starting NODE DL-address field with a value of zero;
- c) for each token-hold-time-array SPDU, schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the token-hold-time-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required token-hold-time-array SPDU as the conveyed DLSDU;
 such that the token-hold-time-array SPDU with a starting NODE DL-address field with a value of zero is sent after all of the other token-hold-time-array SPDUs have been sent.

Otherwise, when the starting NODE DL-address field of the received SPDU is $N0_{16}$, where $N = 1$ to F_{16} , then the LAS DLE shall

- d) form one token-hold-time-array SPDU with a starting NODE DL-address field equal to the starting NODE DL-address field of the received SPDU;
- e) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the token-hold-time-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required token-hold-time-array SPDU as the conveyed DLSDU.

Else, the received SPDU is invalid and shall be ignored.

12.2.10 Receipt of a schedule-summary-request SPDU

The LAS shall

- a) form a schedule-summary SPDU (see 11.3.5.1);
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - TIME-AVAILABLE priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the schedule-summary-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);

- a null SD-parameter field;
- the required schedule-summary SPDU as the conveyed DLSDU.

12.2.11 Receipt of a subschedule-request SPDU

If the schedule version-number field of the received SPDU is equal to the schedule version-number field of the currently active schedule, then

- a) If the subschedule identifier field of the received SPDU is 00, then the LAS DLE shall
- 1) form as many subschedule SPDUs (see 11.3.5.2) as necessary to send all of the subschedules of the currently active schedule;
 - 2) form one subschedule SPDU with a subschedule identifier field with a value of zero;
 - 3) for each subschedule SPDU, schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - TIME-AVAILABLE priority;
 - a SHORT destination DL-address equal to source DL-address of the DLPDU which conveyed the subschedule-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required subschedule SPDU as the conveyed DLSDU;

such that the subschedule SPDU with subschedule identifier field equal zero is sent after all other subschedule SPDUs are sent.

- b) Otherwise, when the subschedule identifier field of the received SPDU is non-zero and equal to the subschedule identifier of one of the subschedules of the currently active schedule, then the LAS DLE shall
- 1) form one subschedule SPDU with a subschedule identifier field whose value is equal to the value of the subschedule identifier field of the received SPDU;
 - 2) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - TIME-AVAILABLE priority;
 - a SHORT destination DL-address equal to source DL-address of the DLPDU which conveyed the subschedule-request SPDU, or equal to the group DL-address for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
 - a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
 - a null SD-parameter field;
 - the required subschedule SPDU as the conveyed DLSDU.
- c) Otherwise, when the subschedule identifier field of the received SPDU is non-zero and is not equal to the subschedule identifier of any of the subschedules of the currently active schedule, then the LAS DLE shall
- 1) form a schedule-summary SPDU (see 11.3.5.1);
 - 2) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - TIME-AVAILABLE priority;
 - a SHORT destination DL-address equal to the source DL-address of the DLPDU which conveyed the subschedule-request SPDU, or equal to the group DL-address

- for the DL-support functions of all LM DLE's on the local link, 0101₁₆ (see 6.3.3.2);
- a SHORT source DL-address of the local LAS, 0400₁₆ (see 6.3.3.2);
- a null SD-parameter field;
- the schedule-summary SPDU as the conveyed DLSDU.

If the schedule version-number field of the received SPDU is not equal to the schedule version-number field of the currently active schedule, then the LAS DLE shall send a schedule-summary SPDU as in c).

12.3 DL-support operation

A DLE's DL-support functions receive each DT DLPDU with the destination DL-address of NODE.0, V(TN).0 (see 8.7.4.1.1d)), addressed to that DLE. When the DLE receives such a DLPDU, the DLS-user data included in the received DT DLPDU shall be interpreted as an SPDU as specified in clause 11:

- a) If the received SPDU is a DL-address query SPDU, then the DLE shall process it as specified in 10.2.3.3 or 10.3.4.3 depending upon the DL-address in the SPDU.
- b) If the received SPDU is a DL-address-report SPDU, then the DLE shall process it as specified in 10.2.3.4 or 10.3.4.4 depending upon the DL-address in the SPDU.
- c) If the received SPDU is a schedule-cancelled SPDU, then the DLE shall process it as specified in 10.4.3.3 or 10.4.3.4.
- d) If the source address of the received DT DLPDU is the default address for the local-link's LAS, 4.0, and if the received SPDU is a node-activation SPDU, then the DLE shall process it as specified in clause 11.
- e) If the received SPDU is an LAS-database-status SPDU, then the DLE shall process it as specified in clause 11.
- f) If the received SPDU is a live-list-change SPDU, then the DLE shall process it as specified in clause 11.
- g) If the received SPDU is a live-list-detail SPDU, then the DLE shall process it as specified in clause 11.
- h) If the received SPDU is a live-list-request SPDU, then the DLE shall process it as specified in clause 11.
- j) If the received SPDU is a relinquish-LAS-role SPDU, then the DLE shall process it as specified in clause 11.
- k) If the received SPDU is a link-master-parameters-reply SPDU, then the DLE shall process it as specified in clause 11.
- m) If the received SPDU is a token-hold-time-array SPDU, then the DLE shall process it as specified in clause 11.
- n) If the received SPDU is a link-master-parameters-request SPDU, then the DLE shall process it as specified in clause 11.
- p) If the received SPDU is a token-hold-time-request SPDU, then the DLE shall process it as specified in clause 11.
- q) If the received SPDU is a schedule-summary-request SPDU, then the DLE shall process it as specified in clause 11.
- r) If the received SPDU is a subschedule-request SPDU, then the DLE shall process it as specified in clause 11.
- s) If the received SPDU is a schedule-summary SPDU, then the DLE shall process it as specified in clause 11.
- t) If the received SPDU is a subschedule SPDU, then the DLE shall process it as specified in clause 11.
- u) If the received SPDU is a scheduling-completed SPDU, then the DLE shall process it as specified in 10.4.3.3.

- v) If the received SPDU is any other SPDU, then the DLE shall process it as specified in clause 11.

12.3.1 Receipt of an LAS-database-status SPDU by an LM DLE

The receiving LM DLE shall compare the live-list revision-number specified in the received SPDU with the revision-number of its own local copy of the live-list, $V(LL)$. If the two revision-numbers differ, then the LM DLE shall

- a) form a live-list-request SPDU (see 11.3.2.5);
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400_{16} (see 6.3.3.2);
 - a SHORT source DL-address equal to the $NODE.0$ DL-address, $V(TN).0$, of the sending DLE's DL-support functions;
 - a null SD-parameter field;
 - a live-list-request SPDU as the conveyed DLSDU.

A reference to the DLE's $NODE$ user-request queue, $Q_N(UR)$, shall be appended to the LM DLE's unscheduled-service queue, $Q(US)$.

The receiving LM DLE shall compare the schedule version-number specified in the received SPDU with the schedule version-number of its own local copy of the link schedule. If the two schedule version-numbers differ, and if the schedule-type field, T , in the received SPDU indicates that the LAS DLE is capable of transferring its schedule to other LM DLEs, then the LM DLE shall

- c) form a schedule-summary-request SPDU;
- d) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400_{16} (see 6.3.3.2);
 - a SHORT source DL-address equal to the $NODE.0$ DL-address, $V(TN).0$, of the sending DLE's DL-support functions;
 - a null SD-parameter field;
 - a schedule-summary-request SPDU as the conveyed DLSDU.

A reference to the DLE's $NODE$ user-request queue, $Q_N(UR)$, shall be appended to the LM DLE's unscheduled-service queue, $Q(US)$.

12.3.2 Receipt of a live-list-change SPDU by an LM DLE

If a received live-list-change SPDU (see 11.3.2.4) indicates a change in the live-list, then the receiving LM DLE shall update its own local version of the live-list in the manner indicated by the received SPDU, and shall increment its live-list revision-number.

12.3.3 Receipt of a live-list-detail SPDU by an LM DLE

The receiving LM DLE shall update

- its own local version of the local-link-live-list, $V(LL)$;
- the live-list revision-number;
- the LAS DLE's $NODE$ DL-address, $V(LN)$;
- the expected-non-response-list, $V(ENRL)$;

- the first-unpolled-node's NODE DL-address, V(FUN);
- the number-of-consecutive-unpolled-nodes, V(NUN)

as indicated by the received SPDU.

12.3.4 Request for LAS parameters by an LM DLE

A link master requires the following parameters to operate as the LAS:

- a) the local link's configuration parameters defined in 6.7.5 and included in the link-master-parameters-reply SPDU (see 11.3.2.11);
- b) the local-link-live-list, V(LL);
- c) the expected-non-response-list, V(ENRL);
- d) the token-circulation-list, V(TCL);
- e) the maximum-token-holding-time-array, V(MTHA);
- f) the local link's schedule;
- g) a sense of the current LAS DLE's DL-time.

If a link master DLE does not have the necessary parameters required to operate as the LAS, as may be the case when the link master DLE has just changed its state to ONLINE, then the link master DLE shall schedule the transmission of the following SPDUs to the local LAS in connectionless DT DLPDUs each with format 1S, NORMAL priority, a destination address of the local LAS, 0400₁₆ (see 6.3.3.2), and a source address of V(TN).00:

- a link-master-parameters-request SPDU (see 11.3.2.11);
- a token-hold-time-request SPDU (see 11.3.2.13);
- a live-list-request SPDU (see 11.3.2.5);
- a schedule-summary-request SPDU (see 11.3.5.3);

the link-master DLE shall note the need to transmit a CT DLPDU at its first opportunity.

After sending the SPDUs, the sending DLE shall wait for reception of the link-master-parameters-reply SPDU, token-hold-time-array SPDUs, live-list-detail SPDU and schedule-summary SPDU;

If these SPDUs are not received by the requesting LM DLE within 15 receptions of a PT DLPDU with a token-use-subfield equal to RESTART addressed to the DLE's NODE DL-address, then the requesting LM DLE shall again schedule the transmission of those request SPDUs appropriate for the missing reply SPDUs.

12.3.5 Receipt of a link-master-parameter-reply SPDU by an LM DLE

The DLE shall use the received SPDU to update the parameters required to operate as LAS and

- if the DLE's maximum-inactivity-to-claim-LAS-delay is less than or equal to the link's configured value of maximum-inactivity-to-claim-LAS-delay, V(MICD), then the DLE shall note that it is capable of operating as the local link's LAS.

12.3.6 Receipt of a token-hold-time-array SPDU by an LM DLE

If the starting NODE DL-address field of the received SPDU (see 11.3.2.14) is $N0_{16}$, where $N = 1$ to F_{16} then the DLE shall

- update the maximum-token-holding-time-array, V(MTHA) from the maximum-token-hold-time values received in the SPDU;

- exclude from the token-circulation-list, $V(TCL)$, those DLEs for which the maximum-token-hold-time is zero.

Otherwise, when the starting NODE DL-address field of the received SPDU is 00, then the DLE shall

- check whether it has received the token-hold-time-array SPDUs for all NODE DL-addresses in the local-link-live-list, $V(LL)$;
- if not, then send one or more token-hold-time-request SPDUs, specifying such values for the starting NODE DL-addresses in those SPDUs, that the corresponding reply SPDUs will contain the maximum-token-hold-times for all the NODE DL-addresses in local-link-live-list, $V(LL)$, whose maximum-token-hold-times are not yet known by the requesting DLE;

Otherwise, when the starting NODE DL-address field of the received SPDU is of the form NM_{16} , where M is non-zero, then the received SPDU is invalid and shall be discarded.

12.3.7 Receipt of a schedule-summary SPDU by an LM DLE

If the schedule version-number field of the received SPDU (see 11.3.5.1) is non-zero, then

- a) If either the DLE has no link schedule, or the schedule version-number of the link schedule stored in the DLE is different from the schedule version-number field of the just received SPDU, then the DLE shall
 - 1) check that it is capable of executing the link schedule whose summary was just received. The DLE shall
 - i) check that it has the capability to execute the number of subschedules in the schedule-summary;
 - ii) check that the DLE's value of maximum-scheduling-overhead, $V(MSO)$, (see 6.7.5.6) is less than or equal to the value of $V(MSO)$ specified in the schedule-summary;
 - iii) check that the DLE has the required storage capacity specified in the schedule-summary;
 - iv) check that the DLE has the required timing resolution specified in the schedule-summary;
 - v) check that the DLE's value of time-source-link, $V(TSL)$, (see 6.7.1.27) is the same as the value of $V(TSL)$ specified in the schedule-summary.

If any of these checks fail, then the DLE shall discard the just received schedule-summary SPDU.

- 2) otherwise, when these checks are all passed, then the DLE shall

- i) form a subschedule-request SPDU with its schedule version-number field equal to the value of the schedule version-number field of the just received schedule-summary SPDU, and with a subschedule identifier equal to zero;
- ii) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400_{16} (see 6.3.3.2);
 - a SHORT source DL-address equal to the NODE.0 DL-address, $V(TN).0$, of the sending DLE's DL-support functions;
 - a null SD-parameter field;
 - the just-formed subschedule-request SPDU as the conveyed DLSDU.

- b)

Otherwise, when the DLE has a link schedule and the version-number of the link schedule stored in the DLE is equal to the schedule version-number field of the just received SPDU, but the DLE does not have the subschedule for all of the subschedules identified in the just received schedule-summary SPDU, then the DLE shall

- 1) form one or more subschedule-request SPDUs, each with a schedule version-number field equal to the value of the schedule version-number field of the just received SPDU, and with a subschedule identifier equal to one of the subschedule-SPDU references of the missing subschedules;
- 2) schedule the transmission of a connectionless DT DLPDU for each such just-formed SPDU, with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400₁₆ (see 6.3.3.2);
 - a SHORT source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions;
 - a null SD-parameter field;
 - one of the just-formed subschedule-request SPDUs as the conveyed DLSDU.

Otherwise, when the schedule version-number field of the just received SPDU is zero, then the DLE shall record that there is no active link schedule.

NOTE Where resources permit, an LM DLE which is incrementally updating its copy of the local link's schedule should retain the last self-consistent prior version of that schedule until such updating is complete.

12.3.8 Receipt of a subschedule SPDU by an LM DLE

If the schedule version-number field of the just received subschedule SPDU (see 11.3.5.2) is equal to the schedule version-number field of the last received schedule-summary SPDU (see 11.3.5.1), then

- a) if the subschedule identifier field of the received SPDU is equal to a subschedule-SPDU reference included in the last received schedule-summary SPDU, then the DLE shall store the subschedule as part of the link schedule;
- b) otherwise, when the subschedule identifier field of the received SPDU is equal to zero, then the DLE shall
 - check whether it has received the subschedule SPDUs for all of the subschedule-SPDU references in the last-received schedule-summary SPDU;
 - if not, then send one or more subschedule-request SPDUs, each with a value of the subschedule identifier requesting one of the missing subschedules;
- c) otherwise, when neither a) nor b) applies, then the DLE has received a subschedule SPDU inconsistent with the DLE's last-received schedule-summary SPDU. Therefore the DLE shall
 - 1) form a schedule-summary-request SPDU;
 - 2) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400₁₆ (see 6.3.3.2);
 - a SHORT source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions;
 - a NULL SD-PARAMETER FIELD;
 - A SCHEDULE-summary-request SPDU as the conveyed DLSDU.

Otherwise, when the schedule version-number field of the just received subschedule SPDU is not equal to the schedule version-number field of the last received schedule-summary SPDU, or if the DLE had never received a schedule-summary SPDU, then the DLE shall send a schedule-summary-request SPDU as in c).

NOTE See note at end of 12.3.7.

12.3.9 Request for LAS transfer by an LM DLE

If an LM DLE receives a request from local DL-management to assume the LAS role, then

- if the DLE does not have the necessary parameters required to operate as the LAS, then it shall follow the procedure of 12.3.4 to obtain those parameters.

After receiving the necessary parameters to operate as LAS, the LM DLE shall

- a) form a Relinquish-LAS-role SPDU;
- b) schedule the transmission of a connectionless DT DLPDU with
 - format 1S;
 - NORMAL priority;
 - a SHORT destination DL-address for the local LAS, 0400₁₆ (see 6.3.3.2);
 - a SHORT source DL-address equal to the NODE.0 DL-address, V(TN).0, of the sending DLE's DL-support functions;
 - a null SD-parameter field;
 - a schedule-summary-request SPDU as the conveyed DLSDU;
- c) wait for reception of a TL DLPDU;
- d) if a TL DLPDU is not received within 15 receptions of a PT DLPDU with a token-use-subfield equal to RESTART, then the LM DLE shall retry this process commencing with step a);
- e) if a TL DLPDU is received, but is rejected as specified in 8.20.4.2, then the LM DLE shall assume that it does not have the necessary parameters required to operate as the LAS and shall repeat the procedures of 12.3.9;
- f) if a TL DLPDU is still not received after two retry attempts (as in d) or e)), then the LM DLE shall inform local DL-management of this failure.

12.4 DL-bridge elements of procedure and bridge sub-protocol

Two compatible versions of a bridge sub-protocol are defined. The first, based on ISO/IEC 10038, supports adaptive bridges which require almost no configuration, but requires all the nodes on the local link implement the full DL-support protocol of 12.3.

The second version of the bridge sub-protocol, based on a subset of ISO/IEC 10038, supports non-adaptive bridges which require extensive configuration, but eliminates the need for all the nodes on the local link to implement the full DL-support protocol of 12.3.

12.4.1 Common features of the Type 1 bridge protocols and elements of procedure

Bridges provide the following capabilities:

- a) Selective forwarding of DLPDUs.
- b) Selective subscription to, and republishing of, published DLSDUs.
- c) Selective propagation of DL-time based on port state.

Bridges conforming to 12.4.1.4 also

- d) Exchange information to self-configure into a spanning tree.

- e) Observe traffic and query DLEs on the local links to self-configure their forwarding databases.

Bridges may conform in an operational sense to both this 12.4.1.4 and 12.4.3; the local method by which they reconcile conflicting management paradigms is not specified.

12.4.1.1 Introduction

Local links of all data rates may be connected together with bridges, which are DL-relay entities. Each individual link has its own independent schedule and operation. The extended link created by the bridges allows the interconnection of end systems attached to separate links as if they were attached to a single unified link. However, time and event synchronization among end systems on the extended link is weaker than on any single constituent link.

Local links of different speeds and media types may be bridged together to form multi-link, or *bridged*, networks. Bridges are used in these networks to interconnect links. In all fieldbusbridged networks, at any moment there is only one operational DLPDU forwarding path between any two DLEs. To ensure this, routing tables within bridges are coordinated with each other to form a *spanning tree*.

The spanning tree represents a configuration of bridges, such that there are only two directions of data flow, either towards the root of the tree, or away from it. There are no loops and no parallel routes. That is, for each pair of links there must be one and only one active bridge leading to the root.

Each bridge in the spanning tree has a single root port and one or more downstream ports. Each bridge port represents a connection to one link. Root ports lead toward (or may themselves be) the root of the spanning tree; downstream ports lead away from the root.

When a message is received at a port, the bridge uses local routing tables to determine whether it is in the path connecting the link from which the message was received and one or more of the links to which the message is to be sent.

When a message is received on a downstream port, the bridge routes it to the appropriate other port(s) based on routing tables within the bridge. The figure below illustrates a bridged network and a spanning tree configuration for it.

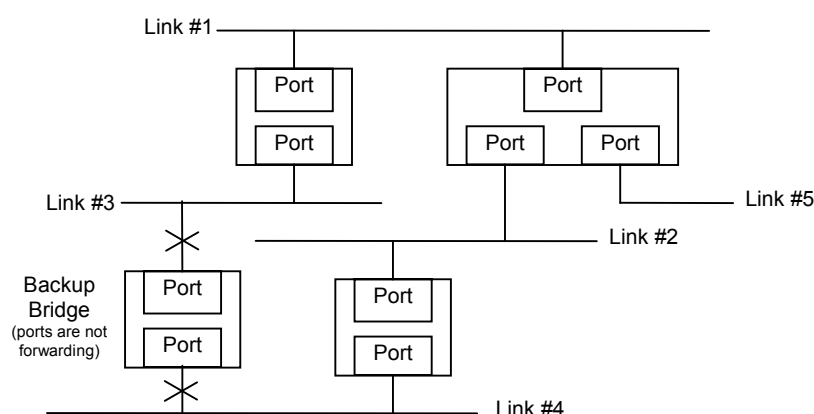


Figure 16 – Bridged network topology

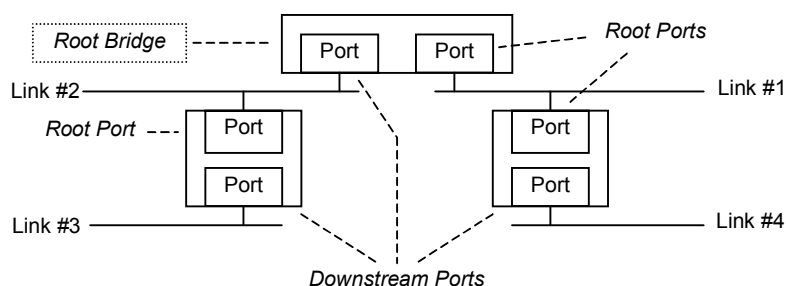


Figure 17 – Spanning tree representation

All bridge DLEs are responsible for performing these DLL activities:

- a) forwarding,
- b) republishing,
- c) data link time synchronization.

Bridge DLEs conforming to 12.4.2 are also responsible for

- d) self-configuring into a spanning tree,
- e) dynamically establishing most of the routing information in their forwarding databases.

To support c), it is necessary for bridge ports to be the Link Active Scheduler (LAS) of their downstream links. Thus the spanning tree connectivity drives the LAS selection procedure. In the figure above, each active downstream port of a bridge functions (and is required to function) as the LAS for its associated link.

Forwarding occurs when a bridge receives a DLPDU with a first DL-address for which the bridge has a rule in its forwarding database. If the address indicates that the message is to be forwarded, the bridge *appends* the message to the FIFO output queue for the appropriate port(s). When the bridge receives the right to transmit from the LAS associated with that port, the bridge transmits entries from the queue in priority order.

Republishing occurs when the bridge is configured to republish DLSDUs. When the bridge receives a DLSDU which it is to republish, it buffers the DLSDU for scheduled transmission on the appropriate outbound ports as specified in its republishing database. The DLSDU is *republished* separately onto each of the outbound links.

Synchronization of DL-time occurs when a DL-time DLPDU is received by a bridge on its root port. DL-time originates with the root port of the root link. When a DLE receives a TD (DL-time) DLPDU, it recalculates the offset for the link scheduling time of each of its outbound ports for which it is the LAS. If the bridge contains a downstream port for which it is not the LAS, then the DL-time cannot be redistributed to that link by that bridge.

A bridge provides the normal DL-functions for each port, plus specialized forwarding, republishing, time propagation and bridge coordination functions. A bridge operates below the DL-Service boundary, and is transparent to protocols operating above this boundary. An extended link may provide the following:

- 1) An effective increase in the physical extent, the number of permissible attachments, or the total performance of a link over that possible without use of bridges.
- 2) Partitioning of the physical link for administrative, maintenance, fault limiting or security reasons.
- 3) Integration of individual links of different data rates into a single coherent network.

NOTE A *bridge* is a DL-relay entity, and should not be confused with a *repeater*, which is a PhL-relay entity. Bridges selectively forward DLPDUs and usually introduce significant delay in the forwarding process; repeaters forward PhPDUs unselectively with almost no delay. Bridges have substantial internal state; repeaters have almost none. Bridges can interconnect *links* of different data rates; repeaters interconnect *segments* of identical data rate, possibly with different modulation or types of media.

12.4.1.2 Extent of specification

This subclause specifies a general method for the operation of bridges. Bridges provide compatible interconnection of automation equipment that implement the corresponding 61158-4, Type 1 protocol. This subclause

- a) positions the bridging function within an architectural description of the DLL;
- b) defines the principles of operation of a bridge in terms of the support and preservation of the DL-Service, and the maintenance of Quality of Service (QoS);
- c) identifies the functions to be performed by bridges, and provides an architectural model of the internal operation of a bridge in terms of processes and entities that provide these functions;
- d) specifies a minimal protocol between bridges in the extended link, such that the resulting bridges (see 12.4.3) interoperate with bridges that meet the full requirements of 12.4.1.4.

The specification of Remote Bridges, which interconnect Type 1 links using other communications protocols for the transmission of DLPDUs among bridges, is outside the scope of this specification.

12.4.1.3 Support of the DL-Service.

Bridges interconnect separate links into an integrated network, known as the *extended link*, by

- a) forwarding DLPDUs between the separate links;
- b) subscribing to information being published on one link, and republishing it on one or more of the other links to which the bridge connects;
- c) providing a synchronized sense of DL-time among the links to which the bridge is directly connected and currently forwarding, or for which the bridge is currently acting as LAS.

The DL-Service is provided to the DLS-user in the end stations and is supported by the forwarding function in the bridge. This subclause discusses provisions of the DL-Service to end stations, support of the DL-Service by bridges, preservation of the DL-Service offered on the extended link, and maintenance of QoS in the extended link.

The DL-service provided to end stations attached to an extended link is supported by the bridges in that extended link. Bridges support all of the DL-services that involve multi-end-station communication.

The style of bridge operation maximizes the availability of the DL-Service to end stations and assists in the maintenance of the extended link. It is therefore desirable that bridges be capable of being configured in the extended link:

- 1) To provide redundant paths between end stations to enable the extended link to continue to provide the DL-Service in case of component failure (of bridge or basic link).
- 2) So that the paths supported between end stations are predictable and configurable given the availability of components of the extended link.
- 3) So that DLPDUs on one link are forwarded selectively to other links, with the forwarding decisions based on configured criteria.
- 4) So that DLSDUs of a DLC on one link are republished selectively to DLCs on other links, with the subscribing and republishing relationships based on configured criteria.
- 5) So that bridges implementing the full Spanning Tree protocol of 12.4.1.4 defer to the non-adaptive bridges of 12.4.3 when constructing the spanning tree for the extended link.

12.4.1.3.1 Preservation of the DL-Service.

The DL-Service offered by an extended link, consisting of single links interconnected by bridges, is similar to that offered by a single link. In consequence,

- a) A bridge is not directly addressed by communicating end stations when forwarding DLPDUs between links; DLPDUs transmitted between end-stations carry DL-addresses of end-stations in their address fields, not the DL-address of the bridge. However, the bridge itself serves as an end-station when republishing, and when providing access to management agents and higher-layer entities collocated with the bridge.
- b) All DL-addresses must be unique and addressable within the extended link.
- c) The topology and configuration of the extended link do not otherwise restrict the DL-addresses of end stations.

12.4.1.3.2 Quality of Service (QoS) maintenance.

Bridges are designed so that the quality of the DL-Service supported by a bridge is not significantly inferior to that provided by a single link. The QoS parameters considered are those related to:

- a) Service availability
- b) Shared sense of DL-time
- c) DLPDU and DLSDU loss
- d) DLPDU and DLSDU misordering
- e) DLPDU and DLSDU duplication
- f) DLPDU transit delay
- g) DLPDU lifetime
- h) Undetected error rate in forwarded DLPDUs and republished DLSDUs
- i) DLPDU and DLSDU priority
- j) DLSDU timeliness
- k) Throughput
- l) Schedule skew when republishing DLSDUs

NOTE Bridges forward DLPDUs and republish DLSDUs. Therefore, to a first approximation, QoS related to DLPDUs applies to forwarding, while QoS related to DLSDUs applies to republishing.

12.4.1.3.2.1 Service availability.

The DL layer provides the DL-Service to end stations attached to a single link or extended link. Service availability is measured as that fraction of some total time during which the service is provided. The operation of a bridge can increase or decrease the service availability.

The service availability can be increased by automatic reconfiguration of the extended link in order to avoid the use of a failed component (that is, a repeater, cable or connector) in the data path. The service availability can be lowered by failure of a bridge itself, through denial of service by a bridge, or through undesired DLPDU discard by a bridge during intervals of bridge resource saturation.

To maximize the service availability, no loss of service or delay in service provision should be caused by bridges, except as a consequence of a failure, removal or insertion of a component of the extended link. This is regarded as an extraordinary event.

NOTE A loss of service caused by congestion at an outbound bridge port is not induced by the bridge, but by the structure and use of the extended link. Such loss can be expected, particularly where a higher-capacity link is forwarding to a much lower capacity one. In contrast, a loss of service caused by congestion at an inbound bridge port is attributable to the design of the bridge itself, in conflict with the above recommendation.

12.4.1.3.2.2 Shared sense of DL-time

The DLL provides a shared sense of DL-time. The degree of time synchronization of any two end-stations on the extended link is limited by the characteristics of the communications path between them. Synchronization degrades with an increasing number of intermediate bridges.

12.4.1.3.2.3 DLPDU and DLSDU loss.

The DL-Service does not guarantee the delivery of DLSDUs. However, some DLCs notify their associated DLS-user of the detection of DLSDU loss.

DLPDUs transmitted by a source station arrive, uncorrupted, at the destination station with high probability. The operation of a bridge introduces minimal additional DLPDU loss.

A DLPDU transmitted by a source station can fail to reach its destination station as a result of

- a) DLPDU corruption during Physical Layer transmission, relaying or reception;
- b) DLPDU discard by a bridge because
 - 1) it is unable to transmit the DLPDU within some maximum specified time and, hence, must discard the DLPDU to prevent the maximum DLPDU lifetime (12.4.1.3.2.7) from being exceeded;
 - 2) it is unable to continue to store the DLPDU due to exhaustion of internal buffering capacity as DLPDUs continue to arrive at a rate in excess of that at which they can be processed and forwarded.

12.4.1.3.2.4 DLPDU and DLSDU misordering.

The DL-Service does not guarantee the delivery order of DLSDUs unless that ordering has been negotiated as part of the QoS for a DLC. However, DLPDUs of the same DL-priority for any given sending and receiving port pair are not misordered within a bridge; when neither DLPDU is discarded, the earlier-received DLPDU will be the earlier forwarded.

Misordering can occur immediately after any reconfiguration of the DL-path between two end stations involving changes to two or more bridges in the DL-path. Reconfiguration involving one bridge substituting for another parallel bridge that is directly connected to the same links does not cause misordering.

12.4.1.3.2.5 DLPDU and DLSDU duplication.

Except for UNORDERED DLCs, the DL-Service does not permit the duplication of DLSDUs. Normal operation of bridges does not introduce duplication of DLPDUs.

The potential for DLPDU duplication in an extended link arises from the possibility of multiple paths between source and destination end stations. This can only arise due to erroneous configuration or malfunction of a bridge, including failover of a bridge to a replacement bridge that is using an anticipatory forwarding strategy.

12.4.1.3.2.6 DLSDU transit delay.

The DL-Service introduces a DLSDU transit delay that is dependent on the particular media and link transmission utilization of the DL-path employed. DLSDU transit delay is measured at the DLS interface (see fFigure 18). DLSDU transit delay is the elapsed time between

- a) the sending DLE's receipt of a DLS primitive requesting the transmission of DLS-user data, or the equivalent event triggering transmission from a publisher DLCEP when no DLSDU send primitive is involved, and
- b) the receiving DLE's signaling of the availability of the DLS-user data at its DLS interface, or the equivalent event signaling reception at a subscriber DLCEP when no DLSDU delivery primitive is involved.

Elapsed time values are computed only on DLSDUs that are successfully transferred.

Since the DLS generally is provided at an abstract interface within the end station, it is not possible to specify precisely the total DLSDU transit delay. It is, however, possible to measure those components of delay associated with access to the medium and with transmission and reception. Likewise, those components of the transit delay introduced by a bridge can be measured.

The minimal additional transit delay introduced by a bridge is the sum of:

- 1) the time taken to (wholly or partially) receive a DLPDU conveying the DLSDU and determine that it needs to be forwarded;
- 2) any time required to prepare the DLPDU for forwarding;
- 3) the time taken to access the medium onto which the DLPDU is to be forwarded.

NOTE 1 If the DLPDU is completely received before it is forwarded, and if a frame check sequence (FCS) error is detected in the received DLPDU, then that DLPDU will be discarded without forwarding.

NOTE 2 If the DLPDU is only partially received before forwarding commences (known as “cut-through switching”), the delay induced by the bridge can be as small as a few octet-durations of the link from which the DLPDU is received.

NOTE 3 The concept of transit delay applies only to the bridge's forwarding activities, and not to its republishing or time-propagation activities.

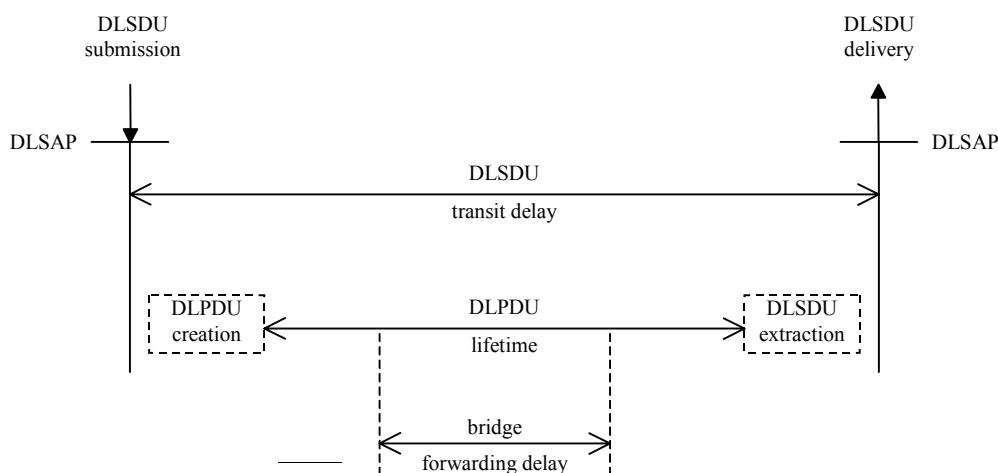


Figure 18 – DLSDU transit delay, DLPDU lifetime and bridge forwarding delay

12.4.1.3.2.7 DLPDU lifetime.

The DL-Service ensures that there is an upper bound to the transit delay experienced for each DLS-instance. To ensure this, the DLS imposes a maximum lifetime on each DLPDU conveying a DLSDU. DLPDU lifetime is measured within the distributed DLE, from the moment when a DLSDU is packaged into DLPDUs for transmission, until the DLSDU is extracted after DLPDU reception (see Figure 18).

To ensure an upper bound to the DLPDU lifetime, both originating end stations and bridges are required to discard any DLPDU that has been queued for transmission for an excessive amount of time. Within a bridge, this queuing time is known as *forwarding delay*, and is measured from the moment when the DLPDU is completely received (the end-of-reception event) to the moment that the DLPDU retransmission commences (the start-of-transmission or transmit-commit event).

Since the information provided by the DL-Protocol to a bridge does not include the transit delay already experienced by any individual DLPDU, bridges must enforce a configurable maximum forwarding delay in each bridge, based only on DLPDU priority. They do this by discarding a DLPDU based on the length of time the DLPDU has been in the bridge, waiting for retransmission as a forwarded DLPDU.

The value of the maximum permissible forwarding delay for any given bridge is configurable for each DLPDU priority, and is based on

- a) the desired maximum DLPDU lifetime for that priority class of DLPDUs, and
- b) the set of DL-paths on the extended link which go through that bridge.

12.4.1.3.2.8 Undetected error rate in forwarded DLPDUs and republished DLSDUs.

Protection against undetected errors is provided by the use of a FCS that is

- a) included at the end of each DLPDU by the sending DLE prior to transmission;
- b) propagated by any bridges, possibly after altering the FCS algorithmically to compensate for the complementation of a single fixed bit in the first octet of the DLPDU, as specified in 5.1.3;
- c) checked by all receiving DLE(s).

DLSDUs retained within the bridge for republishing are also subject to errors due to malfunction of the bridge hardware or software, which can corrupt the DLSDU while within the bridge. The FCS propagation algorithm for forwarded DLPDUs, which is the basis for the end-to-end error detection of this protocol, is not readily applicable to DLSDUs that are republished. A bridge need not detect corruption of DLSDUs awaiting republication, nor need the FCS of any republishing DLPDU be a transformation of the FCS of a corresponding subscribing DLPDU.

12.4.1.3.2.9 DLPDU and DLSDU priority.

Bridges do not alter the priority of received and forwarded DLPDUs. Bridges do not alter the priority of republished DLSDUs.

12.4.1.3.2.10 DLSDU timeliness.

Bridges implement DLSDU timeliness computations on republished DLSDUs according to their configuration as subscribers and republishers. In most cases bridges provide TRANSPARENT timelines, propagating whatever sense of untimeliness was received to those DLEs to whom they republish.

12.4.1.3.2.11 Throughput.

Bridges localize traffic within the extended link by forwarding only selected DLPDUs and republishing only selected DLSDUs. This permits the total capacity of the extended link (in DLSDUs transferred per second) to be significantly greater than that provided by an equivalent single link.

12.4.1.3.2.12 Schedule skew when republishing DLSDUs.

A bridge may require some minimum time after receipt of a DLSDU at a subscriber DLCEP before that same DLSDU is available for republication at a publisher DLCEP on another port. The scheduler for the republishing link may need to have an upper bound on this time interval. Therefore each bridge has a MinRepublishingDelay attribute that specifies the minimum time which the scheduler must allocate for propagation of the DLSDU from the port of the bridge at which it was received to the port of the same bridge at which it is to be republished.

12.4.1.3.3 Forwarding of DLPDUs

A bridge receiving a DLPDU examines the DLPDU's frame control octet and first address, from which the bridge determines whether and to where the DLPDU should be forwarded. Forwarding decisions are based on information in the bridge's filtering database.

A bridge that claims full conformance to 12.4.1.4 forwards a CA, CD, DC, DT, EC, ED or RC DLPDU, together with its received FCS, toward those links that have DLEs that should receive the DLPDU's first DL-address. If the received DLPDU is a CA, CD or ED DLPDU, then such a bridge replies immediately with a SR DLPDU on the link from which the CA, CD or ED DLPDU was received.

No other forwarding of DLPDUs occurs.

12.4.1.3.4 Republishing of DLSDUs

Bridges are configured as subscribers to receive DLSDUs which they also may be configured to republish.

A subscribing bridge completes DLC establishment with its publisher (which could be another bridge acting as a republisher), after which it receives published DT DLPDUs, as do other subscribers. The bridge extracts the DLSDUs and timeliness status from these received DLPDUs, buffers the extracted information, and republishes it on DLCs for which the bridge is configured as a publisher and has completed DLC establishment.

The bridge's Republishing Database is configured with the DLC characteristics of each republishing DLC for which the bridge is responsible, and of the corresponding subscriber DLC which provides the DLSDUs for republishing.

12.4.1.3.5 Propagation of DL-time.

Bridges are configured so that the aggregate set of bridge ports that are in the forwarding state form an acyclic graph, known as a *spanning tree*, which connects all of the links of the extended link.

A bridge attempts to synchronize the sense of DL-time among those ports that are in the FORWARDING state (see 12.4.1.4 and 12.4.3) or are acting as LAS.

If all of the bridge's forwarding ports are acting as LAS, then the bridge is at the *root* of the spanning tree and it propagates its own internal sense of DL-time to all of those ports which are acting as LAS. A bridge that also has a gateway to another network may synchronize its internal sense of DL-time to a time sense received from that other network. In this case, the bridge functions as the root of the spanning tree for its local subnetwork.

Otherwise, the bridge has precisely one port that is forwarding and that is not acting as LAS; this port is designated the *root* port of the bridge. The bridge propagates DL-time from its root port to the ports that are acting as LAS, which includes all of its other ports which are in the FORWARDING state.

NOTE Any ports which are not in the FORWARDING state and which are not acting as LAS will have their time sense synchronized to their attached link, rather than to the internal time sense of the bridge.

12.4.1.4 Principles of operation

This clause establishes the principles of operation, and a model for the operation, of a bridge as follows:

- a) Provides a model of operation of a bridge, explains the principal elements of bridge operation and lists the functions that support these.

- b) Discusses the architectural model for a bridge, found in 6.1.2, which governs the provision of these functions. This includes specifying the states of a bridge port, and the format and associated protocol for BPDUs that facilitate interoperation with spanning tree bridges.
- c) Details the addressing requirements of an extended link and specifies the addressing of entities in a bridge.

12.4.1.4.1 Bridge operation.

The principal elements of bridge operation that differ from the operation of other DLEs are

- a) Selectively forwarding received DLPDUs.
- b) Selectively republishing received DLSDUs on other DLCs, and subscribing to other DLCs to receive these DLSDUs.
- c) Selective propagation of a shared sense of DL-time.
- d) Minimal participation in the inter-bridge spanning tree protocol of 12.4.1.4.

NOTE 1 Conceptual models of DLPDU reception, DLPDU forwarding, local DLPDU origination and DLSDU republishing are intended to facilitate an understanding of these processes.

NOTE 2 DLPDUs addressed to any port of a bridge are addressed to the bridge itself. The filtering process is not involved in their local delivery, and the requirements for forwarding do not apply

12.4.1.4.1.1 Filtering and forwarding.

A bridge forwards individual DLPDUs between the separate links connected to its *ports*. The decision of whether to forward, and to which ports to forward, is based on information in a filtering database.

NOTE The term filtering is used in this sense as a permissive filter, because that is the sense of the information configured in the filtering database. The more common language sense of filtering for discard applies to the set of unselected ports, which is the inverse of the database information.

The order of DLPDUs of a given DL-priority received on one port and retransmitted on another is preserved.

The functions that support the forwarding of DLPDUs and maintain the QoS supported by the bridge are

- a) Reception of all DLPDUs that have no detected error, including retention of the DLPDU's FCS as received.

NOTE This is a generalization of the basic reception function found in all DLEs.

- b) Submission of received DLPDUs to the filtering process for potential forwarding if
 - 1) the DLPDU contains one or more long addresses whose link designators are all non-zero,
 - 2) and
 - i) the DLPDU type is CA, CD, DC, DT, EC, ED or RC, or
 - ii) the DLPDU is a DT DLPDU that was created in accordance with 8.7.4.3.1 or 8.8.4.3 b).
- c) Filtering of locally originated DLPDUs based on the first address of the DLPDU, with forwarding to any or all of the ports of the bridge as specified in the filtering database.
- d) DLPDU discard to ensure that a maximum forwarding delay is not exceeded.
- e) Possible modification of a DLPDU's frame control octet, with compensatory modification of the DLPDU's FCS.
- f) FCS calculation on those DT DLPDUs created as specified by b)2)ii).

NOTE This is part of the basic transmission function found in all DLEs.

- g) DLPDU transmission, with transmission order determined both by DLPDU priority and by forwarding delay (for example, the order in the forwarding queue), and with the (modified) forwarded FCS used where available.

NOTE This is a generalization of the basic transmission function found in all DLEs.

12.4.1.4.1.1.1 Forwarding of a received DLPDU.

A lower-level port-specific process receives a DLPDU, validates its FCS, and classifies the DLPDU based on its frame control octet. If the received DLPDU contained one or more explicit DL-addresses, and all of those addresses were LONG with non-zero link designators, and the link-designator of the first DL-address is in the forwarding database of the DLE, then the DLPDU record is passed to the lower-level transmission process of each port associated with that link designator in the forwarding database, provided that that port is in the FORWARDING state and that it is not the port on which the DLPDU was received.

DLPDUs are forwarded based on DLPDU priority and their order or length of time in the forwarding queue. A new FCS is never calculated for a forwarded DLPDU. However, the received FCS may be modified before forwarding as specified in 5.1.3.

DLPDUs that have been in the forwarding queue longer than the configured MaxForwardingDelay for the DLPDU's priority are discarded.

Statistics and diagnostic information should be kept as specified in 12.5.6.3.

Other aspects of DLPDU reception and subsequent local delivery are identical to those of a non-bridge DLE.

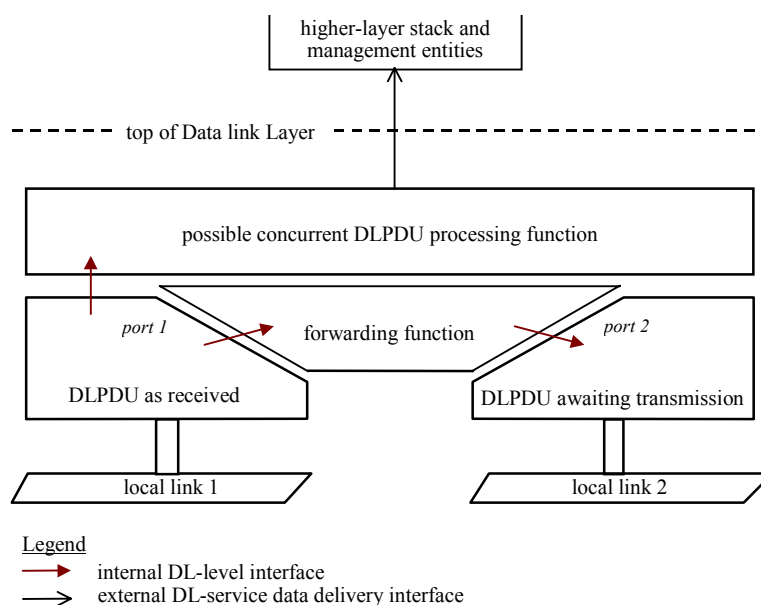


Figure 19 – Forwarding and delivering a received DLPDU

NOTE Figure 19 shows possibly-concurrent forwarding and delivery paths.

12.4.1.4.1.1.2 Local origination of a DLPDU.

A higher-level process creates a DLPDU and passes it to the lower-level transmission process of each port associated in the forwarding database with the link designator of the DLPDU's first DL-address, whether or not that port is in the FORWARDING state. Other aspects of local origination and local delivery are identical to those of a non-bridge DLE.

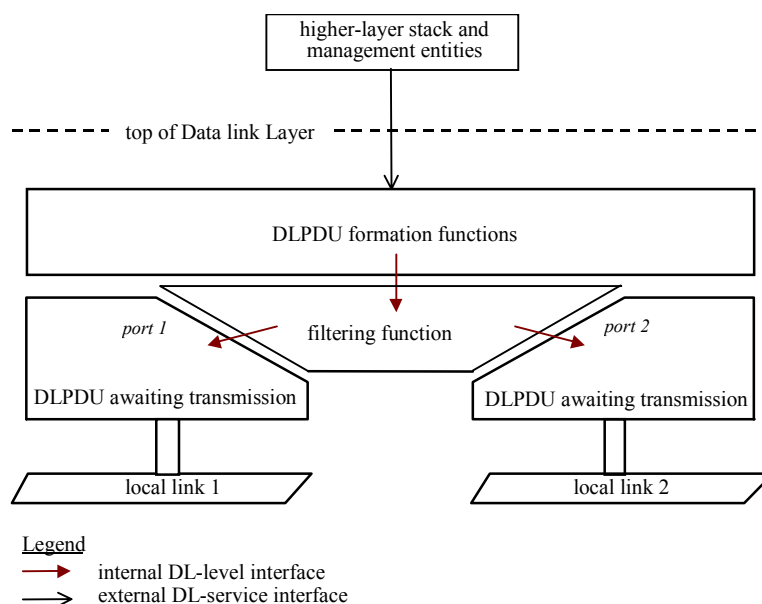


Figure 20 – Forwarding a locally-originated DLPDU

12.4.1.4.1.2 Subscribing to one DLCEP and republishing on another DLCEP.

Each bridge may function as a republisher of published data. The bridge subscribes to one DLC and republishes received DLSDUs on one or more other DLCs as specified by its Republishing Database.

Each entry in the Republishing Database specifies 32-bit delocalized subscriber and publisher DLCEP-addresses and the characteristics of the subscribed-to DLC. The latter are also used as the characteristics of the republishing DLC.

NOTE When periodic republishing is required on a port, the link schedule for that port's link must contain an entry to compel data transmission by the republishing DLCEP.

At port initialization, or whenever the state of a port is changed to FORWARDING, the bridge management entity (BME) initiates all (re)publishing DLCEPs described in its Republishing Database that have a delocalized DLCEP-address whose DL-link designator is associated with that port by the Filtering Database.

At port initialization, the BME initiates all subscribing DLCEPs described in its Republishing Database. Multiple entries in the Republishing Database for a single subscriber DLCEP-address may result in the establishment of only a single subscribing DLCEP.

DL-buffers are allocated as necessary during this process, so that each subscribing DLCEP and all associated (re)publishing DLCEPs are bound to the same DL-buffer.

If the bridge's DL-management changes the state of a port from FORWARDING to another state, the bridge management entity terminates republishing on the DLCEPs associated with that no-longer-forwarding port. The bridge may, but need not, send DC DLPDUs on those republishing DLCs which are being terminated.

NOTE When management action is the mechanism for switching among the bridges of a redundant set of bridges, transmission of a separate DC DLPDU on each of the bridge's open republishing DLCs is not recommended.

The Bridge management entity is not involved in the actual process of republishing DSLDUs. As each new DLSDU is received at a subscribing DLCEP, it is written to the DL-buffer. Writing the DL-buffer and its associated timeliness information is the event which signals each connected republishing DLCEP that the DL-buffer has newly-updated contents.

NOTE Reception and detection of a duplicate DLSDU at a subscribing DLCEP results in no update of the buffered DLSDU and no change in that buffered DLSDU's timeliness attribute.

The rates at which DLSDUs are written to the DL-buffer by the subscribing DLCEP, and are read from the buffer by the publishing DLCEP(s), are determined by the associated link schedules. The two rates need not be identical.

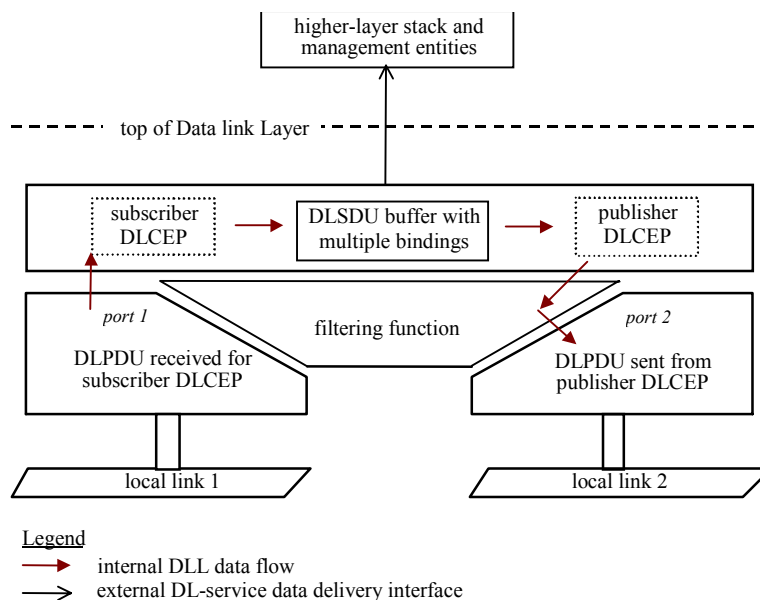


Figure 21 – Republishing a DLSDU received from another link

NOTE The possibly-concurrent forwarding path, from one port to another, for the DLPDU received at the subscribing DLCEP is not shown. It is identical to the forwarding path of figure C-2.

12.4.1.4.1.3 DL-time propagation.

Each bridge functions as a propagator of DL-Time. The bridge adjusts its internal sense of DL-time to that of the single port, if any, which is in the FORWARDING state and is not acting as LAS. (This is necessarily the root port of the bridge.)

The bridge adjusts the sense of DL-time of each port that is acting as LAS to the bridge's internal sense of DL-time.

The functions performed by the bridge to support DL-time propagation are:

- If the port is not acting as LAS, receive TD DLPDUs and adjust local DL-time in the receiving bridge port, regardless of the port's state.

NOTE This is part of the standard functions found in all Basic DLEs.

- If the receiving port is in the FORWARDING state, then use the port's DL-time as the bridge's internal sense of DL-time.

NOTE At most one port can be in the forwarding state and not acting as LAS.

- Adjust local DL-time in each of the bridge ports that are acting as LAS, based on the bridge's internal DL-time and time-rate-of-change.

- Create and transmit new TD DLPDUs on any port at which the bridge is acting as LAS, possibly because of a significant adjustment of the port's DL-time.

NOTE This is part of the standard functions found in all LM DLEs.

12.4.1.4.1.4 Access to higher-layer entities within the bridge.

Each Bridge Port also functions as an end station providing access to its local management agent (through System Management protocols), and possibly to other applications within the device.

12.4.1.4.2 Bridge Architecture.

The bridge consists of normal DL-functions for each port, plus specialized forwarding, republishing, time propagation and bridge coordination functions.

The normal low-level DL-functions of each port are responsible for receiving DLPDUs from the associated link addressed to that port. They are also responsible for sending DLPDUs queued for transmission at that port.

The filtering functions of the bridge use the configured filtering information to determine whether a received DLPDU should be discarded or should be forwarded to one or more ports other than the port on which it was received. They also determine to which ports a locally originated DLPDU should be sent.

The bridge coordination functions of 6.1.2 include the bridge management entity, which manages republishing DLCs and interacts with the bridge management entities of other bridges in the formation and maintenance of the bridge spanning tree. In non-adaptive bridges, only a minimal subset of the 12.4.1.4 spanning tree functionality is required, to permit non-adaptive bridges to coexist and interoperate with adaptive bridges conforming fully to 12.4.1.4.

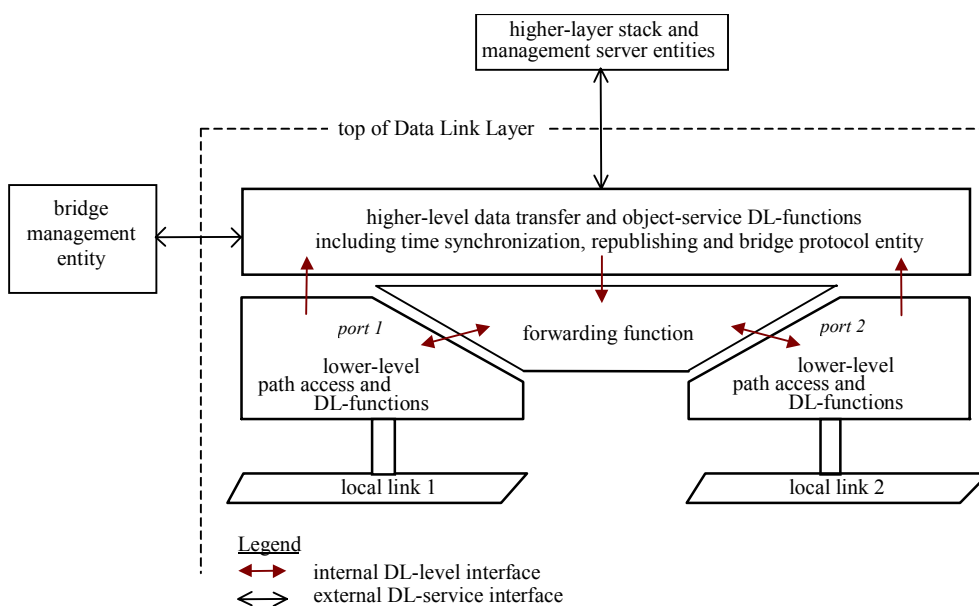


Figure 22 – Bridge architecture

12.4.1.4.2.1 Bridge management entity (BME)

The BME is responsible for

- subscribing to DLCs which are to be republished,
- initiating republishing DLCs and connecting subsequent subscribers to those republishing DLCs, and

- c) interacting with other bridges on the connected links in establishing the topology of the extended network and determining which non-disabled bridge ports should be forwarding and which should be blocking.

The BME initiates each subscribing DLCEP and one or more associated republishing DLCs, as configured in the republishing database. by

- 1) allocating a DL-buffer to be used to hold each DLSDU received at the subscribing DLCEP before it is to be republished,
- 2) subscribing to the DLC which is providing the information to be republished, and
- 3) initiating each publishing DLC which is to republish the subscribed-to information.

The timing of these actions is as follows:

- i) The DL-buffer is allocated before either the subscribing DLCEP or any of the republishing DLCs are established.
- ii) The subscribing DLCEP is established during bridge initialization or change of port state for any port which either is in or enters the Blocking or Forwarding state.

Any connection establishment request addressed to the publisher of a republishing DLC is delivered to the BME, which responds by connecting the would-be subscriber to the existing republishing DLC.

The BME also responds to any connection establishment or disconnection DLPDUs received at those DLCEPs, and to any change in the operational status of the relevant ports.

The actual act of receiving DLSDUs as a subscriber and republishing them as a publisher is handled as part of the standard connection-mode functions of the IEC Data Link, using a DL-buffer that is bound to the subscribing DLCEP and to each republishing DLCEP.

NOTE Reception and detection of a duplicate DLSDU at a subscribing DLCEP results in no update of the buffered DLSDU and no change in that buffered DLSDU's timeliness attribute.

The BME also generates DLPDUs for, and responds to DLPDUs received from, the BMEs of other bridges on the immediately connected links. In 12.4.1.4 the BME (therein called the Bridge Protocol Entity) manages the bridge spanning tree. In non-adaptive bridges, only a minimal subset of the 12.4.1.4 functionality is required, to permit non-adaptive bridges to coexist and interoperate with adaptive bridges conforming fully to 12.4.1.4.

12.4.1.4.2.2 Root port determination

A bridge which is not at the root of the bridge spanning tree has one port which most closely connects it to the root of the spanning tree. That port is known as the bridge's *root* port. In bridges conforming to 12.4.1.4, that port is determined automatically by the ISO/IEC Spanning Tree Protocol.

In non-adaptive bridges which do not implement the full spanning tree protocol, the bridge's root port, if any, is determined by the bridge's RootPort entry as configured by DL-management. For example, a non-zero value could identify that port of the bridge which is the root port; a zero value would indicate that the bridge is itself the root of the bridge spanning tree.

A bridge port which transitions to the FORWARDING state uses the RootPort information to determine whether it should also become LAS on its connected link; the port needs to become the LAS if and only if it is not the bridge's root port.

NOTE This requirement that a bridge which is not the root port become its local link's LAS minimizes the skew in DL-time among the set of connected bridges. The root port receives DL-time on its local link from that link's LAS; thus the root port must not itself act as LAS. The bridge at the root of the spanning tree acts as time source for the entire spanning tree; therefore none of its ports are designated a root port.

12.4.1.4.2.3 Port state information

State information associated with each bridge port governs its participation in the extended link. A port of a bridge is defined to be in one of three or more states set either by DL-Management or by the inter-bridge protocol of 12.4.1.4. The three basic states are:

- a) **DISABLED** — The port is incapable of DL-operation as a Bridge-class DLE, usually because the bridge portion of its database has not been configured. The port is always capable of operation as a Basic-class DLE, at least to the extent of being brought into active participation on its connected link and providing access to any Systems Management agent. The default state for each bridge port, before being configured by DL-Management, is DISABLED.

NOTE A disabled port may function as either a Basic-class or Link Master-class DLE, as specified in its standard NMIB for the port. The safest default states for a port are DISABLED and BASIC class.

- b) **BLOCKING** — The port is capable of DL-operation as a Bridge-class DLE, but is configured to function as a Link Master rather than a Bridge. It does not forward DLPDUs or republish DLSDUs, and does not participate in the Bridge Management or Bridge Annunciation protocol. It subscribes to DLSDUs that it would republish if it were in the FORWARDING state, so that it is prepared to transition to that state. If it is not acting as LAS on its attached link, then it synchronizes its local time sense to the attached link's, rather than to the bridge's internal time sense.

A port that is a redundant backup for a corresponding port of a parallel bridge would be in the BLOCKING state.

- c) **FORWARDING** — The port is operating as a Bridge-class DLE, forwarding DLPDUs and republishing DLSDUs as directed by its DL-management configuration., synchronizing its time sense to that of the root port of the bridge if it is not itself the root port, and participating in the Bridge Annunciation protocol whose purpose is to enable coexistence and interoperation with bridges using the Spanning Tree protocol of 12.4.1.4.

NOTE 1 DLPDUs addressed to any port of a bridge are addressed to the bridge itself. The port states of the receiving port and the addressed port have no affect in their local delivery.

Transitions between these states can be commanded by DL-management. Transitions between the BLOCKING and FORWARDING states also can be commanded by a proprietary bridge-internal redundancy manager which coordinates the port states of parallel ports of parallel bridges in a redundant bridge configuration.

NOTE 2 Proprietary bridge-internal redundancy managers are expected to coordinate only like bridges from a single vendor.

Concurrent with entering the FORWARDING state, a bridge port may have to assume the LAS role for the connected link. This assumption can be by extra-protocol means when it occurs between parallel bridges managed by a common bridge-internal redundancy management protocol. Otherwise this assumption uses the mechanism for transferring the LAS role described in clause 11, using the IEC DLP. A bridge port which is in, or entering, the BLOCKING or DISABLED state shall relinquish its LAS role upon request.

Change of bridge port state is not always the consequence of failure. When two bridges cooperatively exchange port states, with one transitioning from BLOCKING or LEARNING to FORWARDING, and the other from FORWARDING to LEARNING or BLOCKING, the bridges should transition their ports into the new state synchronously with their release of, or acceptance of, the LAS role at that port. This change of port state will affect future filtering decisions on DLPDUs received at or for that port, and future republishing actions. However, it is recommended that a port which has just transitioned to the LEARNING or BLOCKING state continue to transmit already-queued DLPDUs from its forwarding queues.

12.4.1.4.2.4 Filtering database.

A bridge filters DLPDUs by determining to which ports, if any, a received DLPDU should be forwarded. Filtering is used to confine DLPDUs transmitted between end stations to the subset of the extended link that forms a DL-path between those end stations when such a path is known.

NOTE The term filtering is used in this sense as a permissive filter, because that is the sense of the information configured in the filtering database. The more common language sense of filtering for discard applies to the set of unselected ports, which is the inverse of the database information.

Each bridge contains a Filtering Database. Conceptually, the Filtering Database consists of two lists of records, where the first specifies default forwarding for links:

individual DL-addresses

- a) a DL-link designator in the range 1000 to V(ML), the maximum link designator value;
- b) identification of a set of ports to which a DLPDU addressed to this link should be forwarded when all other enabling conditions are met.

The second list specifies forwarding for individual DL-addresses

- c) a DLSAP-address, DLCEP-address or group DL-address;
- d) identification of a set of ports to which DLPDUs with this destination DL-address should be forwarded when all other enabling conditions are met;
- e) identification of a set of ports disjoint from those specified in d) to which DLPDUs containing just a source address (that is, publishing DLPDUs) should be forwarded when all other enabling conditions are met.

NOTE This last set of ports is of use only for publisher DLCEP-addresses.

12.4.1.4.2.5 Republishing database.

A bridge republishes DLSDUs by creating a DL-buffer, subscribing to a DLC on one port, establishing publishing DLCs on one or more other ports, and binding the receive DL-buffer for the subscribing DLCEP as the send DL-buffer for the republishing DLCEP(s).

Each bridge contains a Republishing Database. Conceptually, the Republishing Database consists of a list of records, each specifying

- a) the 32-bit DLCEP-address of the DLC to which the bridge should subscribe for the purpose of republishing;
- b) the characteristics of that DLC (maximum DLSDU size, priority, data delivery features, authentication, etc.);
- c) the 32-bit DLCEP address(es) of the DLC (s) which the bridge should initiate as republisher.

12.4.1.4.3 Addressing

12.4.1.4.3.1 Link designators.

Each of the constituent links, which together comprise the extended link, has its own unique link designator. The forwarding database of each bridge has an entry for each of these links, specifying the bridge port through which (direct or indirect) access to the link is obtained.

12.4.1.4.4 Statistics and diagnostic information.

For each receiving port, statistics should be kept on the number of received DLPDUs

- a) whose first delocalized DL-address specifies a link designator that is not in the Forwarding Database, or
- b) that do not require forwarding, based on information in the Forwarding Database, or
- c) that are queued for forwarding, frequently after being placed in a forwarding buffer, or
- d) that are discarded due to lack of forwarding buffers.

NOTE Each received DLPDU will fall into exactly one of these categories.

For each transmitting port, statistics should be kept on the number of DLPDUs that were queued for forwarding on that port that have been discarded due to excessive forwarding delay.

The header of the last DLPDU whose first DL-address specifies a link designator that is not in the bridge's Forwarding Database should be retained for diagnostic use.

12.4.1.5 Detailed conceptual model of bridge functions (informative)

The forwarding functions of a bridge are a specialized extension of the address-based delivery functions intrinsic to all DLEs. The following conceptual models of processes, entities and data structures, which apply to the non-adaptive bridging subset of IEC DLP, facilitate an understanding of this process. Alternative conceptualizations are possible.

12.4.1.5.1 DLPDU reception.

The lower-level DLL process associated with each bridge port examines all DLPDUs received from the link to which it is attached. DLPDUs that are detected to be in error are discarded; this includes DLPDUs whose FCS is in error.

Received DC, DT and EC DLPDUs with LONG DL-addresses whose link designators are all non-zero are submitted to the forwarding process. Other DLPDUs may be submitted to the forwarding process or not as permitted by the relevant DLPDU-specific subclause of the form 8.nn.4.3.

DLPDUs addressed to the bridge as an end station are submitted to the upper-level-DLL for processing. These may result in delivery of a DLSDU to one or more DLS-user entities at one or more DLSAPs.

12.4.1.5.2 Model of reception and forwarding of a DLPDU.

A lower-level port-specific process receives a DLPDU, validates its FCS, and classifies the DLPDU based on its frame control octet. The receive process creates a received-DLPDU record consisting of:

- a) the DLPDU class;
- b) the delocalized DL-address(es) explicitly or implicitly specified by the DLPDU's header;
- c) the octets representing the *body* of the DLPDU — the DL-parameters, or DLS-user data, or both;
- d) the port number of the port from which the DLPDU was received; and
- e) the DLPDU's lower-level header and trailer octets (that is, received frame control, address and FCS octets).

NOTE Components a) through c) are identical for, and apply to, both bridge and non-bridge DLEs.

The DLPDU class and first delocalized address are used to determine the other DL-processes, if any, which should receive the DLPDU before the DLPDU record is deleted.

- 1) If the first delocalized DL-address is an active DLSAP-address of the DLE, the DLPDU record is passed to the upper-level process that handles DLPDUs received at that DLSAP.
- 2) If the first delocalized DL-address is an active group DL-address of the DLE, for each local DLSAP which is a member of the group, the DLPDU record is passed to the upper-level process that handles DLPDUs received at that DLSAP.
- 3) If the first delocalized DL-address is an active DLCEP-address of the DLE, for each local DLCEP which is associated with that DLCEP-address, the DLPDU record is passed to the upper-level process associated with that DLCEP as follows:

- i) If the DLE has a publisher DLCEP for that DLCEP-address, and the DLPDU class is one which should be forwarded to publishers, then the DLPDU record is passed to the process associated with that publisher DLCEP.
 - ii) If the DLE has one or more subscriber DLCEPs for that DLCEP-address, and the DLPDU class is one that should be forwarded to subscribers, then for each of those subscriber DLCEPs, the DLPDU record is passed to the process associated with that subscriber DLCEP.
- 4) If
- i) the received DLPDU contained one or more explicit DL-addresses, and
 - ii) all of those addresses were LONG with non-zero link designators, and
 - iii) the link-designator of the first delocalized DL-address is in the forwarding database of the DLE,

then the DLPDU record is passed to the lower-level transmission process of each port associated with that link designator in the forwarding database that is in the FORWARDING state, except the port on which the DLPDU was received.

Lower-level port-specific transmission processes retransmit forwarded DLPDUs on their associated links. The actual octets of the received DLPDU, which were retained in fields c) and e) of the received DLPDU record, are used to retransmit the DLPDU. If the FINAL bit of the received frame control octet needs to be complemented before transmission, then the received FCS is modified as specified in 5.1.3 before that FCS is transmitted. A new FCS is never calculated for a forwarded DLPDU.

Receive statistics are kept as specified in 12.5.6.3.

NOTE Alternatives 1) through 3) are identical for, and apply to, both bridge and non-bridge DLEs. However, the set of DL-addresses recognized by bridge DLEs as relating to the bridge itself is generally larger than the similar set for non-bridge DLEs. Each bridge port is required to recognize the DL-addresses of all of that bridge's ports, regardless of the port states of those ports. Without such recognition, it would not be possible for System Management to configure a bridge port other than the one on which the DLPDU was received.

12.4.1.5.3 Model of local origination of a DLPDU.

A higher-level process creates a locally originated DLPDU record consisting of

- a) the DLPDU class;
- b) the delocalized DL-address(es) which are to be specified, explicitly or implicitly, in localized or delocalized form, by the DLPDU's header;
- c) the octets representing the *body* of the DLPDU — the DL-parameters, or DLS-user data, or both.

The DLPDU class and first delocalized DL-address are used to determine the other DL-processes that should receive the DLPDU before the DLPDU record is deleted.

- 1) If the first delocalized DL-address is an active DLSAP-address of the DLE, the DLPDU record is passed to the upper-level process that handles DLPDUs received at that DLSAP.
- 2) If the first delocalized DL-address is an active group DL-address of the DLE, for each local DLSAP which is a member of the group and which is not the sending DLSAP, the DLPDU record is passed to the upper-level process which handles DLPDUs received at that DLSAP.
- 3) If the first delocalized DL-address is an active DLCEP-address of the DLE, for each local DLCEP which is associated with that DLCEP-address and which is not the sending DLCEP, the DLPDU record is passed to the upper-level process associated with that DLCEP as follows:
 - a) If the DLE has a publisher DLCEP for that DLCEP-address, and the DLPDU class is one which should be forwarded to publishers, then the DLPDU record is passed to the process associated with that publisher DLCEP.

- b) If the DLE has one or more subscriber DLCEPs for that DLCEP-address, and the DLPDU class is one which should be forwarded to subscribers, then for each of those subscriber DLCEPs, the DLPDU record is passed to the process associated with that subscriber DLCEP.
- 4) If the link-designator of the first delocalized DL-address is in the forwarding database of the DLE, then the DLPDU record is passed to the lower-level transmission process of each port associated with that link designator in the forwarding database.

Lower-level port-specific transmission processes transmit locally originated DLPDUs on their associated links by

- building a link-appropriate header from fields 1.a) and 1.b), possibly localizing and omitting all or part of the DL-addresses from the DLPDU in the process,
- appending the body from field 1.c), and
- computing the FCS over the DLPDU header and body as transmitted.

NOTE Alternatives 1) through 3) are identical for, and apply to, both bridge and non-bridge DLEs. Alternative 4) can always occur in a non-bridge DLE; the link designator can always be presumed to be in the implicit "forwarding database."

12.4.1.5.4 Model of subscribing to one DLCEP and republishing on one or more other DLCEPs.

Each Bridge may function as a republisher of published data. The entities that model this operation are:

- a) the subscriber DLCEP, responsible for receiving published DLPDUs, detecting duplicate transmissions of sequence-numbered DLPDUs, and buffering the contained DLSDU from a non-duplicated transmission in a DL-buffer, together with the DLSDU's timeliness status.
- b) the DL-buffer that holds the last-DLSDU received as a subscriber, together with its timeliness status, and makes it available as the source DLSDU for republishing.
- c) the publisher DLCEP, responsible for transmitting the DLSDU and its timeliness status on its new DLC, updating any sequence number associated with the transmission only when the buffer has been updated since the prior transmission on the DLCEP.

12.4.1.5.5 DLPDU transmission.

The lower-level DLL process associated with each bridge port transmits DLPDUs on that port in accordance with the associated link's schedule.

DLPDUs which were received from another port and forwarded are retransmitted without alteration, except perhaps for complementation of the FINAL bit in the DLPDU's first octet (the frame control octet) and corresponding complementation of specific bits in the forwarded DLPDU's FCS, where the FCS complementation pattern is a function only of the number of octets in the DLPDU. (See 5.1.3.)

DLPDUs that are locally originated within the DLE are submitted by the upper-level DLL entity directly to the forwarding process for transmission on the port(s) associated with the first address of the DLPDU. DLPDUs that address the sending bridge as an end-station are also delivered to the upper-level DLL entity for processing.

12.4.2 Adaptive bridge sub-protocol and elements of procedure

The ISO/IEC bridge protocol, ISO/IEC 10038, specifies management and operation of the MAC-relay entities of a bridged local area network. This subclause specifies additions and modifications to ISO/IEC 10038:1993 which transform it into a bridge protocol suitable for use with fieldbus DL-relay entities.

In the following text references to clauses of this part of the fieldbus DL protocol specification are preceded by [FB], references to clauses of ISO/IEC 10038:1993 are preceded by [IL].

12.4.2.1 Global changes

Throughout [IL] replace all references to

- a) “MAC” by “DL” or “DL-”, as appropriate;
- b) “Media Access Control” by “Data Link”;
- c) “Sublayer” by “layer”;
- d) “LAN” by “link”;
- e) “Bridged Local Area Network” by “extended link”;
- f) “user priority” with “DLL-priority”;
- g) “frame” with “DLPDU”;
- h) “LLC” with “upper DLL”;
- i) “section” with “clause”.

Throughout [IL] change numeric references to subclauses of [IL] by preceding the reference, if appropriate, with the subclause identification “12.4.2.” to provide a means of discriminating between cross-references within this subclause [IL] and references within the subclause to IEC 61158-4 [FB].

Make all other style, format and nomenclature changes required by IEC Directives, and renumber or reletter the items within each subclause as appropriate after the following changes.

12.4.2.2 Changes to [IL/1]: Introduction

Modify the following indicated paragraphs of [IL/1.5] as shown:

- (1) delete paragraph (b)
- (4) delete this paragraph
- (7) replace “may” with “shall”
- (9) replace paragraph with “shall use 32-bit DL-addresses and their 48-bit MAC-address representations as specified in [IL/3.1.2] and described in [FB/6.3.4].
- (11) replace “may” with “shall” and delete “optional”
- (12) delete paragraphs (a) and (b)
- (16) replace “may” with “shall”
- (17) replace “may” with “shall”
- (19) replace “may” with “shall”
- (20) replace paragraph with “shall support remote management, as specified in [IL/7] or by alternate means”

Delete [IL/1.6 and 1.7].

12.4.2.3 Changes to [IL/2]: Support of the DL-Service

Replace the first paragraph of [IL/2.1] with “The DL-service provided to end stations attached to an extended link is supported by the bridges in that extended link. Bridges support all of the DL-services which involve multi-end-station communication or scheduling.”

Delete “Destination” from [IL/2.21)].

Replace paragraph [IL/2.23)] with “the DL-addresses of end-stations are not restricted by the topology and configuration of the extended link, except as specified in [FB/6.3.2.1] for link-designator values between 1000₁₆ and FFFF₁₆, inclusive.”

Delete paragraph [IL/2.3.22c)].

Replace paragraph [IL/2.3.3] with “The operation of Bridges is unlikely to disorder DLPDUs transmitted with the same DL-priority. Such misordering can occur

- a) under conditions where DL-redundancy is employed to provide multiple simultaneously-usable paths between the two end-stations, or
- b) immediately subsequent to reconfiguration of the DL-path between the two end stations.

NOTE Such reconfiguration can occur only as the result of DL-management action or during recovery from bridge or Ph-layer entity failures.

Replace the second paragraph of [IL/2.3.7] with “For DLPDUs relayed between links, the bridge shall modify the FC octet of each DLPDU and make compensatory modifications to the FCS of that DLPDU as specified in [FB/5.1.3].”

Insert after [IL/2.3.10] “2.3.11 Time synchronization. The bridges establish and preserve the shared sense of DL-time throughout the extended link.”

Replace [IL/2.4] with “2.4 Forwarding of DLPDUs.

“A bridge receiving a DLPDU examines the DLPDU’s frame control octet and first address, from which the bridge determines whether and to where the DLPDU should be forwarded.

“A bridge forwards a CA, CD, DC, DT, EC, ED or RC DLPDU, together with its received FCS, toward those links which have DLEs which should receive the DLPDU’s first DL-address. No other forwarding of DLPDUs occurs.

“If the received DLPDU is a CA, CD or ED DLPDU, then the bridge replies immediately with a SR DLPDU on the link from which the CA, CD or ED DLPDU was received.”

Delete [IL/2.5].

12.4.2.4 Changes to [IL/3]: Principles of operation

Insert before [IL/3.1 (2)]:

- “(2) Relay of the shared sense of DL-time.”

Modify the indicated paragraphs of [IL/3.1.1] as shown:

- (3) *Replace with “DLPDU discard if the DLPDU type is not CA, CD, DC, DT, EC, ED, RC or SR; and DLPDU discard if the DLPDU is a detected duplicate of a prior CA, CD or ED DLPDU, presumably caused by loss of a prior immediate retry (see 8.4.4, 8.5.4 and 8.6.4).”*

- (3+) *Insert before (4)* “Possible DLPDU transformation or discard if the DLPDU type is DT or SR (see 8.7.4.3c) and 8.8.4.3b)).”
- (5) Delete this paragraph.
- (8) Delete this paragraph.
- (9) *Replace with* “modification of a frame control octet, with compensatory modification of the DLPDU’s FCS.”
- (9+) *Insert before (10)* “FCS calculation on transformed DT or SR DLPDUs (both of which become DT DLPDUs).”

Append to [IL/3.1.2 (3)] “... and queries to other bridges and end-stations on the local link.”

Replace [IL/3.2] with “**3.2 Bridge architecture.** Each bridge port receives and transmits DLPDUs to and from the link to which it is attached. The DL-relay-entity handles the functions of relaying DLPDUs between bridge ports, filtering DLPDUs, and learning filtering information.

“The bridge protocol entity handles calculation and configuration of the extended link topology. The bridge protocol entity and other higher-layer protocol users, such as DL-management, make use of higher-level DLL services, which are provided in a port-independent manner.

“Figures 3-2 and 3-3 of [IL] illustrate a bridge and its ports, and the architecture of the bridge for a bridge with two ports. A bridge may have more than two ports. Where Fractional Duty Cycle (FDC) nodes exist on a link, it is also meaningful for a bridge to have just a single port, since a one-port bridge can supply the DLPDU store-and-forward services needed to permit communication with or among FDC nodes.”

Delete the first paragraph of [IL/3.3].

Replace figures 3-2 and 3-3 of [IL] with Figure 23 and Figure 24 to replace the term “LAN” with “local link” and to delete the terms “LLC entities” and “MAC service interfaces”:

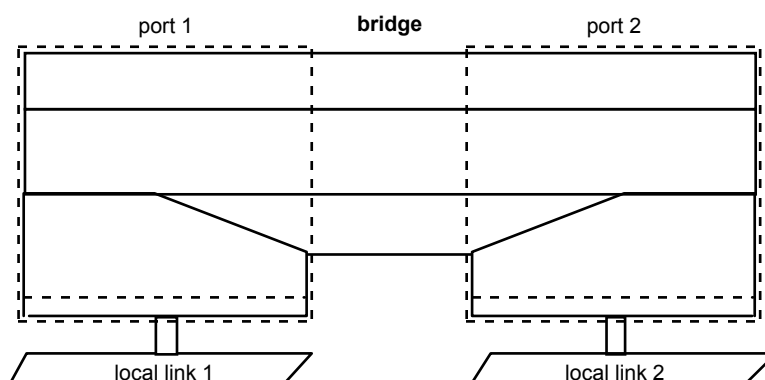


Figure 23 – Replacement for [IL] Fig 3-2 Bridge ports

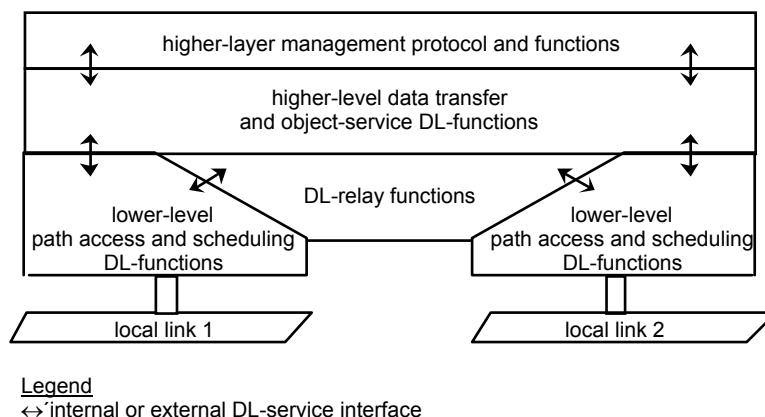


Figure 24 – Replacement for [IL] Fig 3-3 Bridge architecture

Delete the paragraph of [IL/3.3] which begins “Each Bridge Port shall support ...”.

Replace the title of figure 3.4 of [IL] with “Relaying DLPDUs”.

Replace the title of figure 3.5 of [IL] with “Observation of link traffic”.

Delete the dotted blocks “LLC” from figure 3-6 of [IL].

Replace [IL/3.5] with “**3.5 DLPDU reception.** The lower-level DLL entity associated with each bridge port examines all DLPDUs received from the link to which it is attached. DLPDUs that are detected to be in error are discarded; this includes DLPDUs whose FCS is in error. DLPDUs that are detected to be duplicates of prior received DLPDUs, due to immediate retry by the current holder of a scheduled or delegated token, are also discarded after generation of the required immediate reply. All other DLPDUs shall be submitted to the learning process.

“Received DC, EC and RC DLPDUs [IL/2.4] shall be submitted to the forwarding process. Received CA, CD and ED DLPDUs [IL/2.4] shall be submitted to the forwarding process except when they are discarded as duplicates of prior received DLPDUs. Received DT DLPDUs [IL/2.4] containing at least one explicit DL-address shall be submitted to the forwarding process.

“Other DLPDU types shall not be relayed by the bridge. However, the bridge may be required to transform a received SR DLPDU, or a received DT DLPDU without explicit DL-addresses and with null user data, to a DT DLPDU with one explicit DL-address and null user data, and forward that transformed DLPDU as if it had just been received (see 8.7.4.3c) and 8.8.4.3b)).

“DLPDUs addressed to the bridge as an end station shall be submitted to upper-DLL processing except when they are discarded as duplicates of prior received DLPDUs.”

“Locally-originated DLPDUs shall be submitted directly to the forwarding process by the upper-DLL functions.”

Replace [IL/3.6] with “**3.6 DLPDU transmission.** The lower-level DLL entity associated with each bridge port transmits DLPDUs submitted to it by higher DLL functions. Relayed, transformed and locally originated DLPDUs are submitted for transmission by the forwarding process.”

Replace [IL/3.7.1 (3) (a)] with “The DLPDU specifies a destination DL-address and the Filtering Database indicates that DLPDUs with this value of the destination address should be forwarded through the transmission Port (as would happen, for example, if the destination address specified a local link accessible through that transmission Port).”

Replace [IL/3.7.1 (3) (b)] with “The DLPDU specifies only a source DL-address and the Filtering Database indicates that source-only DLPDUs with this value of the source address should be forwarded through the transmission Port (as would happen, for example, if expected receivers of DLPDUs with that source address were attached to a local link accessible through that transmission Port).”

Delete [IL/3.7.2].

Delete [IL/3.7.4].

Replace [IL/3.7.5] with “3.7.5 Modification of DLPDUs. When forwarding a DLPDU, it is sometimes necessary for a bridge to alter the final-use subfield of a DLPDU’s frame control field. Specifically, the bridge may have to complement the value of the final-use subfield of the frame control field. This modification, with a corresponding FCS modification, shall be performed as specified in [FB/5.1.3].”

Delete tables 3-1 and 3-2 of [IL].

*Replace the first paragraph of [IL/3.8] with “**3.8 The learning process.** The Learning Process observes the source DL-addresses, and in some cases the destination DL-addresses and CE-parameters, of DLPDUs received on each Port and updates the Filtering Database conditionally on the state of the receiving Port.*

Replace the third paragraph of [IL/3.8] and its subparagraphs with “The Learning Process may deduce the path through the extended link to particular end stations, by inspection of the source DL-address fields of CA, CD, DC, DT, ED and RC DLPDUs, and by inspection of the EC-parameters and all of the DL-address fields of received EC DLPDUs.

- “(1) For the source DL-address of a CA, CD, DC, DT, ED or RC DLPDU, and for each of the two source DL-addresses of an EC DLPDU, if
 - “(a) the link-designator component of that DL-address is in the range 1000_{16} to $V(ML)$, inclusive, where $V(ML)$ is as defined in [FB/6.7.6.1];
 - “(b) the Port from which the DLPDU was received was in a state that allows learning [IL/4.4];
 - “(c) a static entry [IL/3.9, 3.9.1] for the link-designator component of that source DL-address does not already exist; and
 - “(d) the resulting number of entries would not exceed the capacity of the Filtering Database;

“then the Learning Process shall create or update a dynamic entry [IL/3.9, 3.9.2] in the Filtering Database for the link-designator component of that source DL-address, associating the Port on which the DLPDU was received with that link-designator component.
- “(2) For the source DL-address of a CA, CD, DC, DT, ED or RC DLPDU, and for each of the two source DL-addresses of an EC DLPDU, if
 - “(a) the link-designator component [FB/6.3.1] of the source DL-address is in the range 0080_{16} to $0FFF_{16}$, inclusive;
 - “(b) the Port from which the DLPDU was received was in a state that allows learning [IL/4.4];

“(c) a static entry [IL/3.9, 3.9.1] for that source DL-address does not already exist; and

“(d) the resulting number of entries would not exceed the capacity of the Filtering Database;

“then the Learning Process shall create or update a dynamic entry [IL/3.9, 3.9.2] in the Filtering Database for that source DL-address, associating the Port on which the DLPDU was received with the DL-address.

“(3) For an EC DLPDU, if

“(a) the sender’s-DLCEP-class parameter of the DLPDU’s EC-parameters specifies PUBLISHER [FB/9.1a)4)];

“(b) the Port from which the DLPDU was received was in a state that allows learning [IL/4.4];

“(c) a static entry [IL/3.9, 3.9.1] for the DLPDU’s first source DL-address does not already exist; and

“(d) the resulting number of entries would not exceed the capacity of the Filtering Database;

“then the Learning Process shall create or update a dynamic Publisher entry [IL/3.9, 3.9.2] in the Filtering Database for that source DL-address, associating the Ports to which the DLPDU will be forwarded to the subscriber ports of that DL-address.

“(4) For an EC DLPDU, if

“(a) the sender’s-DLCEP-class parameter of the DLPDU’s EC-parameters specifies SUBSCRIBER [FB/9.1a)4)];

“(b) the Port from which the DLPDU was received was in a state that allows learning [IL/4.4];

“(c) a static entry [IL/3.9, 3.9.1] for the DLPDU’s destination DL-address does not already exist; and

“(d) the resulting number of entries would not exceed the capacity of the Filtering Database;

“then the Learning Process shall create or update a dynamic Publisher entry [IL/3.9, 3.9.2] in the Filtering Database for that destination DL-address, associating the Port on which the DLPDU was received as one of the subscriber ports of that DL-address.

“The Learning Process may also query other DLEs on the local link by sending DL-address list query SPDUs [FB/Clause 11 to the DL-support functions of those DLEs and using the resultant DL-address list reply SPDUs [FB/clause 11] as a basis for adding or removing entries from the Filtering Database.”

Replace the second to last paragraph of [IL/3.8] with “If the Filtering Database is already filled up to its capacity, but a new entry would otherwise be made, then DL-management shall be informed of the Filtering Database overflow, and the new entry shall not be made.”

Replace the last sentence of the first paragraph of [IL/3.9] with “It supports queries by the Forwarding Process as to whether DLPDUs with given values of the destination DL-address, or of the source DL-address in DLPDUs specifying only an explicit source DL-address, should be forwarded to a given Port.”

Replace the second paragraph of [IL/3.9] with “The Filtering Database shall be capable of containing both static entries [IL/3.9.1] and dynamic entries [IL/3.9.2], for both full DL-addresses and DL-address link-designator components [FB/6.3.1]).

“Static and dynamic entries shall not both exist for the same full DL-address; a dynamic entry shall not be created if a corresponding static entry for the same full DL-address already exists; and creation of a static entry shall cause the removal of a corresponding dynamic entry for the same full DL-address if one exists.

“Static and dynamic entries shall not both exist for the same DL-address link-designator component; a dynamic entry shall not be created if a corresponding static entry for the same partial DL-address already exists; and creation of a static entry shall cause the removal of a corresponding dynamic entry for the same partial DL-address if one exists.

“Each entry in the Filtering Database shall be of one of five classes:

- “(a) A DL-address link-designator component entry, which specifies the default single port to which a DLPDU with a destination DL-address containing this link-designator component should be forwarded when the Filtering Database does not contain a specific entry for the destination DL-address specified in the DLPDU.
- “(b) A group DL-address entry, which specifies, separately for each receiving port, the set of ports to which a DLPDU whose destination DL-address equals the specified DL-address should be forwarded.
- “(c) A DLSAP-address entry, which specifies the single port to which a DLPDU whose destination DL-address equals the specified DL-address should be forwarded.
- “(d) A peer DLCEP-address entry, which specifies the single port to which a DLPDU whose destination DL-address equals the specified DL-address should be forwarded.
- “(e) A publisher DLCEP-address entry, which specifies
 - “(i) the single port to which a CA, CD, DT, EC or ED DLPDU whose destination DL-address equals the specified DL-address should be forwarded; and
 - “(ii) the set of ports, all different from (i), to which a DC, DT, EC or RC DLPDU which contains no destination DL-address, and which specifies this DL-address as an explicit source DL-address, should be forwarded.”

Replace the second and following paragraphs of [IL/3.9.1] with “A static entry may specify either a full DL-address or a link-designator component of a DL-address.”

Replace the last five paragraphs of [IL/3.9.2], beginning “Dynamic entries specify ...”, with “A dynamic entry may specify any full DL-address or DL-address link-designator component.”

*Replace all of [IL/3.12] with “**3.12 Addressing.**” That is, delete all text following the title up to but not including [IL/3.12.1].*

In the second paragraph of [IL/3.12.3], replace “LLC Entities” with “DLEs”, and replace “Standard LSAP ... users of LLC” with “DL-bridge functions of DLEs (see 6.3).”

Replace the third paragraph of [IL/3.12.3] with “Each DL-UNITDATA request primitive causes the transmission of a DLPDU which conveys the BPDU in its user data field as a complete DLSDU.”

Delete the fourth paragraph of [IL/3.12.3].

In the fifth paragraph of [IL/3.12.3], delete “, within the scope of the LSAP Assignment”.

Delete the last sentence of the seventh paragraph of [IL/3.12.3], which states “This group address ... are transmitted.”

Delete the eighth paragraph of [IL/3.12.3].

Replace the last paragraph of [IL/3.12.4] with “[FB/6.3] specifies standard DL-addresses for the DL-bridge functions of bridge DLEs.”

Delete the second and third paragraphs of [IL/3.12.5].

Delete [IL/3.12.6] and tables 3-4, 3-5, 3-6 and 3-7 of [IL].

12.4.2.5 Changes to [IL/ 4]: The spanning tree algorithm and protocol

In the second paragraph of [IL/4.3.4], change “must” to “should”.

In tables 4-2 and 4-3, replace the first line after each table with “All times are in seconds when the local link data rate is 1 Mbit/s or greater; otherwise all times are in minutes.”

In table 4-3, replace the line

Bridge Forward Delay	15,0	—	4,0 – 30,0

with

Bridge Forward Delay	0,0	—	0,0 – 30,0

*Insert before [IL/4.6.11] “**4.6.10.3.5** The port’s DLE shall attempt to become the LAS for the local link to which it is attached.”*

12.4.2.6 Changes to [IL/5]: Encoding of bridge protocol data units

Replace the third paragraph of [IL/5.2.5] with “The third most significant octet is assigned the value of the initial octet of the 48-bit Bridge Address. The fourth to eighth octets are similarly assigned the values of the second to the sixth octets of the 48-bit Bridge Address, respectively.”

Delete the fourth paragraph of [IL/5.2.5].

12.4.2.7 Changes to clause 6: Bridge management

Replace [IL/6.35] with “(5) IA5 String, for all text strings.”

Replace [IL/6.7.5.1.2 (2)] with “(2) Address — full DL-address or DL-address link-designator component of the entry.”

Replace [IL/6.7.5.1.2 (3)] with “(3) Entry class; associated port or port maps:

- “(a) DL-address link-designator component — the default single port to which a DLPDU with a destination DL-address containing this link-designator component should be forwarded when the Filtering Database does not contain a specific entry for the destination DL-address specified in the DLPDU
- “(b) group-DL-address — an array which specifies, separately for each receiving port, the set of ports to which a DLPDU specifying this DL-address as a destination DL-address should be forwarded.
- “(c) DLSAP-address — the single port to which a DLPDU specifying this DL-address as a destination DL-address should be forwarded.
- “(d) peer DLCEP-address — the single port to which a DLPDU specifying this DL-address as a destination DL-address should be forwarded.
- “(e) publisher-DLCEP-address —
 - “(i) the single port to which a DLPDU specifying this DL-address as a destination DL-address should be forwarded; and
 - “(ii) the set of ports, all different from (i), to which a source-only DC, DT, EC or RC DLPDU specifying this DL-address as an explicit source DL-address should be forwarded.”

Replace [IL/6.7.5.2.2 (2)] with “(2) Address — full DL-address or DL-address link-designator component of the desired entry.”

Replace [IL/6.7.5.3.2 (2)] with “(2) Address — full DL-address or DL-address link-designator component of the desired entry.”

Replace [IL/6.7.5.3.3 (1)] with “(1) Address — full DL-address or DL-address link-designator component of the desired entry.”

Replace [IL/6.7.5.3.3 (3)] with “(3) Entry class; associated port or port maps; as specified in [IL6.7.5.1.2 (3)].”

Replace [IL/6.7.5.4.3 (3) (a)] with “(a) Address — full DL-address or DL-address link-designator component of the desired entry.”

Replace [IL/6.7.5.4.3 (3) (c)] with “(c) Entry class; associated port or port maps; as specified in [IL/6.7.5.1.2 (3)].”

12.4.2.8 Changes to [IL/7]: Management protocol

Replace the entire clause after the title with:

“The fieldbus management protocol is outside the scope of the IEC 61158 set of standards. Some DL-management capabilities are specified in 11.3 and 12.4.3.1.”

12.4.2.9 Changes to [IL/annex A]: PICS proforma

Delete items 1a, 1b and 2a.

Reword item 2c to read “Are CA, CD, DC, DT, EC, ED, RC and SR DLPDUs the only types of DLPDU relayed, possibly after transformation as specified in 8.7.4.3c) and 8.8.4.3b)?”

Delete items 2d, 2f, 2g, 2n, 2o, 2q, 2r, 2s, 4, 4a, 4b, and 5b.

Reword item 5g to read “Does each static or dynamic entry specify either a full DL-address or the link-designator component of a DL-address?”

Reword item 5i to read “Does each entry which specifies a link-designator component of a DL-address specify a port number?”

Renumber old item 5j to 5p.

Add new item 5j “Does each entry which specifies a group-DL-address specify an array of sets of ports, with one set for each receiving port?”

Add new item 5k “Does each entry which specifies a DLSAP-address specify a port number?”

Add new item 5m “Does each entry which specifies a peer DLCEP-address specify a port number?”

Add item 5n “Does each entry which specifies a publisher-DLCEP-address specify

“a) a port number; and

“b) separately, a set of ports?”

Reword item 7d to read “Can static entries be made for link-designators in the range 1000₁₆ to FFFF₁₆?”

Replace item 7f with “Can static entries be made for DLSAP- addresses?”

Renumber old item 7g to 7j.

Add new item 7g “Can static entries be made for peer DLCEP-addresses?”

Add new item 7h “Can static entries be made for publisher DLCEP-addresses?”

Delete items 9a, 9b, 9c, 10b, 10e, 10f, 11a, 11b, 11d, 11e, 11f, and 12a.

Replace item 10c with “Are all BPDUs addressed to the DL-bridge functions of bridge DLEs?”

12.4.2.10 Delete [IL/annex C] and [IL/annex D], both of which relate to source-routing

12.4.3 Non-adaptive bridge sub-protocol and elements of procedure

Non-adaptive bridges shall implement the subset of bridge functionality described in 12.4.1. The means of configuring the required forwarding and republishing databases is a local issue.

12.4.3.1 Bridge annunciation protocol and BPDU structure.

The bridge management entity is responsible for implementing the Bridge Annunciation protocol, which is a minimal compatible subset of the Spanning Tree protocol of 12.4.1.4. This subset serves to announce the non-adaptive bridge's presence to adaptive bridges conforming to 12.4.1.4 and to cause the latter bridges to defer to non-adaptive bridges in establishing the acyclic graph which is the spanning tree.

The bridge management entity achieves this by transmitting BPDUs (Bridge Protocol Data Units) as the DLS-user data of connectionless DLPDUs, addressed to the standard group DL-address for bridges on the local link (0x0103) as specified in Table 7, with the same

format as in 12.4.1.4. The source DL-address of the DLPDUs is formed from the standard selector 0x01 for the bridge functions of a DLE, as specified in Table 8.

- a) It sends a Topology Change Notification BPDU at startup from each bridge port that is configured to be in the FORWARDING state, and similarly after reconfiguration, from each bridge port that has just been placed in the FORWARDING state. Table 131 shows the coding of a Topology Change Notification BPDU.

Table 131 – Topology change notification BPDU format

Octets	1	2	3	4
01-04	0x00	0x00	0x00	0x80

- b) It sends a Configuration BPDU on a port each time it receives a Topology Change Notification BPDU on that port, and each time it receives a Configuration BPDU on that port with a non-zero value in any of octets 6 through 13. Table 132 shows the coding of a Configuration BPDU, with non-zero values in octets 27 – 35, and with octet 27 specifying the port number of the bridge port from which the Configuration BPDU is being transmitted.

Table 132 – Configuration BPDU format

Octets	4N+1	4N+2	4N+3	4N+4
01-04	0x00	0x00	0x00	0x00
05-08	0x00	0x00	0x00	0x00
09-12	0x00	0x00	0x00	0x00
13-16	0x00	0x00	0x00	0x00
17-20	0x00	0x00	0x00	0x00
21-24	0x00	0x00	0x00	0x00
25-28	0x00	0x00	port #	0xFF
29-32	0xFF	0xFF	0xFF	0xFF
33-35	0xFF	0xFF	0xFF	

12.5 DL-management-information

12.5.1 Scope

This subclause enumerates the set of DL-parameters, defined as variables in 6.7, which need to be preconfigured before proper DLE operation is possible. It also summarizes DLE actions which can be requested through DL-management.

12.5.2 DLE configuration parameters

The following parameters, defined as variables in 6.7, are required to support the three classes of DLE operation.

12.5.2.1 Node-specific DL-configuration parameters

The following parameters shall be assigned the appropriate values for the DLE before the DLE is permitted to transmit any DLPDU on the local link:

12.5.3 Additional node-independent DL-configuration parameters

The following DL-parameters shall be assigned the then-current values for the local link before the DLE is permitted to transmit any DLPDU on the local link:

NOTE Each PN DLPDU conveys the values of the following parameters, plus those of critical PhL parameters. Thus a DLE which receives a PN DLPDU is able to configure itself, and its associated PhE, so that the DLE and associated PhE can reply to that or a subsequent PN DLPDU, starting the process of integrating the replying DLE into an already-functioning network.

12.5.3.1 V(MRD) maximum-response-delay, defined in 6.7.1.3;

The following DL-parameters shall be assigned the then-current values for the local link before the DLE is permitted to transmit any DLPDU other than a PR DLPDU on the local link:

NOTE A single PR DLPDU is able to convey a probe-response SPDU to the local LAS, which will cause the current local-link values of the following parameters to be returned in a DL-node-activation SPDU (see clause 11).

12.5.3.2 V(MRC) maximum-retry-count (see 6.7.1.5);

12.5.3.3 V(NDL) network-DLPDU-lifetime (see 6.7.1.7);

12.5.3.4 V(MEP) MAC-embedding-prefix, defined in 6.7.1.10;

12.5.4 Additional node-independent DL-configuration parameters for link-master class DLEs

The following DL-parameters shall be assigned the then-current values for the local link before the DLE is permitted to function as a DLE of the Link-master (see 6.6.2) or Bridge (see 6.6.3) class (that is, as a potential LAS ready to assume the LAS role), and transmit on the local link:

12.5.4.1 V(MST) maximum-scheduled-traffic, defined in 6.7.5.5;

12.5.4.2 V(DTHT) default-token-holding-time, defined in 6.7.5.8;

12.5.4.3 V(TTRT) target-token-rotation-time, defined in 6.7.5.11;

12.5.4.4 V(NUN) number-of-consecutive-unpolled-nodes ids, defined in 6.7.5.16;

12.5.4.5 V(TDP) time-distribution-period, defined in 6.7.5.18;

12.5.4.6 V(LDDP) LAS-data-base-distribution-period, defined in 6.7.5.20.

12.5.5 Additional node-independent DL-configuration parameters for bridge class DLEs

Since every Bridge class DLE has Link Master capability, 12.5.4 also applies to Bridge class DLEs.

The following DL-parameters shall be assigned the then-current values for the local link before the DLE is permitted to function as a DLE of the Bridge (see 6.6.3) class and transmit on the local link:

12.5.5.1 –all of the additional parameters defined in 12.5.5;

12.5.5.2 –all of the configuration parameters defined in ISO/IEC 10038.

12.5.5.3 Node-independent Ph-configuration parameters required for minimal DL-communication

This subclause enumerates the minimal set of Ph-parameters, defined in IEC 61158-2:2003¹⁰, 6.2.2 and 8.2.4, which need to be preconfigured before proper PhE operation is possible.

NOTE 1 These Ph-parameters, defined in IEC 61158-2:2003¹¹, Table 4, are included to make it possible for a device, newly connected to a fieldbus local link, to learn the minimal information required for successful communication with that local link's LAS, and thereby request any additional needed PhL and DLL configuration information, after which remote management becomes possible.

NOTE 2 Although this measure is named "post-transmission-gap-extension" in IEC 61158-2, it is a measure of the amount of observable non-transmission required between any two transmissions on the local medium, as observed from any point on that medium, whether the two transmissions are from the same source PhE or from two different source PhEs. Thus it could just as correctly be named "pre-transmission-gap-extension", or even more correctly "inter-transmission-gap-extension".

12.5.5.4 V(PhIS) maximum inter-channel signal skew.

12.5.6 DLE-collected fault-management data

NOTE DLE-collected fault-management data are designed to provide the maximum amount of information for network and end-system fault management while imposing the least performance burden on real implementations.

12.5.6.1 Required statistical measures

Each DLE should keep a minimum set of statistical measures to facilitate the identification of communications errors which render the local link inoperable, and the sources of those errors, where that information is available. These statistical measures provide a partial basis for network fault management.

NOTE 1 These statistical measures may be read through higher-layer network-management protocols.

In general, these statistical measures take the form of event counters, where a network transmission or reception event results in the incrementation of a counter. Conceptually, these counters are of unbounded length, are reset only on DLE activation or DLE reset, and record the number of events of the appropriate type which have occurred since the DLE was last activated or reset.

In general, counters which record error events need only be two octets in length, while counters that record events expected during normal error-free operation should be longer to permit infrequent loss-free readout by remote DL-management. The remainder of this subclause specifies both the minimum required size, and where different the recommended size, of each counter.

These counters shall be reset only upon DLE activation or DLE reset; they shall not be reset by any other local or remote means.

NOTE 2 This restriction on resetting the counters is necessary if they are to be meaningful in a multi-manager environment.

12.5.6.1.1 Transmission-related statistical measures

- a) The total number of jabber (streaming) faults which were detected during transmission, if such faults are detectable separately from other modem or MAU faults, or zero if such faults are not separable. This counter shall be at least two octets in length.

¹⁰ Expected date of concurrent publication.

¹¹ Expected date of concurrent publication.

- b) The total number of DLPDUs which were transmitted as immediate retries of the immediately prior transaction (see 3.4.21). This includes DLPDUs retransmitted by the DLE when it is functioning in an initiator role, and DLPDUs retransmitted by the DLE when it is functioning in a responder role. This counter shall be at least two octets in length.
- c) The total number of DLPDUs which were transmitted by the DLE, but excluding and not counting those DLPDUs counted in category b). This counter shall be at least two octets in length, and should be at least four octets in length.

12.5.6.1.2 Reception-related statistical measures

The following two statistical measures shall be kept

- separately for each PhL-medium attached to the DLE, if an implementation so permits, or
- aggregated for all of the PhL-media attached to the DLE.

NOTE 1 These statistical measures facilitate identification of PhE-related problems.

- a) The total number of modem-internal or MAU-internal faults, which were detected during transmission or reception, plus the total number of jabber (streaming) faults not reported in category 12.5.6.1.1a). This counter shall be at least two octets in length.

NOTE Implementations which do not distinguish jabber (streaming) faults from modem-internal or MAU-internal faults will count all such faults in this category.

- b) The total number of **END-OF-ACTIVITY** indication events (see 6.4.1.2) at the PhE interface for which there was not an immediately preceding **END-OF-DATA** indication event. This counter shall be at least two octets in length.

NOTE This is the number of DLPDUs and noise intervals for which reception commenced, but in which a partially received DLPDU was discarded due to detected framing or coding errors.

When a DLE is implemented so that it does not actually receive its own DLPDUs when it is transmitting, but processes them as if they had been received, then the following counters shall not be incremented as a result of a transmission by the DLE.

NOTE 2 This is equivalent to saying that the DLE operates in a physically half-duplex mode.

The following additional statistical measures shall be kept for each DLE, where that information is available:

NOTE 3 These statistical measures facilitate identification of link-related or path-related, but not PhL-media-related, problems.

- c) The total number of partially received DLPDUs which were discarded after examination of the DLPDU's frame control field and first address subfield, if any, and for which the associated FCS was not checked for correctness. This counter shall be at least two octets in length, and should be at least four octets in length.

NOTE If the receiving DLE takes any action based on the received DLPDU, such as an LAS DLE checking the frame control octet's final-use subfield, or any DLE noting the destination address of a CA, CD or ED DLPDU to enable reception of the reply DLPDU, then the receiving DLE necessarily has to check the correctness of the FCS before acting on the information in the DLPDU. In such a case, this specific statistical measure cannot, and does not, apply.

- d) The total number of completely-received DLPDUs which were discarded due to a detected incorrect FCS (see 5.1). This counter shall be at least two octets in length.
- e) The total number of DLPDUs which were discarded as inferred duplicates due to immediate retry of an immediately prior transaction. This counter shall be at least two octets in length.
- f) The total number of received DLPDUs, with a correct FCS, which were not counted in categories a) to e). This counter shall be at least two octets in length, and should be at least four octets in length.
- g) The total number of occurrences which the DLE has detected, of a timeout of one of the DLE's timers which is prescaled by the local link's slot-time, V(ST). This includes
 - 1) timeouts as initiator of a two-phase transaction requiring an immediate reply;
 - 2) timeouts as LAS while monitoring a delegated token holder;

- 3) timeouts as a prior LAS while monitoring a transfer of the LAS role to another DLE;
- 4) timeouts resulting in sending a CL DLPDU in an attempt to claim the LAS role and (re)-initialize the local link.

This counter shall be at least two octets in length.

12.5.6.1.3 Additional reception-related statistical measures required of a bridge DLE

A DLE functioning as a port of a bridge shall keep the following statistical measures in addition to those specified in 12.5.6.1.2, where that information is available.

- a) The total number of DLSDUs which were stored and forwarded for retransmission by the bridge. This counter shall be at least two octets in length, and should be at least four octets in length.
- b) The total number of octets of the DLSDUs specified in a), not counting FCS octets. This counter shall be at least two octets in length, and should be at least four octets in length.

12.5.6.2 Additional required DLE-collected fault-management data

- a) For each of the last four timeout events counted in 12.5.6.1.3g), the DLE shall retain the frame control octet and address subfields, if any, of the DLPDU which the DLE last transmitted or received (as appropriate) immediately before the occurrence of that event, where that information is available.

12.5.6.3 Additional statistical measures

A DLE may also keep additional statistical measures on other aspects of DLE operation to facilitate network and end-system fault management and performance monitoring, where that information is available.

12.5.7 DLE Variables which can be read and set by DL-management

DL-management may read and set

- a) the desired DL-node class of operation: Basic, Link Master, or Bridge;
- b) permitted ranges of QoS values and default QoS (see IEC 61158-3) values for:
 - 1) DLL priority
 - 2) DLL maximum confirm delay
 - 3) DLPDU authentication
 - 4) DL-scheduling policy
 - 5) maximum DLSDU size
 - 6) residual activity (see also 10.2.1.1f)
 - 7) maximum send and receive window sizes (see also 10.2.1.3e)
 - 8) transmit window.
- c) a default schedule for a Link Master or Bridge class DLE to use if it needs to function as the LAS, in situations where the DLE is either incapable of, or unable to, get a more current schedule from a currently functioning LAS DLE.

12.5.8 DLE Actions Requestable by DL-management

The DLE actions which DL-management may request are:

- a) invoke service primitives such as DL-CREATE, DL-BIND and DL-SCHEDULE-SEQUENCE (see IEC 61158-3), with the Action-qualifiers parameter conveying the input sub-parameters specified for the selected service, and the Additional-information parameter returning any resultant output subparameters to DL-management.

- b) change DL-node status between online and offline, or for fractional-duty-cycle (FDC) nodes between “asleep” and “awake” (8.21);
- c) for DL-nodes functioning as a Link Master or a Bridge DLE, to attempt to become the link’s active scheduler (LAS), or to request that the LAS role, when currently held, be transferred to another DLE (for example, in this latter case in anticipation of the DLE currently functioning as LAS going offline).

12.6 Implementation profiles

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

12.6.1 DL-address, queue and buffer management services

The first profile categories specifies whether an implementation supports LONG DL-addresses (see 7.2.1.1) and thus supports direct communication and direct management through a bridge (see 3.4.1).

Permitted choices are:

- YES** LONG, SHORT and VERY-SHORT address representations are supported (that is, are recognized and can be generated);
- NO** only SHORT and VERY-short address representations are supported (that is, are recognized and can be generated).

12.6.2 DL-data delivery services

The first profile category specifies the degree to which an implementation supports those communications priorities which facilitate preemptive use of the shared communications resource. The remaining profile categories specify the degree to which an implementation supports those DLC data-delivery features which enhance the unaugmented DL-capability of conveying limited amounts of DLS-user information, with limited QoS, between DLSAPs:

- a) Three priorities are defined in this standard for both connection-mode and connectionless data communications. The utility for time-critical communications of an implementation of this standard depends substantially on whether the implementation supports multiple priorities of data delivery to the DLS-user. Two levels of support, of decreasing difficulty and utility, have been specified. Which of them does the implementation support?

Permitted choices are:

- MULTI** URGENT, NORMAL and TIME-AVAILABLE priorities are all supported for both connection-mode and connectionless data communications;
- SINGLE** only TIME-AVAILABLE priority is supported; URGENT and NORMAL priorities are not supported.

- b) Four classes of DLC data delivery features are defined in the standard for both peer and multi-peer DLCs. Of these, three (CLASSICAL, ORDERED and UNORDERED) provide substantively different qualities of service. In addition, the quality of service of CLASSICAL peer-to-peer data delivery is different from that of it analogous multi-peer counterpart.

NOTE In the absence of communications errors, DISORDERED and CLASSICAL data delivery services are indistinguishable. Thus the DISORDERED data delivery service is an implementation option when CLASSICAL service is provided, and does not provide a substantively different quality of service from the CLASSICAL service.

The utility for time-critical communications of an implementation of this standard depends heavily on the degree to which the implementation supports the data delivery paradigms needed by the DLS-user. Four levels of support, of decreasing difficulty and utility, have been specified. Which of them does the implementation support?

Permitted choices are:

- A** CLASSICAL, ORDERED and UNORDERED peer and multi-peer DLCs are all supported; DISORDERED peer and multi-peer DLCs may be supported;
 - B** only ORDERED and UNORDERED peer and multi-peer DLCs, and CLASSICAL peer DLCs, are supported; CLASSICAL and DISORDERED multi-peer DLCs are not supported; DISORDERED peer DLCs may be supported;
 - C** only ORDERED and UNORDERED peer and multi-peer DLCs are supported; CLASSICAL and DISORDERED peer and multi-peer DLCs are not supported;
 - D** only UNORDERED peer and multi-peer DLCs are supported; ORDERED, CLASSICAL and DISORDERED peer and multi-peer DLCs are not supported.
- c) Multi-peer DLCs can be capable of conveying DLSDUs from subscribers to their publisher, providing a capability roughly equivalent to the inverse of a connectionless multi-cast. Is this optional capability supported?

Permitted choices are:

- YES** DLSDUs may be conveyed from subscribers to their publisher;
 - NO** DLSDUs cannot be conveyed from subscribers to their publisher.
- d) CLASSICAL, ORDERED and DISORDERED peer and publisher DLCs are capable of conveying DLSDUs up to 16 times the size of the DLSDUs conveyable with UNORDERED DLCs or connectionless services. What is the ratio of the maximum size DLSDU conveyable on CLASSICAL and ORDERED peer and multi-peer DLCs to that conveyable with UNORDERED DLCs or connectionless services?
- Permitted choices are the integers 1 to 16, inclusive. If CLASSICAL DLCs are supported and the ratios are different for CLASSICAL and ORDERED DLCs, or if the ratios are different for peer and multi-peer DLCs, then the smallest of the ratios shall be specified. If the choice specified for category b) above was **D**, then this value shall be 1 (one).
- e) ORDERED peer and multi-peer DLCs are capable of computing TRANSPARENT, RESIDENCE, UPDATE and SYNCHRONIZED timeliness of DLSDU conveyance. Which of these forms of timeliness computations are supported?
- Permitted choices are NONE, just TRANSPARENT, or TRANSPARENT plus any combination of the other three. If the choice specified for category b) above was **D**, then this value shall be NONE.
- f) ORDERED peer and multi-peer DLCs are capable of conveying DL-time stamps with DLSDUs. Are this capability, and the capability of noting the DL-time of DL-PUT request primitives, and of providing TRANSPARENT timeliness, all supported?

Permitted choices are:

- YES** all three capabilities are supported, with best achievable time resolution dependent on the time class specified in 12.6.3;
- NO** not all of these capabilities are supported.

If the choice specified for category b) above was **D**, then this value shall be NO.

- g) Three forms of connectionless data delivery are defined in the standard:
- 1) unitdata transfer with local confirmation;
 - 2) unitdata transfer with remote confirmation;
 - 3) unitdata exchange.

Which of these forms of connectionless data delivery are supported?

Permitted choices are:

- X** all three forms are supported;
- Y** both forms of unitdata transfer are supported; unitdata exchange is not supported;
- Z** only unitdata transfer with local confirmation is supported; unitdata transfer with remote confirmation and unitdata exchange are not supported;

12.6.3 DL-time and time-based scheduling services

The last three profile categories specify the degree to which an implementation supports a fieldbus-wide sense of time and time-based scheduling:

- a) The maximum asynchronism in the fieldbus-shared sense of time determines the coarseness or fineness of such shared activities as distributed time-based scheduling and distributed sequence-of-events determination. Eight classes of time synchronism have been defined:

1 µs Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most 250 ns for Basic and Link Master DLEs, and 125 ns for Bridge DLEs, under the condition that a single TD DLPDU is received once each 5 ms, as shown in Table 133.

NOTE When connected to a similar reference node by a multi-link fieldbus with no more than two intervening bridges, in which the reference node, the intervening bridges, and the node under measurement all meet this requirement, such a DL-implementation can provide a sense of time differing from that in the reference node by less than 500 ns, and thus from any node similarly removed from the reference node by less than 1 µs.

10 µs Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most 3 µs for Basic and Link Master DLEs, and 1 µs for Bridge DLEs, under the condition that a single TD DLPDU is received once each 50 ms, as shown in Table 133.

NOTE When connected to a similar reference node by a multi-link fieldbus with no more than two intervening bridges, in which the reference node, the intervening bridges, and the node under measurement all meet this requirement, such a DL-implementation can provide a sense of time differing from that in the reference node by less than 5 µs, and thus from any node similarly removed from the reference node by less than 10 µs.

100 µs Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most 25 µs for Basic and Link Master DLEs, and 10 µs for Bridge DLEs, under the condition that a single TD DLPDU is received once each 500 ms, as shown in Table 133.

NOTE When connected to a similar reference node by a multi-link fieldbus with no more than two intervening bridges, in which the reference node, the intervening bridges, and the node under measurement all meet this requirement, such a DL-implementation can provide a sense of time differing from that in the reference node by less than 50 µs, and thus from any node similarly removed from the reference node by less than 100 µs.

1 ms Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most $\frac{1}{4}$ ms for Basic and Link Master DLEs, and $\frac{1}{16}$ ms for Bridge DLEs, under the condition that a single TD DLPDU is received once each 5 s, as shown in Table 133.

NOTE When connected to a reference node by a multi-link fieldbus, in which

- a) the reference node, two bridges in series, and the node under measurement all meet this requirement, or
 - b) the reference node and three bridges in series meet the requirements of the 10 µs class, and an additional two bridges in series and the node under measurement all meet the requirement of the 1 ms class,
- such a DL-implementation can provide a sense of time differing from that in the reference node by less than 0,5 ms, and thus from any node similarly removed from the reference node by less than 1 ms.

10 ms Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most 2,5 ms for Basic and Link Master DLEs, and 0,5 ms for Bridge DLEs, under the condition that a single TD DLPDU is received once each 10 s, as shown in Table 133.

NOTE When connected to a reference node by a multi-link fieldbus, in which

- a) the reference node, two bridges in series, and the node under measurement all meet this requirement, or
- b) the reference node and five bridges in series meet the requirements of the 1 ms class, and an additional two bridges in series and the node under measurement all meet the requirement of the 10 ms class,

such a DL-implementation can provide a sense of time differing from that in the reference node by less than 5 ms, and thus from any node similarly removed from the reference node by less than 10 ms.

100 ms Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most 25 ms for Basic and Link Master DLEs, and 5 ms for Bridge DLEs, under the condition that a single TD DLPDU is received once each 25 s, as shown in Table 133.

NOTE When connected to a reference node by a multi-link fieldbus, in which

- a) the reference node, two bridges in series, and the node under measurement all meet this requirement, or
- b) the reference node and five bridges in series meet the requirements of the 10 ms class, and an additional two bridges in series and the node under measurement all meet the requirement of the 100 ms class,

such a DL-implementation can provide a sense of time differing from that in the reference node by less than 50 ms, and thus from any node similarly removed from the reference node by less than 100 ms.

1 s Under steady-state conditions, the DL-implementation can maintain a sense of DL-time whose maximum long-term mean error relative to the local LAS's sense of DL-time is at most 250 ms for Basic and Link Master DLEs, and 50 ms for Bridge DLEs, under the condition that a single TD DLPDU is received once each 55 s, as shown in Table 133.

NOTE When connected to a reference node by a multi-link fieldbus, in which

- a) the reference node, two bridges in series, and the node under measurement all meet this requirement, or
- b) the reference node and five bridges in series meet the requirements of the 100 ms class, and an additional two bridges in series and the node under measurement all meet the requirement of the 1 s class,

such a DL-implementation can provide a sense of time differing from that in the reference node by less than 0,5 s, and thus from any node similarly removed from the reference node by less than 1 s.

NONE The DL-implementation does not provide a sense of time which is synchronized with that of other nodes on a connected fieldbus.

NOTE 1 In this last case, the DL-implementation may still be able to report the last time information heard during a fieldbus DL-time broadcast, and may even be able to estimate the intervening interval. However, the DL-implementation is unable to maintain a synchronized sense of DL-time with only low-frequency time broadcasts from a reference DLE on the fieldbus (that is, from the local link's LAS).

NOTE 2 The time-synchronism class of a link's LAS determines the best achievable time synchronism for the entire link. The time-synchronism class of a bridge determines the best achievable time synchronism for that portion of the extended link which inter-communicates through the bridge.

- b) Time-based scheduling services are dependent on the time synchronism of the fieldbus DLE, and so can be provided only when some sense of time synchronism exists within the DLE and on the local link. Are local requests for time-based scheduling services, including consequent scheduling interactions with the local link's LAS DLE, supported?

Table 133 – Maximum permitted phase-tracking error in a DLE's sense of DL-time at the minimum requireable Time Distribution period

Time-synchronism class	Maximum permitted error by DLE class		Minimum requireable value for V(TDP)
	Basic or Link Master DLE	Bridge DLE	
1 μ s	250 ns	125 ns	5 ms
10 μ s	3 μ s	1 μ s	50 ms
100 μ s	25 μ s	10 μ s	500 ms
1 ms	250 μ s	1/16 ms	5 s
10 ms	2,5 ms	0,5 ms	10 s
100 ms	25 ms	5 ms	25 s
1 s	250 ms	50 ms	55 s
none	> 100 yr	> 100 yr	not applicable

Permitted choices are:

YES PERIODIC schedules, and ONE-TIME schedules with a specified starting time, are supported, with the best achievable time resolution dependent on the time class specified in 12.6.3;

NO these capabilities are not supported.

If the choice specified for category a) above was NONE, then this value shall be NO.

NOTE A DLE with no sense of DL-time may still be able to respond to time-based scheduled activities, such as periodic receipt of an ES DLPDU enabling periodic local schedule execution within the DLE. However, such a DLE cannot make a time-based scheduling request to the LAS, because it cannot specify the start time for the request with reference to the LAS' DL-time. Thus all such activities can only occur due to DL-management preconfiguration of the DLE through extra-protocol means, and not as a result of DLS-user requests for DL-service.

- c) Time-based scheduling activities within an LAS DLE also are dependent on the time synchronism of the fieldbus DLE, and so can be provided only when some sense of time synchronism exists. Five classes of support for these services have been defined:

DYNAMIC The DL-implementation of LAS functions supports

- 1) time-based schedules which have been pre-constructed by DL-management;
- 2) requests for time-based scheduling services received from DLEs on the local link and modification of the current schedule to accommodate those requests, where feasible;
- 3) reception of the current schedule from a previous LAS DLE;
- 4) redistribution of the current schedule to other potential LAS DLEs;

SHARABLE The DL-implementation of LAS functions supports

- 1) time-based schedules which have been pre-constructed by DL-management;
- 2) time-based schedules which have been pre-constructed by a previous DYNAMIC-class LAS DLE;
- 3) reception of the current schedule from a previous LAS DLE;
- 4) redistribution of the current schedule to other potential LAS DLEs.

Such a DLE does not have the ability to construct schedules dynamically.

UPDATABLE The DL-implementation of LAS functions supports

- 1) time-based schedules which have been pre-constructed by DL-management;
- 2) time-based schedules which have been pre-constructed by a previous DYNAMIC-class LAS DLE;
- 3) reception of the current schedule from a previous LAS DLE.

Such a DLE does not have the ability to redistribute the current schedule to other potential LAS DLEs.

NOTE This inability to redistribute the schedule implies that a received schedule is transformed into an internal representation in which some of the received scheduling information has been discarded.

STATIC The DL-implementation of LAS functions supports time-based schedules which have been pre-constructed by DL-management. It does not have the ability to receive the current schedule from a previous LAS DLE.

NONE Either

- 1) the DLE is a BASIC DLE, or
- 2) the implemented LAS functions do not support time-based schedules, but do support free-running schedules which have been pre-constructed by DL-management.

13 Type 1: PICS proforma

NOTE This clause only partially follows ISO/IEC Directives Part 3 because it must be usable as a separable standalone document.

13.1 Introduction

To evaluate conformance to a particular implementation, it is necessary to have a statement of which capabilities and options have been implemented for a given OSI protocol. Such a statement is called a Protocol Implementation Conformance Statement (PICS).

13.2 Scope

This clause provides the PICS proforma for the Type 1 fieldbus Data Link Layer protocol as specified in the body of this standard, in compliance with the relevant requirements, and in accordance with the relevant guidelines, given in ISO/IEC 9646-2.

When the supplier completes this proforma, this document becomes the PICS for the specified implementation. The PICS is then used to perform static conformance review of the implementation to determine that all mandatory features are implemented and that all conditional features are correctly supported dependent on the options claimed to be implemented. The PICS also is used as an aid in test case selection.

NOTE It is anticipated that this PICS will be extended in future revisions of this standard.

13.2.1 Audience

The PICS proforma shall be used by the supplier of an implementation to state the capabilities of the implementation. The completed PICS shall be used by the test operator and the test analyst to determine the static conformance of an implementation under test (IUT), to select test cases and test suites to use when testing the IUT, and to aid in test results analysis. The completed PICS may be used by a user to determine the adequacy of an implementation.

13.3 Normative references

See the current editions of ISO/IEC 9646-1 and ISO/IEC 9646-2, as listed in the Bibliography.

13.4 Definitions

This clause uses the following terms defined in ISO/IEC 9646-1:

- a) PICS proforma;
- b) protocol implementation conformance statement (PICS);
- c) static conformance review.

13.5 Abbreviations

The following abbreviations are used in defining the status of a feature, timer, parameter, or other capability:

M	: mandatory;
O	: optional;
X	: prohibited;
N/A	: not applicable;
C	: conditional.

The following abbreviations are used in defining the support of a feature, timer, parameter, or other capability:

Y or **y** or **yes** : implemented;

N or **n** or **no** : not implemented

The following abbreviations are used in defining the range or existence of a feature, parameter, or other capability:

F : false;

T : true

References of the form “/n” or “/n.x” refer to the conformance claimed for row “n”, or for row “n” column “x” respectively, in the same table. References of the form “m/n” or “m/n.x” are a corresponding reference to the conformance claimed in table “m”.

13.6 Conformance

The supplier of a protocol implementation which is claimed to conform to IEC 61158-4, Type 1 is required to complete a copy of the PICS proforma provided in this clause and is required to provide the information necessary to identify both the supplier and the implementation.

13.7 Instructions

The supplier of a fieldbus Data Link protocol implementation shall take the following steps to complete the PICS proforma.

The supplier shall complete the identification section.

The supplier shall enter either a ‘Y’ or an ‘N’ (or equivalents), as just defined under ‘Abbreviations’, in each entry in the support columns of the following tables. The supplier shall enter a ‘Y’ if the feature, timer, parameter, or capability is implemented. When a conformance requirement defines the allowable or required range or set of values for a capability or parameter, the “allowed” or “required” column contains the specified value(s), and the supplier of the implementation shall only enter a ‘Y’ in the support entry if the implementation completely supports the specified range or set of values. The supplier shall enter a ‘Y’ in the support entry even if access to the feature is not provided as an option (that is, the feature is unconditionally present).

The supplier shall enter an ‘N’ if the feature, timer, parameter, or capability is not implemented at all. If the supplier enters an ‘N’ for a mandatory feature, then the supplier shall attach an explanation as to why the feature was not implemented.

The supplier shall enter the specified ranges, values, default value (as set by the supplier of the implementation) and units of measure, where applicable, for all implemented parameters. Continuous ranges shall be entered in the form: lowest-value ‘:’ highest-value. Series of discrete values or subranges shall be entered from lowest value to highest value, with each value or subrange separated by commas or semicolons.

If the units of measure, or any other column, is not applicable, then a ‘—’ should be recorded in the appropriate entry.

13.8 Identification

Date:

Data Link Layer Implementation Information

Implementation Name:

Implementation Identifier (including version/release):

Vendor Information

Company Name:

Technical Representative:

Address:

Telephone:

Facsimile:

Telex:

E-mail

Protocol Identification

Conformance is hereby claimed to **IEC 61158-4 : 2003, Digital data communications for measurement and control – Fieldbus for use in industrial control systems – Part 4: Type 1 Data Link protocol specification**

The protocol versions supported are:

Overall Conformance Statement

Are all mandatory capabilities of the claimed version(s) of the above protocol fully supported? _____

Are all mandatory capabilities of the profiles(s) claimed under 13.9 fully supported? _____

Answering “No” to either of these questions indicates non-conformance to the protocol specification. Non-supported mandatory capabilities are to be identified in the PICS, with an explanation of why the implementation is non-conforming.

13.9 Implementation profile

This subclause identifies the major implementation choices which limit the DL-services and DL-service features which the implementation provides and supports. It is based on 12.6.

13.9.1 DL-address, queue and buffer management services

The implementation may, but need not, support LONG DL-addresses (see 12.6.1).

NOTE Direct communication and direct management through bridges is based upon the use of LONG DL-addresses.

Supported DL-address lengths				
Item number	Supported DL-address lengths	A) Reference	B) Status	C) Support
1	LONG DL-addresses	12.6.1	O	
2	SHORT DL-addresses	12.6.1	M	
3	VERY-SHORT DL-addresses	12.6.1	M	

13.9.2 Data delivery services

13.9.2.1 Data delivery priorities

The implementation must support the TIME-AVAILABLE priority of data transfer. It may also support other priorities: NORMAL and URGENT.

Supported data delivery priorities				
Item number	Supported data delivery priorities	A) Reference	B) Status	C) Support
1	TIME-AVAILABLE	12.6.2 a	M	
2	NORMAL	12.6.2 a	O	
3	URGENT	12.6.2 a	C1	
C1 : if /2 then M else X				

13.9.2.2 Connection-mode data delivery features

The implementation must support UNORDERED peer-to-peer and multi-peer DL-connections. It may also support other DLC data delivery features: ORDERED, CLASSICAL and DISORDERED.

Supported DLCEP classes and data delivery features				
Item number	Supported DLCEP class(es) and data delivery features	A) Reference	B) Status	C) Support
1 (A)	CLASSICAL, ORDERED and UNORDERED peer and multi-peer DLCs, with DISORDERED DLCs either unsupported, or supported for both peer and multi-peer	12.6.2 b	C1	
2 (B)	only ORDERED and UNORDERED peer and multi-peer DLCs, and CLASSICAL peer DLCs, with DISORDERED peer DLCs optional	12.6.2 b	C1	
3 (C)	only ORDERED and UNORDERED peer and multi-peer DLCs	12.6.2 b	C1	
4 (D)	only UNORDERED peer and multi-peer DLCs	12.6.2 b	C1	
C1 : exactly one of these alternatives must be chosen				

13.9.2.3 Support for subscribers-to-publisher DLSDUs

The implementation may provide the ability for subscribers on a multi-peer DLC to transmit DLSDUs to their publisher, providing a capability roughly equivalent to the inverse of a connectionless multi-cast.

Support for subscribers-to-publisher DLSDUs				
Item number	Support for subscribers-to-publisher DLSDUs	A) Reference	B) Status	C) Support
1	UNORDERED multi-peer DLCs	12.6.2 c	O	

13.9.2.4 Ratio of connection-mode maximum DLSDU size to maximum DLPDU-data size

The maximum-size DLSDU which can be transferred by an unordered DLC is the same as that which can be conveyed at the same priority by the connectionless data transfer service. ORDERED, CLASSICAL and DISORDERED DLCs provide the capability to transfer DLSDUs that are up to 16 times as large, by segmenting the DLSDU into 16 or fewer segments which will be transferred by separate DLPDUs. Thus the maximum ratio of DLSDU-size to DLPDU-data-size is 1 for UNORDERED DLCs and connectionless data transfer, and is between 1 and 16, inclusive for ORDERED, CLASSICAL and DISORDERED DLCs.

Supported maximum DLSDU : DLPDU-data ratios				
Item number	Maximum DLSDU-size / DLPDU-data size ratios	Reference	Allowed range	Support
1	CLASSICAL multi-peer DLCs	12.6.2 d	0:16 C1	
2	CLASSICAL peer DLCs	12.6.2 d	0:16 C1	
3	ORDERED peer and multi-peer DLCs	12.6.2 d	0:16	
4	UNORDERED peer and multi-peer DLCs	12.6.2 d	1	
C1 : if the supported value is non-zero then it must be equal to the value claimed in /3				

13.9.2.5 DL-timeliness

The implementation may provide the ability to record the DL-time at which a DLSDU is written to a buffer; to compute the timeliness of that buffer writing, or of subsequent buffer reading, or both, according to pre-established criteria; to convey that timeliness across a DLC; and to report both sending and receiving timeliness to a receiving DLS-user when delivering the DLSDU.

DL-timeliness support				
Item number	Supported types of timeliness	A) Reference	B) Status	C) Support
1	NONE	12.6.2 e	M	
2	TRANSPARENT	12.6.2 e	C1	
3	RESIDENCE	12.6.2 e	C2	
4	UPDATE	12.6.2 e	C2	
5	SYNCHRONIZED	12.6.2 e	C2	
C1 : if 13.9.2.2/4.C then X else O				
C2 : if /2.C then O else X				

13.9.2.6 DL-time-stamped DLSDUs

The implementation may provide the ability to record the DL-time at which a DLSDU is written to a buffer by a DL-PUT request primitive and to convey that time across a DLC for subsequent use by a receiving DLE or for reporting to a receiving DLS-user when delivering the DLSDU.

DL-time-stamped DLSDU support				
Item number	DL-time-stamped DLSDU support	A) Reference	B) Status	C) Support
1	NO	12.6.2 f	M	
2	YES	12.6.2 f	C1	
C1 : if 13.9.2.5/2.C then O else X				

13.9.2.7 Connectionless data delivery features

The implementation must support unitdata transfer with local confirmation. It may also support unitdata transfer with remote confirmation, and unitdata exchange.

Supported connectionless data delivery features				
Item number	Supported connectionless data delivery features	A) Reference	B) Status	C) Support
1 (X)	unitdata exchange, and unitdata transfer with both local and remote confirmation	12.6.2 g	C1	
2 (Y)	only unitdata transfer with both local and remote confirmation	12.6.2 g	C1	
3 (Z)	only unitdata transfer with local confirmation	12.6.2 g	C1	
C1 : exactly one of these alternatives must be chosen				

13.9.3 DL-time and scheduling services

13.9.3.1 DL-time synchronism

The implementation may provide time synchronism with other devices on the extended fieldbus.

Time synchronism support						
Item number	Measure of provided time synchronism	A) Reference	B) Status	C) Support	Minimum required value for V(TDP)	
					D) Range	E) Support
1	none	12.6.3 a	C1		none	
2	1 s	12.6.3 a	C1		≥ 55 s	C2
3	100 ms	12.6.3 a	C1		≥ 25 s	C3
4	10 ms	12.6.3 a	C1		≥ 10 s	C4
5	1 ms	12.6.3 a	C1		≥ 5 s	C5
6	100 µs	12.6.3 a	C1		≥ 500 ms	C6
7	10 µs	12.6.3 a	C1		≥ 50 ms	C7
8	1 µs	12.6.3 a	C1		≥ 5 ms	C8
C1 : exactly one of these alternatives must be chosen C2 : if /2.C then M else X C3 : if /3.C then M else X C4 : if /4.C then M else X C5 : if /5.C then M else X C6 : if /6.C then M else X C7 : if /7.C then M else X C8 : if /8.C then M else X						

13.9.3.2 Time-based scheduling services

The implementation may support local time-based scheduling services.

Time-based scheduling service support				
Item number	Support for PERIODIC schedules and ONE-TIME schedules with a specified starting time	A) Reference	B) Status	C) Support
1	YES	12.6.3 b	C1	
2	NO	12.6.3 b	C2	
C1: if time_synchronism_supported then C2 else X		time_synchronism_supported = 13.9.3.1/2 or /3 or /4 or /5 or /6 or /7 or /8		
C2: exactly one of these alternatives must be chosen				

13.9.3.3 LAS time-based scheduling activities

The fieldbus Data Link layer protocol requires at least one device on each local link to be functioning as a Link Master or a Bridge. Both Link Master and Bridge class devices are capable of initializing the token on the local link, and of serving as the local Link Active Scheduler (LAS). When functioning as an LAS, these devices provide centralized schedule-execution services, and may provide centralized schedule distribution and schedule construction services, for the entire link.

Bridge-class and Link Master-class devices can be full-capability devices, capable of creating a schedule for the local link, resources permitting, from any combination of scheduling requests received from the various devices on the link. Such devices instead may offer lesser capability, permitting construction of reduced-functionality networks. This reduced-functionality may be adequate for many devices which will become the LAS only if there are no bridges or higher-capability devices active on the link, or when the intended usage is only on preconfigured local links.

Five levels of scheduling support for Link Master-class and Bridge-class devices have been defined:

LAS time-based scheduling activity support				
Item number	LAS time-based scheduling activity support	A) Reference	B) Status	C) Support
1	DYNAMIC — can construct, distribute, receive and execute time-based schedules for the local link	12.6.3 c	C1	
2	SHARABLE — can distribute, receive and execute time-based schedules for the local link, but cannot construct such schedules	12.6.3 c	C1	
3	UPDATABLE — can receive and execute time-based schedules for the local link, but cannot distribute such schedules	12.6.3 c	C1	
4	STATIC — can execute time-based schedules for the local link, but cannot receive such schedules	12.6.3 c	C1	
5	NONE — no time-based scheduling for link	12.6.3 c	C2	
C1: if time_synchronism_supported then C2 else X		time_synchronism_supported = 13.9.3.1/2 or /3 or /4 or /5 or /6 or /7 or /8		
C2: exactly one of these alternatives must be chosen				

13.10 Major low-level capabilities

This subclause identifies many of the major conformance requirements of the implementation. The conformance requirements specified in this subclause are those related to the lower-level medium-access functions of the DL-protocol. Subclause 13.11 specifies the conformance requirements for the higher-level functions of the protocol — those which are independent of the underlying medium-access functions.

In a number of cases (13.10.3.1 and 13.10.3.2, and 13.10.4.4 to 13.10.4.8), conformance requirements for reception are Mandatory, but with the added note “only a minimally responsive subset need be implemented.” This should be interpreted as meaning that when reception of the specified DLPDU initiates some sort of response, only a minimal-effort response, rejecting the requested action, needs to occur, and that complete handling of the received DLPDU’s parameter and DLS-user data fields is not required for such implementations.

13.10.1 DLE classes implemented

The fieldbus Data Link layer protocol defines three major lower-level classes of DLE operation, each a superset of the prior class. The support requirements for these classes differ.

DLE Classes Implemented				
Item number	DLE Class	Reference	Status	Support
1	Basic	6.6.1	M	
2	Link Master	6.6.2	O	
3	Bridge	6.6.3	C1	
C1 : if /2 then O else X				

13.10.2 Basic addressing requirements

DL-addresses have multiple lengths, and potentially have two synonyms. Where such synonyms exist, the choice of the appropriate synonym for use in a specific DLPDU is dependent both on the required QoS and on the availability of similar synonyms for all of the other addresses in the DLPDU.

Basic addressing requirements				
Item number	Addressing requirements	Reference	Status	Support
1	Are all LONG DL-addresses four octets in length?	7.2.2.1, 6.3.1	M	
2	Are all SHORT DL-addresses two octets in length?	7.2.2.2, 6.3.2	M	
3	Are all DL-addresses interpreted as specified in Table 2, Table 3 and Table 5?	6.3.2	M	
4	Are all predefined DL-addresses for the implemented classes claimed under 13.10.1 recognized as specified in Table 6 through Table 8?	6.3.3	M	
5	Are all LONG DL-addresses with a link-designator subfield (HL) whose value is not equal to zero and is not equal to V(TL) considered non-local?	6.3.2	M	
6	Are all LONG DL-addresses with a link-designator subfield (HL) whose value is zero or is V(TL), and with a node-designator subfield whose value is not equal to zero and is not equal to V(TN), considered link-local?	6.3.2	M	
7	Are all LONG DL-addresses with a link-designator subfield (HL) whose value is zero or is V(TL), and with a node-designator subfield whose value is equal to zero or is equal to V(TN), considered node-local?	6.3.2	M	
8	Are all SHORT DL-addresses with a node-designator subfield (N) whose value is not equal to zero and is not equal to V(TN), considered link-local?	6.3.2	M	
9	Are all SHORT DL-addresses with a node-designator subfield (N) whose value is equal to zero or is equal to V(TN), considered node-local?	6.3.2	M	
10	During address recognition, are all node-local DL-addresses with the same subnode selector (S) considered equivalent?	6.3.2	M	
11	During address recognition, are all link-local DL-addresses with the same sublink selector (N.S) considered equivalent?	6.3.2	M	
12	During DLPDU formation, when the DLS-user's DLPDU-authentication QoS was SOURCE or MAXIMAL, then is a source DL-address included within each resultant DLPDU, when possible?	6.2.4	M	
13	During DLPDU formation, when the DLS-user's DLPDU-authentication QoS was MAXIMAL, then are all DL-addresses within each resultant DLPDU LONG?	6.2.4	M	
14	During DLPDU formation, when at least one of the DLPDU's addresses is non-local, then are all DL-addresses within that DLPDU LONG?	7.2.1.1	M	
15	During DLPDU formation, when a DL-address is both link-local and LONG, then is the link component of the DL-address equal to the local link's link_id, and not zero unless that link_id is itself zero?	6.3.2.1	M	
16	During DLPDU formation, when a DL-address is node-local, then is the node component of the DL-address equal to the node's node_id, and not zero?	7.2.2.1, 6.3.2.2	M	
17	During DLPDU formation, when the DLS-user's DLPDU-authentication QoS was ORDINARY or SOURCE, and all DL-addresses within the DLPDU are link-local, then are those DL-addresses all SHORT?	6.2.4	M	

13.10.3 DLPDU support

This subclause enumerates the types of DLPDU which the implementation can receive and recognize, and, separately, which the implementation can transmit, as a function of the highest conformance class claimed under 13.10.1; and separately as a function of the momentary role within the local data link assumed by the implementation.

NOTE It is assumed that an implementation can receive (in a rudimentary sense) all DLPDUs, whether well-formed or not, but that well-formed DLPDUs which are not recognized are ignored.

13.10.3.1 Static DLPDU support

The fieldbus Data Link layer protocol defines three major classes of operation, each a superset of the prior class. This table specifies the static (that is, independent of momentary role) conformance requirements as a function of the claimed capabilities of the implementation. Only the entries for the highest claimed class (right-most in the table) need be completed.

Class dependency of supported DLPDUs														
Item number	Column letter →		Basic class				Link Master class				Bridge class			
			A) Receiving and recognizing		B) Forming and sending		C) Receiving and recognizing		D) Forming and sending		E) Receiving and recognizing		F) Forming and sending	
	DLPDU type	Reference	Stat-us	Sup-port	Stat-us	Sup-port	Stat-us	Sup-port	Stat-us	Sup-port	Stat-us	Sup-port	Stat-us	Sup-port
1	EC	8.1	M1		M1		M1		M1		M1		M1	
2	DC	8.2	M1		M1		M1		M1		M1		M1	
3	RC	8.3	M1		C2		M1		C2		M1		M1	
4	CA	8.4	M1		C3		M1		C3		M		M	
5	CD	8.5	M		M		M		M		M		M	
6	ED	8.6	M1		C4		M1		C4		M		M	
7	DT	8.7	M		M		M		M		M		M	
8	SR	8.8	O		M		M		M		M		M	
9	CT	8.9	X		C5		M		C5		M		C5	
10	TD	8.10	C5		X		C5		M		C5		M	
11	RQ	8.11	M		C5		M		C5		M		C5	
12	RR	8.12	C5		M		C5		M		C5		M	
13	PN	8.13	M		X		M		M		M		M	
14	PR	8.14	M		M		M		M		M		M	
15	PT	8.15	M		X		M		M		M		M	
16	ES	8.16	M1		X		M1		M		M1		M	
17	RT	8.17	X		M		M		M		M		M	
18	RI	8.18	X		O		M		O		M		O	
19	CL	8.19	X		X		M		M		M		M	
20	TL	8.20	X		X		M		M		M		M	
21	WK	8.21	O		X		O		M		X		M	
22	IDLE	8.22	X		O		X		O		X		O	
23	spare	8.23	X		X		X		X		X		X	
24	don't use	8.24	X		X		X		X		X		X	
M1: only a minimally-responsive subset need be implemented C2: if reliable_CO then M else X C3: if acknowledged_CL then M elseif 13.9.2.3/1 then O else X C4: if 13.9.2.7/1 then M else O C5: if time_synchronism_supported then M else O							reliable_CO = 13.9.2.2/1 or /2 acknowledged_CL = 13.9.2.7/1 or /2 time_synchronism_supported = 13.9.3.1/2 or /3 or /4 or /5 or /6 or /7 or /8							

13.10.3.2 Dynamic DLPDU support

The fieldbus Data Link layer protocol defines four major roles for operating devices. This table specifies the dynamic (that is, dependent on momentary role) conformance requirements of the implementation as a function of the then-appropriate role. For an implementation claiming only Basic Class conformance, only the left-most four columns (A–D) need to be completed. For all other implementations, all columns need to be completed.

Role dependency of supported DLPDUs																		
			When DLE's operational Node address is unknown and DLE has assumed a temporary Node address				When DLE's operational Node address is known and DLE is functioning as a Basic DLE				When DLE's operational Node address is known and DLE is functioning as a Link Master or Bridge DLE and DLE is not acting as the LAS				When DLE's operational Node address is known and DLE is functioning as a Link Master or Bridge DLE and DLE is acting as the LAS			
	Column letter →		A) Receiving and recog- nizing		B) Forming and sending		C) Receiving and recog- nizing		D) Forming and sending		E) Receiving and recog- nizing		F) Forming and sending		G) Receiving and recog- nizing		H) Forming and sending	
Item number	DLPDU type	Refer- ence	Stat us	Sup port	Stat us	Sup port	Stat us	Sup port	Stat us	Sup port	Stat us	Sup port	Stat us	Sup port	Stat us	Sup port	Stat us	Sup port
1	EC	8.1	O		X		M1		M1		M1		M1		M1		M1	
2	DC	8.2	O		X		M1		M1		M1		M1		M1		M1	
3	RC	8.3	O		X		M1		C2		M1		C2		M1		M1	
4	CA	8.4	M1		X		M1		C3		M1		C3		M		M	
5	CD	8.5	M		X		M		M		M		M		M		M	
6	ED	8.6	M1		X		M1		C4		M1		C4		M		M	
7	DT	8.7	M		X		M		M		M		M		M		M	
8	SR	8.8	O		X		M		M		M		M		M		M	
9	CT	8.9	X		X		X		C5		M		C5		M		C5	
10	TD	8.10	C5		X		C5		X		C5		M		C5		M	
11	RQ	8.11	X		X		M		C5		M		C5		M		C5	
12	RR	8.12	X		X		C5		M		C5		M		C5		M	
13	PN	8.13	M		X		M		X		M		X		X		M	
14	PR	8.14	M		M		M		M		M		M		M		X	
15	PT	8.15	M		X		M		X		M		X		X		M	
16	ES	8.16	X		X		M1		X		M1		X		X		M	
17	RT	8.17	X		X		X		M		X		M		M		X	
18	RI	8.18	X		X		X		O		X		O		M		X	
19	CL	8.19	X		X		X		X		M		M		M		X	
20	TL	8.20	X		X		X		X		M		X		X		M	
21	WK	8.21	X		X		M		X		M		X		X		M	
22	IDLE	8.22	X		X		X		O		X		O		X		O	
23	spare	8.23	X		X		X		X		X		X		X		X	
	don't use	8.24	X		X		X		X		X		X		X		X	
M1 : only a minimally responsive subset need be implemented																		
C2 : if reliable_CO then M else X										reliable_CO = 13.9.2.2/1 or /2								
C3 : if acknowledged_CL then M elseif 13.9.2.3/1 then O else X										acknowledged_CL = 13.9.2.7/1 or /2								
C4 : if 13.9.2.7/1 then M else O																		
C5 : if time_synchronism_supported then M else O										time_synchronism_supported = 13.9.3.1/2 or /3 or /4 or /5 or /6 or /7 or /8								

13.10.4 DLPDU-class specific fields and subfields

This subclause enumerates the ranges and states the static conformance requirements for each DLPDU-class-specific field and subfield of each DLPDU. Conformance is only required on those DLPDU classes which were claimed under 13.10.3.1 and 13.10.3.2, and then only for the direction(s) — receiving and recognizing, or forming and sending, or both — which were claimed. Thus any conditions or options specified in the relevant entry or entries in 13.10.3.1 and 13.10.3.2 apply uniformly to all of the specific requirements of the DLPDU class, as detailed in the following subclauses, for the corresponding direction of use. Requirements in the following DLPDU-specific subclauses for directions of use not claimed under either 13.10.3.1 or 13.10.3.2 can be ignored.

13.10.4.1 Specific fields and subfields of the EC DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.1.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.1.1.1	M		M	
2.2	FINAL		M		M	
3	DL address formats:	7.2.2,				
3.1	format 1L — LONG destination and two sources	8.1.1.1, 8.1.1.2	M		M	
3.2	format 1S — SHORT destination and two sources		M		M	
3.3	format 2L — two LONG sources only		M		M	
3.4	format 2S — two SHORT sources only		M		M	
4	EC-parameter:	8.1.1.3,				
4.0	version 1	9.1	M		M	
5	permitted octets of user data in DLPDU	7.2.4, 8.1.1.4	0 : 128 octets M		0 : 128 octets M	
C1 : if 13.9.1/1 then M else X						

13.10.4.2 Specific fields and subfields of the DC DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.2.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.2.1.1	M		M	
2.2	FINAL		M		M	
3	DL address formats:	7.2.2,				
3.1	format 1L — LONG destination and source	8.2.1.1,	M		M	
3.2	format 1s — SHORT destination and source	8.2.1.2	M		M	
3.3	format 2L — LONG source only		M		M	
3.4	format 2s — SHORT source only		M		M	
4	DC-parameter:	8.2.1.3,				
4.0	version 1		M		M	
5	permitted octets of user data in DLPDU	7.2.4,	0 : 128		0 : 128	
		8.2.1.4	octets M		octets M	
C1 : if 13.9.1/1 then M else X						

13.10.4.3 Specific fields and subfields of the RC DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.3.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.3.1.1	M		M	
2.2	FINAL		M		M	
3	priority designator	7.2.1.3,				
3.1	URGENT	8.3.1.1	C2		C2	
3.2	NORMAL		C2		C2	
3.3	TIME-AVAILABLE		M		M	
4	DL address formats:	7.2.2,				
4.1	format 1L — LONG destination and source	8.3.1.1, 8.3.1.2	M		M	
4.2	format 1S — SHORT destination and source		M		M	
4.3	format 2L — LONG source only		M		M	
4.4	format 2S — SHORT source only		M		M	
5	RC-parameter:	8.3.1.3,				
5.0	version 1		M		M	
Item number	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
6	permitted octets of user data in DLPDU	7.2.4,				
6.1	URGENT priority	8.5.1.4	0 : 64 C2		0 : 64 C2	
6.2	NORMAL priority		0 : 128 C2		0 : 128 C2	
6.3	TIME-AVAILABLE priority		0 : 256 M		0 : 256 M	
C1 : if 13.9.1/1 then M else X						
C2 : if 13.9.2.1/2 then M else x						

13.10.4.4 Specific fields and subfields of the CA DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.4.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.4.1.1	M		M	
2.2	FINAL		M		M	
3	priority designator	7.2.1.3,				
3.1	URGENT	8.4.1.1	M2		C3	
3.2	NORMAL		M2		C3	
3.3	TIME-AVAILABLE		M		M	
4	DL address formats	7.2.2,				
4.1	format 1L — LONG destination and source	8.4.1.1, 8.4.1.2	M		M	
4.2	format 1s — SHORT destination and source		M		M	
4.3	format 2L — LONG destination, omitted source		M		M	
4.4	format 2s — SHORT destination, omitted source		M		M	
5	SD-parameter:	8.4.1.3,				
5.0	version 1 supported	9.4	M		M	
Item number	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
6	permitted octets of user data in DLPDU	7.2.4,				
6.1	URGENT priority	8.4.1.4	0 : 64 M2		0 : 64 C3	
6.2	NORMAL priority		0 : 128 M2		0 : 128 C3	
6.3	TIME-AVAILABLE priority		0 : 256 M		0 : 256 M	
C1 :if 13.9.1/1 then M else X M2 :if 13.9.2.1/2 then M else only a minimally responsive subset need be implemented C3 :if 13.9.2.1/2 then M else X						

13.10.4.5 Specific fields and subfields of the CD DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.5.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.5.1.1	M		M	
2.2	FINAL		M		M	
3	priority designator	7.2.1.3,				
3.1	URGENT	8.5.1.1	M2		C3	
3.2	NORMAL		M2		C3	
3.3	TIME-AVAILABLE		M		M	
4	DL address formats	7.2.2,				
4.1	format 1L — LONG destination and source	8.5.1.1,	M		M	
4.2	format 1S — SHORT destination and source	8.5.1.2	M		M	
4.3	format 2L — LONG destination, omitted source		M		M	
4.4	format 2S — SHORT destination, omitted source		M		M	
5	SD-parameter:	8.5.1.3,				
5.0	version 1 supported	9.4	M		M	
6	user data in DLPDU	8.5.1.4	X		X	
C1 :if 13.9.1/1 then M else X M2 :if 13.9.2.1/2 then M else only a minimally responsive subset need be implemented C3 :if 13.9.2.1/2 then M else X						

13.10.4.6 Specific fields and subfields of the ED DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.6.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.6.1.1	M		M	
2.2	FINAL		M		M	
3	priority designator	7.2.1.3,				
3.1	URGENT	8.6.1.1	M2		C3	
3.2	NORMAL		M2		C3	
3.3	TIME-AVAILABLE		M		M	
4	DL address formats	7.2.2,				
4.1	format 1L — LONG destination and source	8.6.1.1, 8.6.1.2	M		M	
4.2	format 1S — SHORT destination and source		M		M	
4.3	format 2L — LONG destination, omitted source		M		M	
4.4	format 2S — SHORT destination, omitted source		M		M	
5	SD-parameter:	8.6.1.3,				
5.0	version 1 supported	9.4	M		M	
Item number	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
6	permitted octets of user data in DLPDU	7.2.4,				
6.1	URGENT priority	8.6.1.4	0 : 64 M2		0 : 64 C3	
6.2	NORMAL priority		0 : 128 M2		0 : 128 C3	
6.3	TIME-AVAILABLE priority		0 : 256 M		0 : 256 M	
C1 :if 13.9.1/1 then M else X						
M2 :if 13.9.2.1/2 then M else only a minimally responsive subset need be implemented						
C3 :if 13.9.2.1/2 then M else X						

13.10.4.7 Specific fields and subfields of the DT DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.7.1.1	C1		C1	
1.2	SHORT		M		M	
2	final token use designator	7.2.1.2,				
2.1	NOT-FINAL	8.7.1.1	M		M	
2.2	FINAL		M		M	
3	priority designator	7.2.1.3,				
3.1	URGENT	8.7.1.1	M2		C3	
3.2	NORMAL		M2		C3	
3.3	TIME-AVAILABLE		M		M	
4	DL address formats	7.2.2,				
4.1	format 1L — LONG destination and source	8.7.1.1, 8.7.1.2	M		M	
4.4	format 1S — SHORT destination and source		M		M	
4.2	format 2L — LONG destination, omitted source		M		M	
4.5	format 2S — SHORT DESTINATION, omitted source		M		M	
4.3	format 3L — long SOURCE only		M		M	
4.6	format 3S — short SOURCE only		M		M	
4.7	format 4 — omitted DESTINATION and omitted source		M		M	
4.8	format 5 — omitted SOURCE only		M		M	
5	SD-parameter:	8.7.1.3,				
5.0	version 1 SUPPORTED		M		M	
Item number	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
6	permitted octets of user data in DLPDU	7.2.4,				
6.1	URGENT priority	8.7.1.4	0 : 64 M2		0 : 64 C3	
6.2	NORMAL priority		0 : 128 M2		0 : 128 C3	
6.3	TIME-AVAILABLE priority		0 : 256 M		0 : 256 M	
C1 if 13.9.1/1 then M else X						
M2 if 13.9.2.1/2 then M else only a minimally responsive subset need be implemented						
C3 if 13.9.2.1/2 then M else X						

13.10.4.8 Specific fields and subfields of the SR DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: omitted destination, NODE source	7.2.2.3, 8.8.1.2	M (16: 255)		M (16: 255)	
2	SR-parameter	8.8.1.3	M		M	

13.10.4.9 Specific fields and subfields of the CT DLPDU

The CT DLPDU has no variable fields or subfields.

13.10.4.10 Specific fields and subfields of the TD DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: NODE source, omitted destination	7.2.2.3, 8.10.1.2	M (16: 255)		M (16: 255)	
2	TD-parameter	8.10.1.3	M		M	

13.10.4.11 Specific fields and subfields of the RQ DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: NODE.0 destination and source	7.2.2.3, 8.11.1.2	M (4,16: 255).0		M (4,16: 255).0	
2	RQ-parameter	8.11.1.3	M		M	

13.10.4.12 Specific fields and subfields of the RR DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: NODE.0 destination and source	7.2.2.3, 8.12.1.2	M (4,16: 255).0		M (4,16: 255).0	
2	RR-parameter	8.12.1.3, 8.12.2	M		M	

13.10.4.13 Specific fields and subfields of the PN DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	DL address format: NODE destination, omitted source	7.2.1.1, 8.13.1.2	M (16: 255)		M (16: 255)	
2	PN-parameters	8.13.1.3, 8.13.2				

13.10.4.14 Specific fields and subfields of the PR DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
1	permitted octets of SPDU in DLPDU	7.2.4, 8.14.1.4	0 : 64 M		0 : 64 M	

13.10.4.15 Specific fields and subfields of the PT DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	priority designator	7.2.1.2,				
1.1	URGENT	8.15.1.1	M		M	
1.2	NORMAL		M		M	
1.3	TIME-AVAILABLE		M		M	
2	DL address format: NODE destination, omitted source	7.2.1.1, 8.15.1.2	M (16: 255)		M (16: 255)	
3	DD-parameters	8.15.1.3, 8.15.2	M		M	

13.10.4.16 Specific fields and subfields of the ES DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	address size designator	7.2.1.1,				
1.1	LONG	8.17.1.1	C1		C1	
1.2	SHORT		M		M	
2	restart/continue designator	7.2.1.3, 8.17.1.1				
2.1	RESTART		M		M	
2.2	CONTINUE		M		M	
3	DL address formats:	7.2.2,				
3.1	format 1L — LONG destination, omitted source	8.17.1.1, 8.17.1.2	M		M	
3.2	format 1S — SHORT destination, omitted source		M		M	
4	DD-parameters	8.17.1.3	M		M	
C1 : if 13.9.1/1 then M else X						

13.10.4.17 Specific fields and subfields of the RT DLPDUs

The RT DLPDU has no variable fields or subfields.

13.10.4.18 Specific fields and subfields of the RI DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	DD-parameters	8.18.1.3	M		M	

13.10.4.19 Specific fields and subfields of the CL DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: omitted destination, NODE source	7.2.2.3, 8.19.1.2	M (16: 255)		M (16: 255)	

13.10.4.20 Specific fields and subfields of the TL DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: NODE destination, omitted source	7.2.2.3, 8.20.1.2	M (16: 255)		M (16: 255)	
Item number	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
2	permitted octets of SPDU in DLPDU	7.2.4, 8.20.1.4	0 : 64 M		0 : 64 M	

13.10.4.21 Specific fields and subfields of the WK DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status and range	Support	Status and range	Support
1	DL address format: NODE destination, omitted source	7.2.2.3, 8.21.1.2	M (16: 255)		M (16: 255)	

13.10.4.22 Specific fields and subfields of the Idle DLPDU

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Range (in octets)	Supported range	Range (in octets)	Supported range
1	permitted octets of user data in DLPDU	7.2.4, 8.22.1.4	0 : 64 O		0 : 64 O	

13.11 Major high-level capabilities

This subclause identifies many of the major conformance requirements of the implementation. The conformance requirements specified in this subclause are those related to the higher-level medium-access-independent functions of the DL-protocol. Subclause 13.10 specifies the conformance requirements for the lower-level functions of the protocol — those which are related to the underlying medium-access functions.

13.11.1 DLCEP classes implemented

The fieldbus Data Link layer protocol defines four major classes of DLCEP operation, each a superset of the prior class. The support requirements for these classes differ.

DLCEP classes implemented					
Item number	DLCEP class	DLC profiles	Reference	Status	Support
1	UNORDERED peer, publisher and subscriber DLCs	A, B, C, D	10.2, 13.9.2.2	M	
2	ORDERED peer, publisher and subscriber DLCs	A, B, C	10.2, 13.9.2.2	O	
3	CLASSICAL peer DLCs	A, B	10.2, 13.9.2.2	C1	
4	CLASSICAL publisher and subscriber DLCs	A	10.2, 13.9.2.2	C2	
C1 : support of this class requires support of ORDERED peer, publisher and subscriber DLCEPs					
C2 : support of this class requires support of CLASSICAL peer DLCEPs					

13.11.2 Parameters of the EC, DC, RC, CA, CD, ED and DT DLPDUs

This subclause enumerates the ranges and states the static conformance requirements for each variable parameter of each EC, DC, RC, CA, CD, ED and DT DLPDU. Conformance is only required on those DLPDUs which were claimed under 13.10.3.1 and 13.10.3.2, and then only for the direction(s) — receiving and recognizing, or forming and sending, or both — which were claimed. Thus any conditions or options specified in the relevant entry or entries in 13.10.3.1 and 13.10.3.2 apply uniformly to all of the DLPDU's specific requirements, as detailed in the following subclauses, for the corresponding direction of use. Requirements in the following DLPDU-specific subclauses for directions of use not claimed under either 13.10.3.1 or 13.10.3.2 can be ignored.

13.11.2.1 Communications paradigms implemented

The fieldbus Data Link Layer protocol specifies a number of communications paradigms. The support requirements for many of these paradigms differ.

Communications paradigms implemented				
Item number	Communications paradigm	Reference	Status	Support
1	Unacknowledged Connectionless	6.5.2, 10.3	M	
2	Acknowledged Connectionless	6.5.2, 10.3	C1	
3	Connectionless Exchange	6.5.2, 10.3	C2	
4	Unordered Connection as Peer	6.5.3, 10.2,13.9.2.2	M	
5	Ordered Connection as Peer	6.5.3, 10.2,13.9.2.2	C3	
6	Classical Connection as Peer	6.5.3, 10.2,13.9.2.2	C4	
7	Disordered Connection as Peer	6.5.3, 10.2,13.9.2.2	C6	
8	Unordered Connection as Publisher	6.5.3, 10.2,13.9.2.2	M	
9	Ordered Connection as Publisher	6.5.3, 10.2,13.9.2.2	C3	
10	Classical Connection as Publisher	6.5.3, 10.2,13.9.2.2	C5	
11	Disordered Connection as Publisher	6.5.3, 10.2,13.9.2.2	C7	
12	Unordered Connection as Subscriber	6.5.3, 10.2,13.9.2.2	M	
13	Ordered Connection as Subscriber	6.5.3, 10.2,13.9.2.2	C3	
14	Classical Connection as Subscriber	6.5.3, 10.2,13.9.2.2	C5	
15	Disordered Connection as Subscriber	6.5.3, 10.2,13.9.2.2	C7	
C1 : if 13.9.2.7/3 then X else M. C2 : if 13.9.2.7/1 then M else X. C3 : if DLC_conformance_class = {A, B, or C} then M else O. C4 : if DLC_conformance_class = {A or B} then M else O. C5 : if DLC_conformance_class = A then M else O. C6 : if C4 then O else X. C7 : if C5 and C6 then M else X.				

13.11.2.2 Support for implied QoS for data link connections (DLCs)

Some aspects of data link connections also impact on conformance requirements.

Support for implied QoS (Quality of Service) parameters for data link connections							
Item number	Column letter →			A) Receiving data		B) Sending data	
	QoS parameter	Reference	Allowed range	Status	Support	Status	Support
1	segmentation — what is the ratio of the maximum DLSDU size to the maximum DLPDU size?	13.9.2.2	1 : 16	C1		C1	
2	what is the maximum window size on CLASSICAL DLCs?	9.1 Table 35	1 : 15	C2		C2	
C1 : if DLC_conformance_class = {A, B, or C} then 1:16 M else 1 M; the same value must be specified for both columns. C2 : if DLC_conformance_class = {A or B} then M else O; the same value must be specified for both columns.							

13.11.2.3 Parameters of the EC DLPDU – protocol version 1

13.11.2.3.1 DLC basic attributes

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	DLC protocol version number	9.1 a4	1 M		1 M	
2	reply requested	9.1 a1	F, T M		F, T M	
3	publisher-DLCEP-address reuse-discriminator	9.1 a2	F, T M		F, T M	
4	DLL path diversity	9.1 a3				
4.1	ANY-PATH		M		M	
4.2	THIS-PATH		M		M	
5	DLCEP class	9.1 a5				
5.1	peer DLCEP		M		M	
5.2	publisher DLCEP		M		M	
5.3	subscriber DLCEP		M		M	
6	DLL priority	9.1 a6				
6.1	URGENT		M		O	
6.2	NORMAL		M		O	
6.3	TIME-AVAILABLE		M		M	
7	DL address size	9.1 a7				
7.1	LONG		M		M	
7.2	SHORT		M		O	
7.3	VERY-SHORT		M		O	
8	DLPDU authentication	9.1 a8				
8.1	MAXIMAL		M		M	
8.2	SOURCE		M		M	
8.3	ORDINARY		M		M	
9	maximum DL-CONNECT, DL-RESET and DL-SUBSCRIBER-QUERY confirm delay	9.1 b				
9.1	UNLIMITED (encoded as 0xFFFF)		65 535 M		65 535 M	
9.2	minimum delay required (in ms)		1 : 60 000		1 : 60 000	
9.3	maximum delay supported (in ms)		1 : 60 000 M		1 : 60 000 M	
10	maximum DL-DATA confirm delay	9.1 b				
10.1	UNLIMITED (encoded as 0xFFFF)		65 535 M		65 535 M	
10.2	minimum delay required (in ms)		1 : 60 000		1 : 60 000	
10.3	maximum delay supported (in ms)		1 : 60 000 M		1 : 60 000 M	

13.11.2.3.2 DLC attributes as sender

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	DLC data delivery features	9.1 c1				
1.1	UNORDERED — peer, publisher and subscriber		M		M	
1.2	ORDERED — peer, publisher and subscriber		M		C1	
1.3	CLASSICAL — peer		M		C2	
1.4	DISORDERED — peer		M		O	
1.5	CLASSICAL — publisher and subscriber		M		C3	
1.6	DISORDERED — publisher and subscriber		M		C4	
2	DLC residual activity required	9.1 c2				
2.1	FALSE (no background activity)		M		M	
2.2	TRUE (background activity)		M		M	
3	DLSDU storage model at DLCEP	9.1 c3				
3.1	QUEUE		M		M	
3.2	BUFFER		M		M	
4	DLC maximum window size	9.1 c4	0:15 M		0:15 M1	
5	SD-parameter basic-DLC-parameters format	9.1 c5				
5.1	A – NULL format		M		C1	
5.2	B – SHORT format		M		C1	
5.3	C – LONG format		M		C1	
5.4	D – UNSEGMENTED LONG format		M		C1	
5.5	E – SUBSCRIBER format		M		C1	
5.6	F – PUBLISHER format		M		C1	
5.7	G – UNSEGMENTED PUBLISHER format		M		C1	
6	2-way data exchange permitted	9.1 c6				
6.1	FALSE		M		M	
6.2	TRUE		M		O	
7	timeliness included in DLPDU	9.1 c8				
7.1	FALSE		M		M	
7.2	TRUE		M		C5	
8	SD-parameter time-stamp format	9.1 c9				
8.1	J – NULL format		M		M	
8.2	K – two-octet format		M		C6	
8.3	L – three-octet format		M		C6	
8.4	M – six-octet format		M		C6	
9	DLC maximum DLSDU size	9.1 c10	0:4096 M		0:4096 M2	
C1 : if DLC_conformance_class = {A, B, or C} then M else O.						
C2 : if DLC_conformance_class = {A or B} then M else O.						
C3 : if DLC_conformance_class = A then M else O.						
C4 : if C3 and disordered_peer then M else O.		disordered_peer = /1.4				
C5 : if DLC_conformance_class = {A or B} then M else O.						
C6 : if DLC_conformance_class = A then M else O.						
M1: must be 1:N, where N ≥ the value claimed in 13.11.2.2/2						
M2 : must be 1:N, where N = 256 times the value claimed in 13.11.2.2/1						

13.11.2.3.3 DLC attributes as receiver

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	DLC data delivery features	9.1 c1				
1.1	UNORDERED — peer, publisher and subscriber		M		M	
1.2	ORDERED — peer, publisher and subscriber		M		C1	
1.3	CLASSICAL — peer		M		C2	
1.4	DISORDERED — peer		M		O	
1.5	CLASSICAL — publisher and subscriber		M		C3	
1.6	DISORDERED — publisher and subscriber		M		C4	
2	DLC residual activity required	9.1 c2				
2.1	FALSE (no background activity)		M		M	
2.2	TRUE (background activity)		M		M	
3	DLSDU storage model at DLCEP	9.1 c3				
3.1	QUEUE		M		M	
3.2	BUFFER		M		M	
4	DLC maximum window size	9.1 c4	0:15 M		0:15 M1	
5	SD-parameter basic-DLC-parameters format	9.1 c5				
5.1	A – NULL format		M		C1	
5.2	B – SHORT format		M		C1	
5.3	C – LONG format		M		C1	
5.4	D – UNSEGMENTED LONG format		M		C1	
5.5	E – SUBSCRIBER format		M		C1	
5.6	F – PUBLISHER format		M		C1	
5.7	G – UNSEGMENTED PUBLISHER format		M		C1	
6	2-way data exchange permitted	9.1 c6				
6.1	FALSE		M		M	
6.2	TRUE		M		O	
7	timeliness included in DLPDU	9.1 c8				
7.1	FALSE		M		M	
7.2	TRUE		M		C5	
8	SD-parameter time-stamp format	9.1 c9				
8.1	J – NULL format		M		M	
8.2	K – two-octet format		M		C6	
8.3	L – three-octet format		M		C6	
8.4	M – six-octet format		M		C6	
9	DLC maximum DLSDU size	9.1 c10	0:4096 M		0:4096 M2	
C1 : if DLC_conformance_class = {A, B, or C} then M else O.						
C2 : if DLC_conformance_class = {A or B} then M else O.						
C3 : if DLC_conformance_class = A then M else O.						
C4 : if C3 and disordered_peer then M else O.		disordered_peer = /1.4				
C5 : if DLC_conformance_class = {A or B} then M else O.						
C6 : if DLC_conformance_class = A then M else O.						
M1 : must be 1:N, where N ≥ the value claimed in 13.11.2.2/2						
M2 : must be 1:N, where N = 256 times the value claimed in 13.11.2.2/1						

13.11.2.4 Parameters of the DC DLPDU – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	DLC protocol version number	9.2 a3	1 M		1 M	
2	reply requested	9.2 a1	F, T M		F, T M	
3	reason for requested action	9.2 b	M		M	

13.11.2.5 Parameters of the RC DLPDU – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	DLC protocol version number	9.3 a, 9.3.a3	1 M		1 M	
2	reply requested	9.3 a, 9.3.a1	F, T M		F, T M	
3	reason for requested action	9.3 b	M		M	
4	sequence number residue	9.3 c	M		M	

13.11.2.6 SD-parameters of the CA, CD, DT and ED DLPDUs – protocol version 1

13.11.2.6.1 Unitdata transaction-support parameters

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	SD-parameter connectionless-mode parameters formats	9.4.1				
1.1	format P — NULL	9.4.1 a	M		M	
1.2	format R — INITIATOR	9.4.1 b	M		C1	
1.3	format U — RESPONDER	9.4.1 c	C1		M	
C1 if 13.9.2.7/3 then X else M						

13.11.2.6.2 Basic DLC parameters

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	SD-parameter basic-DLC-parameters formats	9.4.2.1				
1.1	format A — NULL	9.4.2.1 A	M		M	
1.2	format B — SHORT	9.4.2.1 B	C1		C1	
1.2.1	retransmit	9.4.2.1 B1	F, TC1		F, TC1	
1.2.2	acknowledge	9.4.2.1 B2	F, TC1		F, TC1	
1.2.3	timeliness	9.4.2.1 B3	F, TC1		F, TC1	
1.2.4	modulus number of post-acknowledged or requested DLSDU	9.4.2.1 B4	C1		C1	
1.2.5	modulus number of associated DLSDU or of highest-numbered DLSDU sent	9.4.2.1 B5	C1		C1	
1.3	format C — LONG	9.4.2.1 C	C1		C1	
1.3.1	modulus number of post-acknowledged or requested DLSDU	9.4.2.1 C1	C1		C1	
1.3.2	(zero-origin) requested segment number	9.4.2.1 C2	0:N C1		C1	
1.3.3	retransmit	9.4.2.1 C3	F, TC1		F, TC1	
1.3.4	acknowledge	9.4.2.1 C4	F, TC1		F, TC1	
1.3.5	timeliness	9.4.2.1 C5	F, TC1		F, TC1	
1.3.6	modulus number of associated DLSDU or of highest-numbered DLSDU sent	9.4.2.1 C6	C1		C1	
1.3.7	(zero-origin) total number of segments in DLSDU	9.4.2.1 C7	0:N C1		C1	
1.3.8	(zero-origin) associated segment number	9.4.2.1 C8	0:N C1		C1	
1.4	format D — UNSEGMENTED LONG	9.4.2.1 D	C1		C1	
1.5	format E — SUBSCRIBER	9.4.2.1 E	C1		C1	
1.6	format F — PUBLISHER	9.4.2.1 F	C1		C1	
1.7	format G — UNSEGMENTED PUBLISHER	9.4.2.1 G	C1		C1	
C1 : if DLC_conformance_class = {A, B, or C} then M else O.						
N = : (the value claimed in 13.11.2.2/2) minus 1						

13.11.2.6.3 Timeliness parameters

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	SD-parameter time-stamp format	9.4.2.2				
1.1	format J – no-time format	9.4.2.2 J	M		M	
1.2	format K – 2-second format	9.4.2.2 K	M		C1	
1.3	format L – 8-minute format	9.4.2.2 L	M		C1	
1.4	format M – full-time format	9.4.2.2 M	M		C1	
C1 if 13.9.2.5/2 and 13.9.2.6/2 then M else O						

13.11.3 Parameters of the SR DLPDU– protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	SR-parameter formats	9.5				
1.1	format X — NULL	9.5 a	M		C1	
1.2	format Y — ERROR	9.5 b	M		C1	
C1 if potential_LAS then M else O			potential_LAS = 13.10.1/2 or /3			

13.11.4 Parameters of the TD DLPDU – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	link originating DL-time	9.6 a	M		C2	
2	DL-time-quality measures	9.6 b	M		C2	
2.1	limiting time-synchronism class	9.6 b1	0:7 M		0:N C2	
2.2	number of intervening links	9.6 b2	0:7 M		0:7 C2	
2.3	source type	9.6 b3	0:3 M		0:3 C2	
3	DL-time offset	9.6 c	C1		C2	
4	DL-time prior to end-of-transmission	9.6 d	C1		C2	
5	DL-time adjustment	9.6 e	C1		C2	
C1	if time_synchronism_supported then M else O		time_synchronism_supported = 13.9.3.1/2 or /3 or /4 or /5 or /6 or /7 or /8			
C2	if potential_LAS then M else O		potential_LAS = 13.10.1/2 or /3			
N = if 13.9.3.1/1 then 0 elseif 13.9.3.1/2 then 1 elseif 13.9.3.1/3 then 2 elseif 13.9.3.1/4 then 3 elseif 13.9.3.1/5 then 4 elseif 13.9.3.1/6 then 5 elseif 13.9.3.1/7 then 6 else 7						

13.11.5 Parameters of the RQ DLPDU – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	sending time-stamp	9.7 a	M		C1	
C1 if potential_LAS then M else O			potential_LAS = 13.10.1/2 or /3			

13.11.6 Parameters of the RR DLPDU – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	DL-time-quality measures	9.8 a	C1		M	
1.1	limiting time-synchronism class	9.8.a1	0:7 M		0:N C2	
1.2	support for estimated C(NT)	9.8.a2	C2		M	
2	RQ sending time-stamp	9.8 b	C1		M	
3	RQ receiving time-stamp	9.8 b	C1		M	
4	RR sending time-stamp	9.8 c	C1		M	
C1 if potential_LAS then M else O			potential_LAS = 13.10.1/2 or /3			
C2 if potential_LAS or N≠0 then M else O						
N = if 13.9.3.1/1 then 0 elseif 13.9.3.1/2 then 1 elseif 13.9.3.1/3 then 2 elseif 13.9.3.1/4 then 3 elseif 13.9.3.1/5 then 4 elseif 13.9.3.1/6 then 5 elseif 13.9.3.1/7 then 6 else 7						

13.11.7 Parameters of the PN DLPDU – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status or range	Support	Status or range	Support
1	DLC protocol version number	9.9 a3	1 M		1 M	
2	PhL inter-channel signal skew	9.9 a1	0:7 M		0:7 C1	
3	PhL post-transmission-gap extension units	9.9 a4	0:7 M		0:7 C1	
4	PhL preamble extension units	9.9 a5	0:7 M		0:7 C1	
5	slot-time	9.9 b1	1:4095 M		1:4095 C1	
6	maximum-response-delay	9.9 b2	1:12 M		1:12 C1	
7	minimum-inter-PDU-delay	9.9 b3	0:120 M		0:120 C1	
C1 if potential_LAS then M else O			potential_LAS = 13.10.1/2 or /3			

13.11.8 DD parameters of the PT, ES and RI DLPDUs – protocol version 1

Supported parameters						
Item number	Column letter →		A) Receiving and recognizing		B) Forming and sending	
	Parameter	Reference	Status	Support	Status	Support
1	delegated or requested duration	9.10 a	0:65000 M		0:65000 M	

14 Type 2: Overview of the DL-protocol

14.1 General

14.1.1 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 14.1.2) 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 134 :

Table 134 – 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 DLSUs 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 25. The arrowheads illustrate the primary direction of flow of data and control.

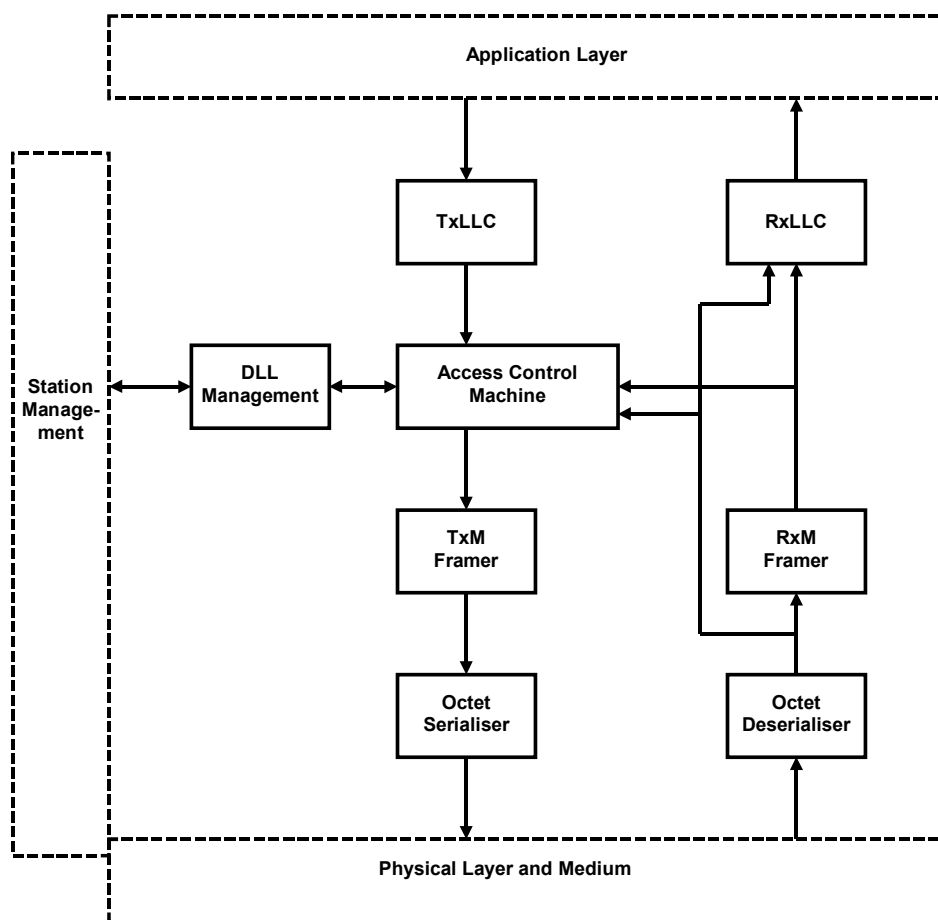


Figure 25 – Data Link Layer internal architecture

14.1.2 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.

14.1.3 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.

14.2 Services provided by the DL

14.2.1 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.

14.2.2 QoS

14.2.2.1 Definition

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

14.2.2.2 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.

14.2.2.3 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.

14.2.2.4 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.

14.3 Structure and definition of DL-addresses

14.3.1 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.

14.3.2 MAC ID address

MAC ID addresses identify physical nodes on the local link as shown in Figure 26.

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



Figure 26 – Basic Structure of a MAC ID address

Table 135 specifies the use of MAC ID addresses.

Table 135 – 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
0x 64 to 0xFE	reserved
0xFF	broadcast address to all MAC ID nodes on this link

14.3.3 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 27.

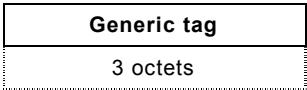


Figure 27 – Basic Structure of a Generic tag address

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.

14.3.4 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 28.

Fixed tag service	Destination MAC ID
1 octet	1 octet

Figure 28 – Basic Structure of a Fixed tag address

The first octet shall specify the service access point in the destination node as specified in Table 136. 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 136 – 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 - 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.

14.4 Services assumed from the PhL

14.4.1 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 137), 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.

14.4.2 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 137. The M_0 and M_1 will be encoded into Ph_Symbols that represent Manchester biphasic 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 29 to provide an example of the data encoding rules shown in Table 137.

Table 137 – 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

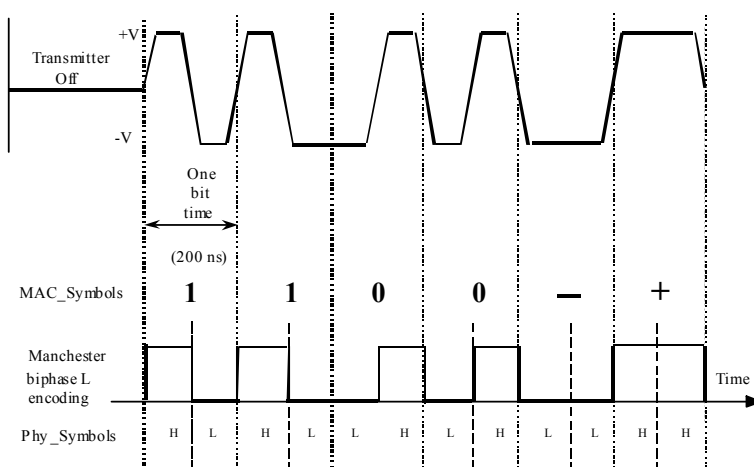


Figure 29 – M_symbols and Manchester encoding at 5 MHz (informative)

14.4.3 DLL to PhL interface

14.4.3.1 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.

14.4.3.2 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.

14.4.3.3 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.

14.4.3.4 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.

14.4.3.5 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 137. Valid M_symbols shall be M_0 {0}, M_1 {1}, M_ND+ {+}, and M_ND- {-} as shown in Table 138.

Table 138 – 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.

14.4.3.6 Ph_status_indication

ph_status_indication shall represent the delimiter status of the frame which was received from the PhL as shown in Table 139. 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 139 – 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

14.4.3.7 `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 138. `ph_data_request` shall indicate M_0 when no data is to be transmitted (and `ph_frame_request = false`).

14.4.3.8 `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.

14.5 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.

15 Type 2: General structure and encoding of PhIDUs and DLPDUs and related elements of procedure

15.1 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.

15.2 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 18.2.

Each NUT is divided into three major parts: scheduled, unscheduled, and guardband as shown in Figure 30. 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.

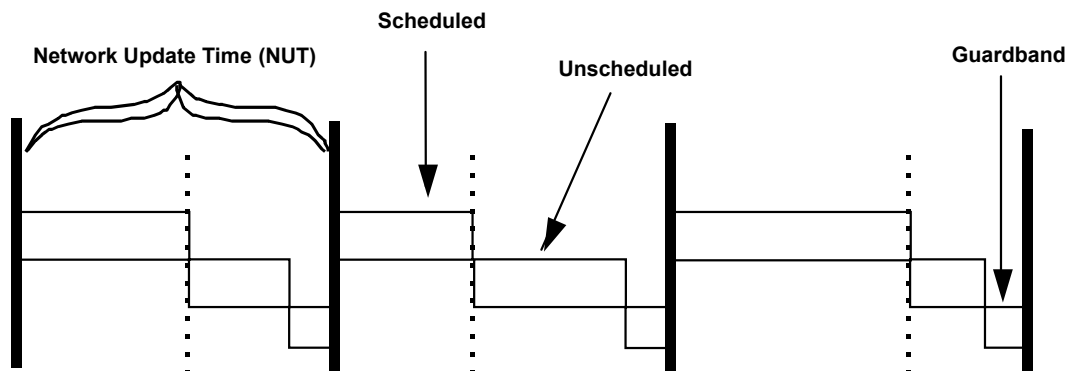


Figure 30 – NUT structure

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 31 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.

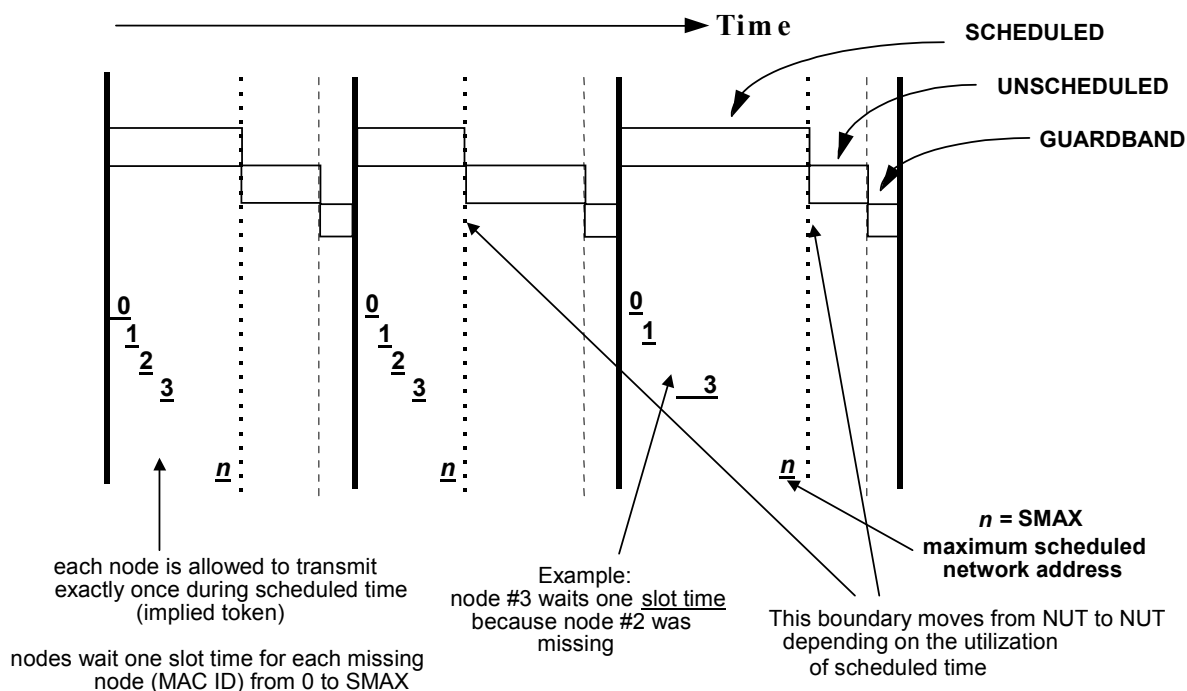


Figure 31 – Media access during scheduled time

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 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 32 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.

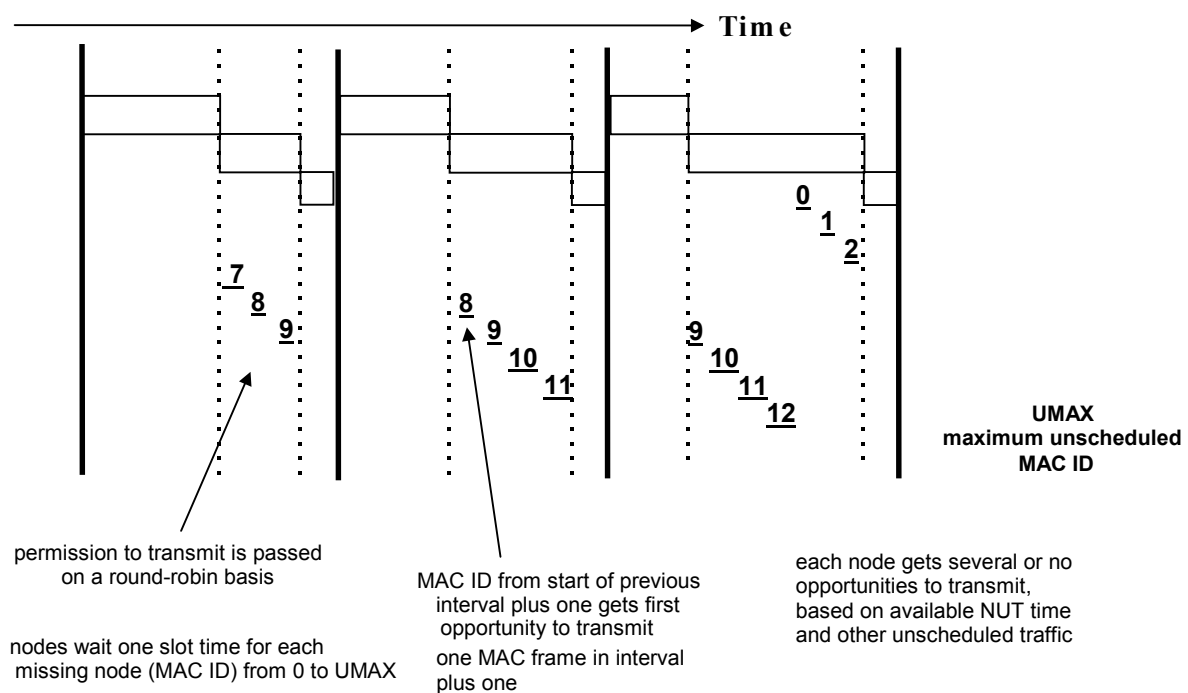


Figure 32 – Media access during unscheduled time

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.

15.3 DLPDU structure and encoding

15.3.1 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.

15.3.2 DLPDU components

The general DLPDU structure is shown in Figure 33 and managed by the DLL support units TxFramer, RxFramer, Octet Serializer and Octet Deserializer as shown in Figure 25.

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.

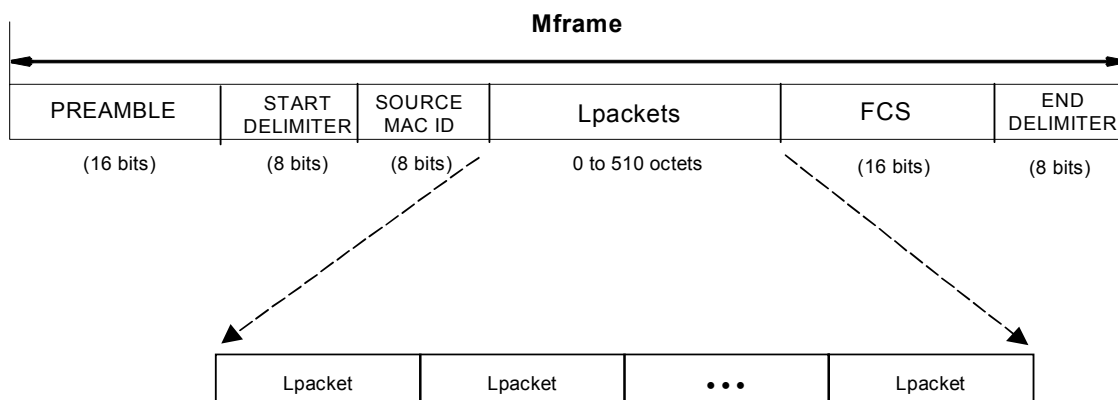


Figure 33 – DLPDU format

15.3.3 Preamble

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

15.3.4 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.

15.3.5 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).

15.3.6 Source MAC ID

The source MAC ID address shall be a single octet as specified in 14.3.2. A node may also temporarily assume a MAC ID of zero to perform link maintenance.

15.3.7 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.

15.3.8 Frame check sequence (FCS)

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

NOTE The generation of the FCS is detailed in 5.1.1.

15.3.9 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.

15.3.10 Abort DLPDU

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

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

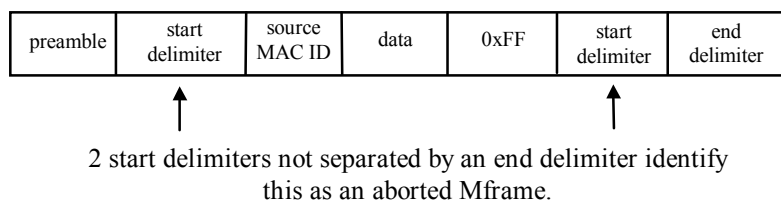


Figure 34 – Aborting a DLPDU during transmission

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 34.

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.

15.4 Lpacket components

15.4.1 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 35.

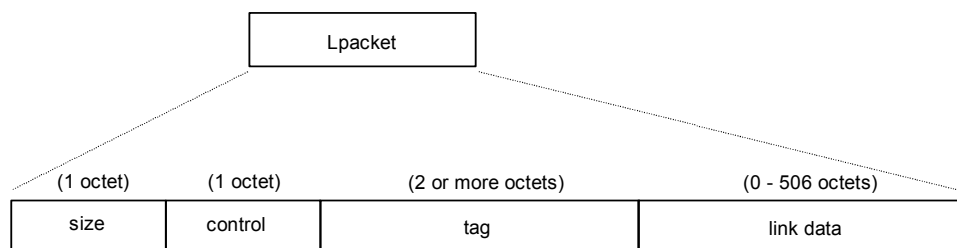


Figure 35 – Lpacket format

15.4.2 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.

15.4.3 Control

15.4.3.1 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 14.3.4).
- When clear (bit 0 = 0), the Lpacket shall be a generic tag Lpacket (see 14.3.3).

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.

15.4.3.2 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.

15.4.3.3 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.

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

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

15.4.4 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 36. The 3 octet generic tag address (see 14.3.3) shall identify the connected mode information carried in the link data field

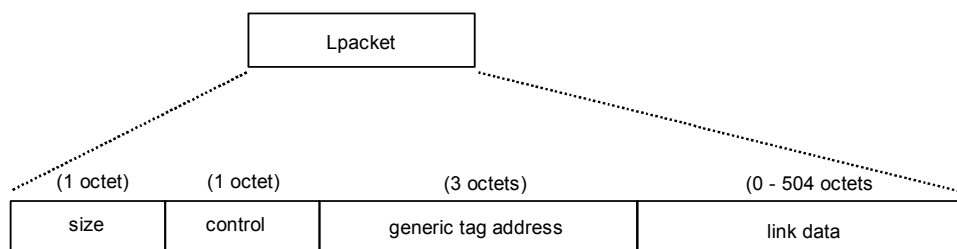


Figure 36 – Generic tag Lpacket format

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.

15.4.5 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 37. The 2 octet fixed tag address (see 14.3.4) shall identify the service access point in the the destination node and the address of the destination node.

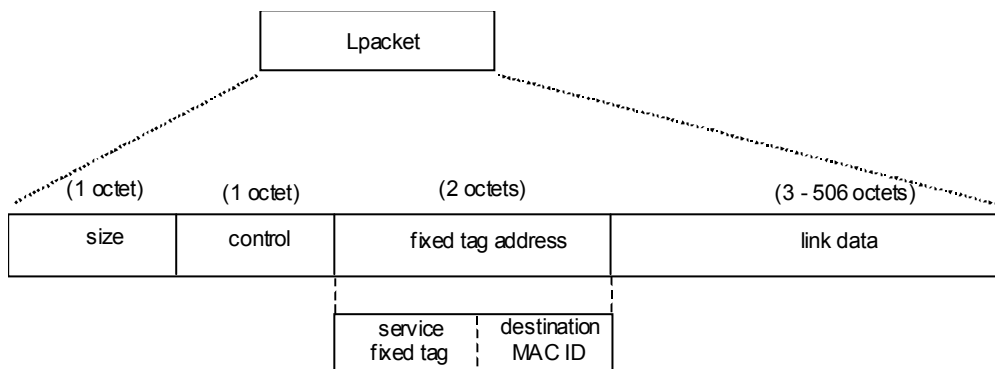


Figure 37 – Fixed tag Lpacket format

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.

15.5 DLPDU procedures

15.5.1 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 19.3).

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.

15.5.2 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.

15.5.3 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.

15.5.4 Receiving DLPDUs

15.5.4.1 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.

15.5.4.2 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.

15.6 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 140. 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 140 – DLL support services and objects

DLL item	Description	Location
DLS user services		see IEC 61158-3
connected mode transfer service	a local service for requesting, indicating and confirming generic tag (connected mode) data transfer by Lpackets	see 16.2.2
connectionless mode transfer service	a local service for requesting, indicating and confirming generic tag (connected mode) data transfer by Lpackets	see 16.2.2
queue maintenance service	a local service for flushing unsent Lpacket messages from the internal DL service queue.	see 16.2.3
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 16.2.4
link synchronization service	a local service for informing the local DLS user of the current NUT number to maintain schedule pointers	see 16.2.5
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 16.2.6
event reports service	local services to report various events for use by management	see 16.2.7
bad FCS service		see 16.2.8
current moderator		see 16.2.9
power-up and on-line		see 16.2.10
enable moderator		see 16.2.11
listen only		see 16.2.12
Generic tag service	an Lpacket service for connected mode communication	see 16.3
Fixed tag services	a range of Lpacket service addresses for various connectionless mode communication.	see 15.4.5
moderator	an Lpacket broadcast in every NUT for managing time precision and timing for change of link parameters	see 16.4
time distribution service	an Lpacket for publishing time values and offsets used in coordinating the same time sense in all stations .	see 16.5
UCMM	an Lpacket service to transfer DLSDUs from a DLS user to the unconnected message manager in a peer DLS-user.	see 16.6
Keeper UCMM	an Lpacket service to transfer DLSDUs from a DLS user to the Keeper objects in other nodes	see 16.7
TUI	an Lpacket broadcast to all stations assisting their synchronization to the current link configuration	see 16.8
link parameters	an Lpacket service for publishing pending link parameters in advance of a synchronized change of link operating conditions	see 16.9
tMinus	an Lpacket service to start a count down leading to a change of link parameters and return the result.	see 16.9
I'm alive	an Lpacket broadcast by new and reset devices to assist their entry into an operating system	see 16.10
ping request and reply	an Lpacket service soliciting identification responses from all other active devices on the link	see 16.11
WAMI	an Lpacket allowing temporary devices to identify a local device MAC ID when they are plugged into its network access port.	see 16.12
debug	an Lpacket service to trace object state transitions	see 16.13

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 17.2
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 17.3
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 17.4
TCP/IP interface object	provides the mechanism to configure the TCP/IP interface of a device.	see 17.5
Ethernet link object	maintains link-specific counters and status information for a physical Ethernet ISO/IEC 8802-3 port.	see 17.6

16 Type 2: Specific DLPDU structure, encoding and procedures

16.1 Modeling language

16.1.1 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

16.1.2 Use of DLL- prefix

The primitives and parameters described in IEC 61158-3, Type 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.

16.1.3 Data types

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

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 141. Additional detail is specified in IEC 61158-6.

Table 141 – Elementary data types

Keyword	Description	Range	
		Minimum	Maximum
BOOL	Boolean		
SINT	Short Integer	-128	127
INT	Integer	-32768	32767
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	65535
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)		
OCTET / OCTET	bit string - 8 bits		
WORD	bit string - 16-bits		
DWORD	bit string - 32-bits		
LWORD	bit string - 64-bits		

16.2 DLS user services

16.2.1 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.

16.2.2 Connected mode and connectionless mode transfer service

16.2.2.1 Service encoding

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

a) Connection mode transfers use Generic tag request and confirmation.

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

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

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

c) Connection mode transfers use Generic tag indication.

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

d) Connectionless mode transfers use Fixed tag indication.

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

16.2.2.2 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 15.3 and 15.4).

16.2.2.3 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_rcv_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_rcv_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.

16.2.3 Queue maintenance service

16.2.3.1 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 );
```

16.2.3.2 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.

16.2.4 Tag filter service

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

16.2.4.2 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.

16.2.5 Link synchronization service

16.2.5.1 Service encoding

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

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

16.2.5.2 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.

16.2.6 Synchronized parameter change service

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

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
};                          // may not transmit passed this limit
```

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

```
DLL-tMinus-start-countdown-request(      DLL-tMinus-start-countdown-confirm(
    IDENTIFIER id,                        IDENTIFIER id,
    USINT      start_count );              BOOL      result );
```

NOTE The tMinus fixed tag service (see 16.9) 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( );
```

16.2.6.2 Procedures for request, confirmation and indication

The IDENTIFIER, id, shall correlate a confirmation with a corresponding request. The result shall return TRUE if the request was successful and FALSE if unsuccessful.

The moderator Lpacket contains a field, called tMinus, that shall be used to synchronize the update of the current link parameters. The DLL-tMinus-start-countdown-request shall cause a node to participate in a tMinus countdown, and, if the node is the moderator, shall initialize the tMinus field of the moderator Lpacket. The moderator node shall decrement this field before transmitting each moderator until the field equals zero. When the tMinus field transitions from 1 to 0, the DLL in each node participating in the countdown shall locally generate a DLL_tminus_zero_indication and copy its pending link parameters into its current copy. If the tMinus field transitions to 0 from any value except 1, the countdown shall be aborted and no DLL_tminus_zero_indication shall be generated.

16.2.7 Event reports service

The indication used to alert the Station Management entity of various events internal to the Data Link Layer shall be of the form:

```
DLL_event_indication(
    DLL_event event,
    USINT      mac_id /*[optional]*/ );
```

The DLL_event, event, shall be one of the events enumerated in Table 142. The mac_id parameter shall be used in the case of (event = DLL_EV_badFrame); it shall indicate the source MAC ID of the node that transmitted the bad DLPDU.

NOTE Since the DLPDU was damaged, the source MAC ID may not be correct.

Table 142 – DLL events

DLL events	Description
DLL_EV_rxGoodFrame	A good DLPDU was received. This shall include DLPDUs that contain no data (null DLPDUs), but shall exclude moderators.
DLL_EV_txGoodFrame	A good DLPDU was transmitted. This shall include DLPDUs that contain no data (null DLPDUs), but shall exclude moderators.
DLL_EV_badFrame	A damaged DLPDU was received. The optional parameter, which is the apparent source MAC ID of the offending node, shall also be reported.
DLL_EV_errA	A bad DLPDU was received on channel A of the physical medium, or a good DLPDU was received on channel B and ph_frame_indication from channel A stayed false.
DLL_EV_errB	A bad DLPDU was received on channel B of the physical medium, or a good DLPDU was received on channel A and ph_frame_indication from channel B stayed false.
DLL_EV_txAbort	A transmit DLPDU was terminated with an abort sequence.
DLL_EV_NUT_overflow	NUT is not large enough to accommodate all the scheduled traffic.
DLL_EV_dribble	Scheduled Lpackets could not be sent during scheduled time.
DLL_EV_nonconcurrency	An event was detected that indicates that this node is out of step with the access control protocol.
DLL_EV_rxAbort	A DLPDU was received that was terminated with an abort sequence.
DLL_EV_lonely	Have not heard a DLPDU from another node on the link for 8 NUTs
DLL_EV_dupNode	Another node on the link is using this node's MAC ID.
DLL_EV_noisePulse	ph_lock_indication went true then false before ph_frame_indication went true, but ph_lock_indication was not true long enough to indicate a possibly damaged DLPDU.
DLL_EV_collision	ph_frame_indication was true when this node was about to transmit.
DLL_EV_invalidModAddress	A moderator was received from a node that does not have the lowest MAC ID on the link.
DLL_EV_rogue	A moderator DLPDU was received that does not match the link configuration information at this node.
DLL_EV_deafness	Cannot hear the moderator DLPDU even though other link traffic is present.
DLL_EV_supernode	A moderator was received from MAC ID 0.

16.2.8 Bad FCS service

The indication used to alert the Station Management entity that a received DLPDU had an invalid FCS shall be of the form:

```
DLL_badFCS_indication( CHANNEL channel );
```

The channel parameter shall indicate which PhL entity provided the DLPDU which was in error. This parameter shall be one of CHA or CHB, corresponding to PhL channel A and PhL channel B, respectively. This indication shall be provided, at most, once per error DLPDU per channel.

16.2.9 Current moderator service

The indication used to inform the Station Management entity which node is currently the moderator shall be of the form:

```
DLL_currentMod_indication( USINT macID );
```

The macID parameter shall indicate the MAC ID of the node that last transmitted a valid moderator DLPDU.

16.2.10 Power up and Online services

16.2.10.1 Service encoding

The request and confirmation which places the Data Link Layer on-line shall be of the form:

```
DLL_online_request( BOOL online );           DLL_online_confirm( BOOL online );
```

The indication that specifies that the Data Link Layer has completed its initialization shall be of the form:

```
DLL_powerup_indication( void );
```

16.2.10.2 Procedures for request and confirmation

At power-up, the DLL shall wait until the online parameter equals TRUE. The DLL shall then begin the process of going on-line. When the online parameter is FALSE, the DLL shall go off-line at the end of the current NUT. When off-line the DLL shall not transmit, and shall ignore any link activity.

16.2.11 Enable moderator service

16.2.11.1 Service encoding

The request and confirmation services that enable and disable the ability of a node to assume the moderator role shall be of the form:

```
DLL_enable_moderator_request( BOOL enable );
DLL_enable_moderator_confirm( BOOL enable );
```

16.2.11.2 Procedures for request and confirmation

When the enable parameter is TRUE, the DLL shall enable the transmission of moderator DLPDUs. When the enable parameter is FALSE, transmission of moderator DLPDUs shall be disabled.

NOTE This request is used by station management entities to allow a new node to non-disruptively join a working link. The moderator switch over protocol does not tolerate the node with the lowest MAC ID suppressing moderator DLPDUs for an extended period. The Network Attachment Monitor (NAM) re-enables moderator transmission whenever another device is detected on the link.

16.2.12 Listen only service

16.2.12.1 Service encoding

The request and confirmation services that allows a device to receive, but disables the ability of a device to transmit, shall be of the form:

```
DLL_listen_only_request( BOOL enable );
DLL_listen_only_confirm( BOOL enable );
```

16.2.12.2 Procedures for request, confirmation and indication

When the enable parameter is TRUE, the Data Link Layer shall participate in the access protocol and transmit DLPDUs. When the enable parameter is FALSE, transmission of DLPDUs shall be disabled; however, the ability of the node to receive DLPDUs shall not be impaired.

16.3 Generic Tag Lpacket

16.3.1 General

The generic tag Lpacket is used for transferring connection data. The connection ID or Generic Tag is a unique identifier for previously negotiated resources and parameters at its source, destination(s) and any intermediate transit points. Only the generic Tag is necessary to identify the message data.

16.3.2 Structure of the Generic Tag Lpacket

The size and control fields shall follow the Generic Tag format. The address shall be the DLS-generic-tag provided by the local DL-generic-request service. The link data field shall contain the DLS-user-data provided by the DL-generic-request.

16.3.3 Sending and receiving the Generic Tag Lpacket

The sending station shall send the Lpacket at the QoS priority specified by the DL-generic-request.

The status of each generic tag Lpacket transmission is confirmed to the DL user by returning the DLS-user-id (an internal handle or identifier) and the DLS Txstatus with one of the following values.

- a) "OK" — message successfully sent;
- b) "TXABORT" — sending process failed;
- c) "FLUSHED" — message has been removed from the pending queue before sending.

NOTE 1 Txstatus is a local variable so the coding is not specified.

NOTE 2 The FLUSHED status is only used in response to a deletion requested by the Queue maintenance service

NOTE 3 The parameter value OK is not an indication that the message has been received

All receiving stations whose local Tag Filter service contains an address matching the received Lpacket address, shall pass each correctly received data unit to its local DL user, together with the data unit size and the address (the generic tag for the connection).

16.4 Moderator Lpacket

16.4.1 General

The moderator MAC Lpacket is used for DLL management, access timing and synchronization, it is sent as a moderator DLPDU containing only one Lpacket, the moderator Lpacket.

16.4.2 Structure of the moderator Lpacket

The size and control fields shall follow the fixed tag format. The address shall be formed from the moderator fixed tag and 0xFF the broadcast address for all MAC IC nodes on this link. The link data field shall contain the following in the listed order.

```

class moderatorLpacket : public fixedLpacket
{
public:
    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;       // 1 µs increments allowed for line turnaround
    USINT    blanking;        // 1,6 µs increments to disable RX after DLPDU
    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    usr;            // unscheduled start register
    USINT    interval_count;  // current NUT number (0 to modulus)
    USINT    modulus;         // modulus of the interval counter
    USINT    tMinus;          // countdown to synchronized link parameter change
    USINT    gb_prestart;     // transmit cut-off, 10 µs intervals before tone
    USINT    reserved;        //
};

```

16.4.3 Sending and receiving the moderator Lpacket

The moderator DLPDU containing the moderator Lpacket shall be transmitted once per NUT by the node with the lowest MAC ID. It shall be sent in the guardband, a fixed part of every NUT reserved for the moderator DLPDU.

To support DLL management actions, the sender and receivers of the moderator DLPDU shall use the contents of the moderator link data field as follows.

The `NUT_length` shall specify the duration of the current NUT. `NUT_length` shall be a 16 bit integer representing the number of 10 µs ticks and shall be in the range 50 (500 µs) to 10000 (100 ms). The beginning of the NUT shall be called the tone. At the tone, a counter that decrements every 10 µs shall be loaded with the `NUT_length`.

NOTE 1 Subclause 18.2 further constrains the value of `NUT_length`.

To guarantee that the link is quiet during moderator transmission, all nodes shall cease transmitting when their internal counter reaches the value of `gb_prestart`. All nodes may then assume that the link is quiet between `gb_start` and `gb_center`. When the moderator node's counter reaches the value of `gb_center`, it shall begin transmitting a DLPDU containing only the moderator Lpacket. The moderator Lpacket shall be completely transmitted at least 30 µs before the decrementing counter would have expired.

NOTE 2 Subclause 18.2 calculates the values of `gb_prestart`, `gb_start`, and `gb_center` to guarantee these constraints. Factors that affect the calculation of these values include clock accuracy, missing nodes, and propagation delay between nodes. In all cases, `gb_prestart` > `gb_start` > `gb_center` > 7.

The highest MAC ID transmitting during the scheduled portion of the NUT shall be `smax`. It shall be in the range 0 to 254. `umax` shall determine the highest MAC ID that transmits during the unscheduled portion of the NUT. It shall be in the range `smax` to 254. The unscheduled start register, `usr`, shall select which node transmits first in the unscheduled portion of the NUT. The `usr` shall increment each NUT. This counter shall be updated modulo (`umax` + 1).

A node shall listen to the PhL directly after it finishes transmitting. `blanking` shall determine the amount of time to suspend listening and shall be in 1,6 µs ticks.

The moderator node shall increment an 8 bit counter, called `interval_count`, once per NUT. This counter shall be updated modulo (`modulus` + 1). Since the guardband is at the end of the NUT, the value of `interval_count` shall correspond to the NUT that just completed.

NOTE 3 Subclause 18.2 and the specifications of the ControlNet Object further constrain the value of `smax`, `umax`, `blanking` and `modulus`.

The `tMinus` field shall maintain a countdown until all nodes adopt new link parameters (see 16.9). The `slotTime` field shall determine the time to wait for a missing node and shall be in 1,0 µs ticks. An additional 1,0 µs shall be added to the value of the `slotTime`. The `slotTime` field shall be in the range 8 to 254; the other values shall be reserved. The `reserved` octet shall be set to zero.

16.5 Time distribution Lpacket

16.5.1 General

The moderator DLPDU provides a common reference marker that is synchronized between all nodes on the network. By distributing and processing time stamps relative to the reference instead of to the time distribution message, implementations are simplified whilst accuracy is improved by several orders of magnitude. Phase and frequency synchronization is inherent in this DL-protocol to a very high level of accuracy. The accuracy of time synchronization using the time distribution format defined in this subclause is implementation dependent, however it can be better than 10 µs.

16.5.2 Structure of the time distribution Lpacket

16.5.2.1 Packet format

The size and control fields shall follow the fixed tag format. The address shall be formed from the time distribution fixed tag and 0xFF the broadcast address. The link data field shall contain the following in the listed order:

```
class TimeDist_Lpacket : public fixedLpacket
{
public:
    USINT    revision;    // revision of time distribution format
    USINT    leap;        // leap second offset
    UINT     goodness;    // time relay control field
    DINT     gse;         // global squared error relative to ultimate master
    LINT     dctz;        // distribution channel time zero
    LINT     ts_ref;      // time stamp of previous reference pulse
    LINT     ts_tx;       // time stamp of this message's transmission
};
```

16.5.2.2 Revision

The revision parameter shall be set to zero. This parameter shall represent the revision of the time distribution format.

16.5.2.3 leap

The leap parameter shall specify the Universal Coordinated Time (UTC) leap seconds. This number, when added to the system time, shall give actual UTC time. This number can increment or decrement by one twice a year on the solstices, as dictated by the US Naval Observatory, to maintain synchronism between terrestrial and astronomical time. If zero, then the number of leap seconds is unknown.

NOTE The leap parameter should not be used in any control situations, but may be needed in some time relays to distribution channels that are based on UTC rather than Global Positioning Satellite (GPS) time.

16.5.2.4 goodness

16.5.2.4.1 Parameter structure

The goodness parameter shall be partitioned as shown in Figure 38.

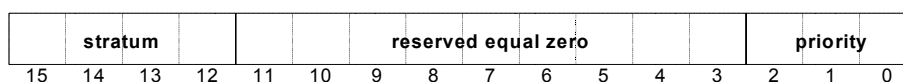


Figure 38 – goodness parameter of TimeDist_Lpacket

16.5.2.4.2 Stratum

The most significant field, called stratum, shall specify the number of time relays between this message, and a source of absolute time. A value of 0 shall signify an exact reference, and the value shall be incremented for every intermediate time relay. If the priority field is set to zero (lock not achieved), or the number of intermediate time relays exceeds 15, the goodness parameter shall be set to 15. Bits 3 through 11 shall be reserved and set to zero.

NOTE A time relay is a link-to-link router which distributes time synchronization messages on its links based on the time synchronization messages received on its other links.

16.5.2.4.3 source quality

The priority field shall indicate the quality of the time message as shown in Table 143.

Table 143 – Time distribution priority

Value	Meaning
7	Absolute system time. Acting as master
6	Absolute system time. Acting as dependent
5	Human set system time. Acting as master
4	Human set system time. Acting as dependent
3	Lock established to node on distribution channel other than this one
2	Lock established to node on this channel. System time unknown
1	(invalid)
0	Not synchronized with any other node

16.5.2.5 gse

The gse parameter shall indicate the cumulative rms stability squared. This parameter shall approximate the node's worst case stability relative to the rest of the system. The units of this parameter shall be $(0,1 \mu\text{s})^2$. When the rms stability is unknown or not yet determined, it shall be 0xFFFFFFFF.

16.5.2.6 LINT time parameters

In each of the LINT time parameters, the most significant bit shall be zero – only 63 bits shall be used. The units of these parameters shall be $0,1 \mu\text{s}$.

16.5.2.7 dctz

The dctz parameter shall indicate the system time offset from the distribution channel's arbitrary time zero, established when the networks are synchronized.

16.5.2.8 ts_ref

The ts_ref parameter shall indicate the time stamp of the last tone following a moderator DLPDU which had its interval_count equal to zero. The value of zero shall indicate that this value is not known. System time zero shall be defined as that used for the Global Positioning Satellites: 12:00 midnight, Jan 6, 1980 GMT.

16.5.2.9 ts_tx

The ts_tx parameter shall indicate the time stamp at the transmission of this message. The value of zero shall indicate that this value is not known. System time zero shall be defined as that used for the Global Positioning Satellites: 12:00 midnight, Jan 6, 1980 GMT. This parameter shall be set to zero.

NOTE This protocol does not use GPS time, it only uses the time zero point for GPS. So the 1999 roll over of GPS had no relevance for system time.

16.5.3 Sending and receiving the time distribution Lpacket

The current master Keeper for the link is responsible to sent the time distribution Lpacket at intervals and QoS values configured by the DL user to match its application requirements.

All nodes are responsible to receive the time distribution Lpacket.

16.6 UCMM Lpacket

16.6.1 General

The Unconnected Message Manager (UCMM) is a DLS-user entity providing an unconnected request/response message service for at least one outstanding message. It uses the DLL transmit service and the UCMM Lpacket for its messages.

16.6.2 Structure of the UCMM Lpacket

The size and control fields shall follow the fixed tag format. The address shall be formed from the UCMM fixed tag and destination MAC ID provided by the DLL transmit service. The link data field shall contain the DLS user data provided by the DLL transmit service.

16.6.3 Sending and receiving the UCMM Lpacket

The UCMM Lpacket shall be sent at the QoS priority specified by the DLL data transfer service.

The status of each UCMM transmission is confirmed to the DL user by returning the DLS-user-id (an internal handle or identifier) and the DLS Txstatus with one of the following values.

- a) "OK" — message successfully sent;
- b) "TXABORT" — sending process failed;
- c) "FLUSHED" — message has been removed from the pending queue before sending.

NOTE 1 Txstatus is a local variable so the coding is not specified.

NOTE 2 The FLUSHED status is only used in response to a deletion requested by the Queue maintenance service

NOTE 3 The parameter value OK is not an indication that the message has been received

The receiving station whose MAC ID matches that in the address, shall pass each correctly received data unit to its local DL user, together with the data unit size, the Lpacket service type and its source ID obtained from the DLPDU which conveyed the Lpacket.

16.7 Keeper UCMM Lpacket

16.7.1 General

Devices that implement the Keeper object shall also support sending and receiving of Keeper UCMM Lpackets. Only the Master Keeper shall send the Keeper UCMM Lpacket.

NOTE Using a specific address for Keeper objects reduces the decoding activity required by devices which do not include a Keeper object.

16.7.2 Structure of the Keeper UCMM Lpacket

The size and control fields shall follow the fixed tag format. The address shall be formed from the Keeper UCMM fixed tag and 0xFF the broadcast address. The link data field shall contain the user data provided by the Keeper object.

16.7.3 Sending and receiving the Keeper UCMM Lpacket

The Keeper UCMM Lpacket shall only be sent by a station with its Keeper object in the master state. It shall be requested at the QoS priority specified by the invoking Keeper object.

The status of each Keeper UCMM transmission is confirmed to the DL user by returning the DLS-user-id (an internal handle or identifier) and the DLS Txstatus with one of the following values.

- a) "OK" — message successfully sent;
- b) "TXABORT" — sending process failed;
- c) "FLUSHED" — message has been removed from the pending queue before sending.

NOTE 1 Txstatus is a local variable so the coding is not specified.

NOTE 2 The FLUSHED status is only used in response to a deletion requested by the Queue maintenance service

NOTE 3 The parameter value OK is not an indication that the message has been received

The receiving station whose MAC ID matches that in the address, shall pass each correctly received data unit to its local DL user, together with the data unit size, the Lpacket service type and its source ID obtained from the DLPDU which conveyed the Lpacket.

NOTE 4 As this is an unconnected message, the source MAC ID is required to enable a response as appropriate.

16.8 TUI Lpacket

16.8.1 General

A Table Unique Identifier (TUI) is used by all ControlNet objects to reference a consistent set of link configuration attributes. Two separate sets of attributes are held, each with their own TUI, one for current operation and one for pending operation. Changes from TUI data in the current link configuration to the TUI data in the pending link configuration shall occur at the completion of a synchronized parameter change.

The TUI Lpacket is published as a scheduled Lpacket by the Master Keeper to enable all attached devices to verify they have current configuration and schedule data.

16.8.2 Structure of the TUI Lpacket

The size and control fields shall follow the fixed tag format. The address shall be formed from the TUI fixed tag and 0xFF the broadcast address. The link data field shall contain the listed items shown in Table 144.

Table 144 – Format of the TUI Lpacket

Name	Data type	Description of parameter	Semantics of values
Size	USINT	Size of the Lpacket in words.	0x0D
Control	USINT	Link layer Lpacket control octet.	0x01 (Fixed tag Lpacket)
Fixed_Tag	USINT	Fixed tag value.	0x84 (TUI Lpacket)
Destination_Mac_Id	USINT	Who receives the Lpacket.	0xFF (Broadcast)
Unique_Id	UDINT	CRC of the Keeper's current attributes.	Least significant octet first.
Status_Flag	UINT	TUI flag values.	See the Keeper object TUI attribute for details.
Keeper_Mac_Id	USINT	MAC ID of the Keeper broadcasting the TUI.	See the Keeper object TUI attribute for details.
Reserved	USINT	Reserved for data alignment	
Net_Resource_Vendor_Id	UINT	Vendor ID of object holding Net Resource for exclusive use	
Net_Resource_Serial_Number	UDINT	Serial number of object holding Net Resource for exclusive use	
Net_Resource_Class	UDINT	Class number of object holding Net Resource for exclusive use	
Net_Resource_Instance	UDINT	Instance number of object holding Net Resource for exclusive use	

When generating the transmitted TUI Lpacket, any reserved fields shall use the values as specified in the corresponding reserved fields of Keeper object attribute 0x0101. During the configuration process, the programming tool shall set any reserved fields in attribute 0x0101, to zero.

16.8.3 Sending and receiving the TUI Lpacket

The requested QoS priority shall be scheduled.

When the Keeper object is in the MASTER_VERIFY, FAULTED_MASTER_VERIFY, MASTER, FAULTED_MASTER, NET_CHANGE_MASTER or NET_CHANGE_FAULTED_MASTER operating state, it shall attempt to transmit a Table Unique Identifier (TUI) Lpacket as scheduled data once every 4 NUTs (on NUT numbers 0, 4, 8, 12, 16 ...). All Keeper devices shall reserve 26 octets of scheduled link transmit time once every 4 NUTs for transmitting this Lpacket. (If sent unscheduled, it shall be the highest QoS priority unscheduled information.)

Reception of TUI Lpackets shall be enabled at power up by the ControlNet object in each device. Received TUI Lpackets shall be passed to the local ControlNet object.

16.9 Link parameters Lpacket and tMinus Lpacket

16.9.1 General

To synchronize the changing of local link parameters, Station Management shall enable the reception of two fixed tag Lpackets: 0x15 (tMinus) and 0x81 (link parameters) via the DLL_enable_fixed_request service of the DLL.

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 change synchronization is supported by the tMinus service.

16.9.2 Structure of Link parameters and tMinus Lpackets

The size and control fields shall follow the fixed tag format. The address shall be formed from the fixed tag for the service and 0xFF the broadcast address. The link data field for the appropriate Lpacket shall contain the following contents in the listed order.

a) Contents of the Link parameters Lpacket data field:

```
class LinkParm_Lpacket : public fixedLpacket
{
public:
    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
    UINT  zero;          // reserved
    USINT modulus;       // modulus of the interval counter
    USINT gb_prestart;   // transmit cut-off, 10 µs intervals before tone
    USINT unique_id;     // 32 bit CRC calculated from Keeper configuration table
    USINT status_flag;   // 16 bit status table set by the Keeper
    USINT reserved[8];   // all zeros
};
```

NOTE The format of the instance attributes 0x80 and 0x81 of the ControlNet Object are exactly the same as the link data of a fixed tag 0x81 Lpacket.

b) Contents of the tMinus Lpacket data field:

```
class tMinus_Lpacket : public fixedLpacket
{
public:
    USINT start_count;
    USINT reserved[3];
};
```

where start count is a number of NUTs to be counted before the transition and shall not be less than 8 and the reserved field shall contain zeros.

16.9.3 Sending and receiving the tMinus and Link parameters Lpackets

The Link parameters and tMinus Lpackets are issued by the master Keeper when a change is required to the link parameters. They shall be requested at the QoS priority specified by the invoking Keeper object.

Upon the reception of a LinkParm_Lpacket, each node shall load the pending copy of its link parameters using their local DLL_set_pending_request service. The pending parameters shall be loaded within 3 unscheduled transmit opportunities or within 1 second, whichever is greater. The pending link parameter attribute of the ControlNet Object shall also be updated with the pending copy of the link parameters.

Upon reception of a tMinus_Lpacket, the node shall initiate a synchronized parameter change using the DLL-tMinus-start-countdown-request service passing the start_count parameter. The ControlNet object shall copy its pending link parameter attribute to its current link parameter attribute when the DLL_tminus_zero_indication signals the completion of the synchronized parameter change.

NOTE The DLL copies its own pending link parameters to its current copy at the completion of the tMinus countdown and no DLL_tminus_zero_indication is signaled at the completion of the countdown unless the completion is preceded by a DLL-tMinus-start-countdown-request..

The moderator Lpacket contains a field, called tMinus, that shall be used to synchronize the update of the current link parameters. The DLL-tMinus-start-countdown-request shall cause a node to participate in a tMinus countdown, and, if the node is the moderator, shall initialize the tMinus field of the moderator Lpacket. The moderator node shall decrement this field before transmitting each moderator until the field equals zero. When the tMinus field transitions from 1 to 0, the DLL in each node participating in the countdown shall locally

generate a `DLL_tminus_zero_indication` to its `ControlNet` object which shall copy its pending link parameters into its current copy. If the `tMinus` field transitions to 0 from any value except 1, the countdown shall be aborted and no `DLL_tminus_zero_indication` shall be generated.

16.10 I'm alive Lpacket

16.10.1 General

This service allows all nodes on a link to recognize that another node has just been reset or power cycled.

16.10.2 Structure of the I'm Alive Lpacket

The size and control fields shall follow the fixed tag format. The address shall be formed from the I'm alive fixed tag and 0xFF the broadcast address. The link data field shall contain the following in the listed order.

```
class Im_Alive_Lpacket : public fixedLpacket
{
    public:
        USINT      Lpacket_variant;
        USINT      sourceID;
        USINT      reserved[6];
};
```

The `Lpacket_variant` and `reserved` fields shall be set to zero. The `sourceID` field shall be set to MAC ID of the node which transmits the Lpacket.

16.10.3 Sending and receiving I'm Alive

The QoS priority shall be specified by the invoking DL user.

When a node is first brought on-line before starting full operations, the network attachment monitor (NAM) in the new node shall broadcast at least 3 `Im_Alive_Lpackets`, subject to the rules for I'm Alive the state processing.

Receiving stations shall notify their management services.

NOTE Among the clients of these broadcasts are the higher layer connection managers of every other node on the link. The entities will abort any transaction records associated with the newly activated MAC ID and cancel any connections which had passed through the newly activated MAC ID. Since connection IDs (generic tags) are allocated dynamically, this ensures that connection ID's issued by the old device at the MAC ID are not improperly reused.

16.10.4 I'm alive state processing

A node shall only transmit one `Im_Alive_Lpacket` per NUT. To minimize the processing requirements in each node, a node which is broadcasting its three `Im_Alive_Lpackets` shall not count as a valid transmission any `Im_Alive_Lpacket` that is preceded in the NUT by more than seven other such Lpackets. These Lpackets shall be transmitted in the unscheduled portion of the NUT (QoS priority of either LOW or HIGH).

NOTE Because the timing of when nodes power up or reset relative to each other cannot be controlled, there is the possibility of an I'm alive Lpacket broadcast storm if a large number of nodes enter the I'm alive state at the same time. To minimize processing requirements, a transmission smearing and back off algorithm is recommended to monitor and control the intensity of any broadcast storm activity. Such an algorithm would spread the I'm alive Lpackets out over longer and longer periods of time, should a large number of nodes power up concurrently. If a small number of nodes power up together, the algorithm should allow them to exit the I'm alive state quickly since there cannot be a broadcast storm. An example of one such algorithm is shown in Figure 39.

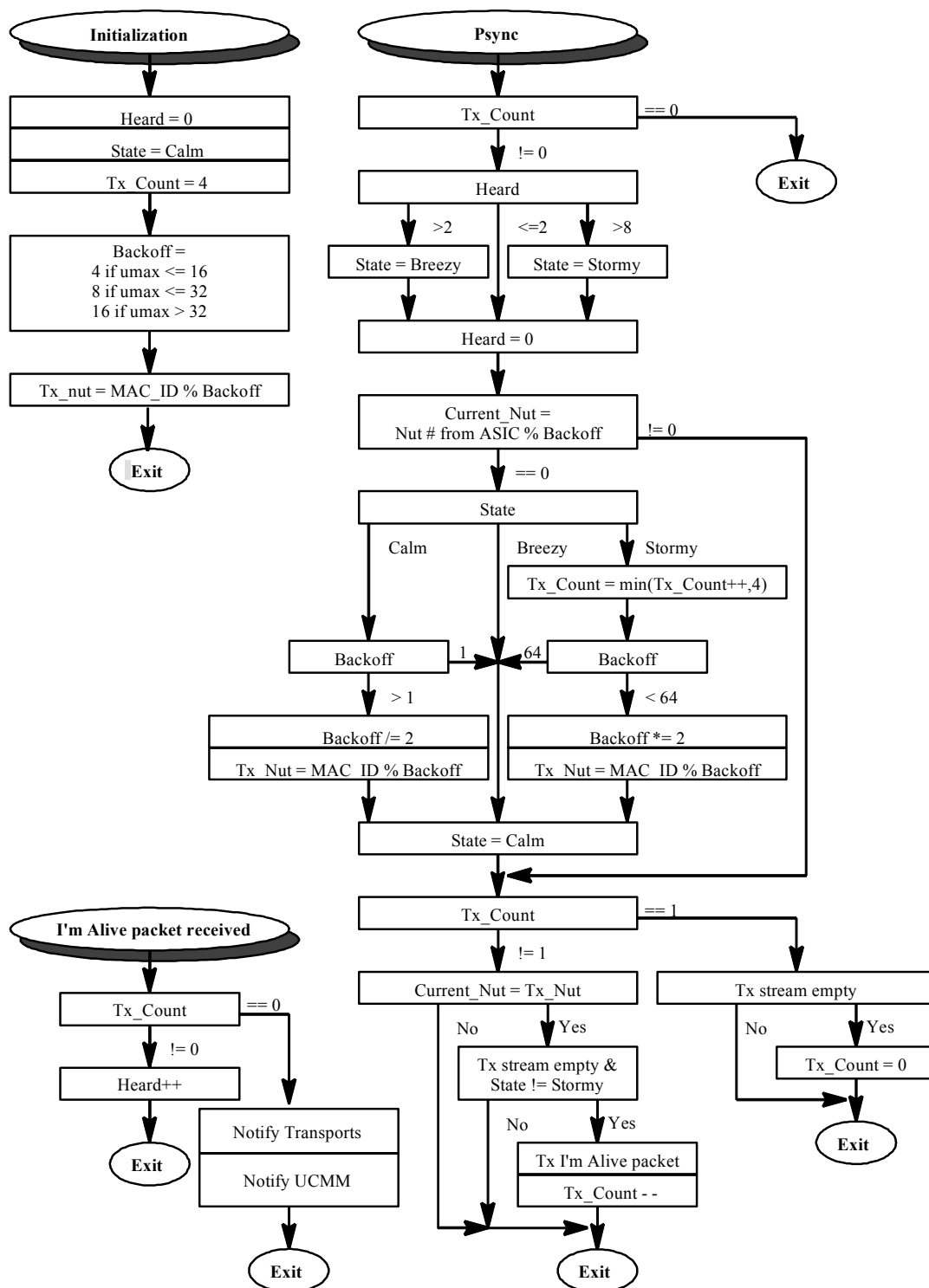


Figure 39 – Example I'm alive processing algorithm

16.11 Ping Lpackets

16.11.1 General

The ping service allows station management to initiate a ping request and receive a ping response from each active station, or one addressed station attached to the DLS provider.

All stations contain the ping service and at power-up the enable-fixed-request service is used by the DLS user to activate reception and response to ping requests within the local DLS provider.

16.11.2 Structure of the Ping Lpackets

The size and control fields shall follow the fixed tag format.

- a) When sending a ping request, the address shall be formed from the ping request fixed tag and an address specified by the DL user. The specified address may be a single MAC ID or 0xFF the broadcast address.

The link data field shall contain the following in the listed order.

```
class Ping_request : public fixedLpacket
{
public:
    USINT    client_ID;
    USINT    reserved[3];
};
```

where client_ID is the MAC ID of the request originator, and the reserved field shall contain zeros.

- b) When sending a ping reply, the address shall be formed from the ping reply fixed tag and the client_ID address from the corresponding ping request indication, (used here as destination for the reply).

The link data field shall contain the following in the listed order;

```
class Ping_reply : public fixedLpacket
{
public:
    USINT    server_ID;
    USINT    vendor_specific[4];
    USINT    reserved[3];
};
```

where the server_ID field shall contain the MAC ID of the node responding to the request, the vendor_specific field may contain any data and the reserved field shall contain zeros.

The DLL_xmit_fixed_request(id, Lpacket, sizeof(Ping_reply), HIGH, 0x29, client_ID) service shall be used to send the reply to the MAC ID specified in the client_ID field of the received Ping_request Lpacket.

NOTE A typical use for the vendor_specific field is for debugging or diagnostic information.

16.11.3 Sending and receiving the Ping Lpackets

The QoS priority for a ping request shall be specified by the invoking DL user. The QoS priority for a ping response shall be High.

Sending of ping-request Lpackets may be initiated by local management in any station.

At power up, the DLL_enable_fixed_request(id, 0x09) internal service of the Data Link Layer shall be invoked to enable the reception of fixed tag ping requests.

Upon receipt of a ping-request, each receiving station assembles a ping reply fixed tag Lpacket and sends it to the requesting station.

Upon receipt of a ping response, the requesting node passes the indication to local management.

16.12 WAMI Lpacket

16.12.1 General

A “Where Am I?” (WAMI) server shall be present in all permanent nodes which have a Network Access Port (NAP). The server shall not be present on transient nodes.

NOTE The WAMI (where am I) server allows a transient node (a node which connects to the network via the network access port or NAP) to determine the MAC ID of the node to which it is attached. This is a convenient way for software in a transient device to establish communication with the local permanent node.

16.12.2 Structure of the WAMI Lpacket

The size and control fields shall follow the fixed tag format. The address shall be formed from the WAMI fixed tag and 0xFF the broadcast address for all MAC IC nodes on this link. The link data field shall contain the following in the listed order.

```
class WAMI_Lpacket : public fixedLpacket
{
    public:
        USINT    requestID;
        USINT    responseID;
        USINT    reserved[6];
};
```

In a request WAMI, the requestID shall contain the MAC ID of the transient node that is sending the Lpacket. The other fields shall contain zeros. To convert a request into a response, the server shall copy its own MAC ID into the responseID field and reply. The QoS priority is Low.

NOTE 1 After WAMI fixed tag reception is enabled by the DLL_enable_fixed_request(id, 0x86) service of the DLL, fixed tags are received by the DLL_rcv_fixed_indication service..

NOTE 2 The Network Attachment Monitor (see 18.1) describes procedures to ensure the transient MAC ID does not duplicate an existing link MAC ID.

16.12.3 Sending and receiving the WAMI Lpacket

The QoS priority shall be Low.

The WAMI server is only implemented in devices having a NAP, these devices shall repeat all messages from the main link to the NAP and all messages from the NAP to the main link. Additionally these devices shall include means of detecting messages received via their NAP, and they shall only respond to Lpackets which match each of the following criteria:

- WAMI fixed tag 0x86;
- destination broadcast (0xFF);
- received via the device local NAP.

Lpackets that match these criteria shall be called request WAMI Lpackets. For all such Lpackets, the server shall generate a response WAMI that is addressed to the MAC ID which sent the request WAMI. The WAMI server shall discard all other WAMI fixed tag 0x86 Lpackets.

16.13 Debug Lpacket

This service may be used to transmit the state of an object building a trace of object state transitions on the wire.

The size and control fields shall follow the fixed tag format. The address shall be formed from the debug fixed tag and destination MAC ID provided by the DLL transmit service. The link data field shall contain the following in the listed order.

```
class Debug_Lpacket : public fixedLpacket
{
    public:
        UINT    object_class;
        UINT    data1;
        UINT    data2;
};
```

The value of `object_class` shall be the class code (see IEC 61158-6) of the object transmitting its state. The meaning of the `data1` and `data2` fields shall be defined in the object definition. Devices may not generate more than one fixed tag debug Lpacket per 10 seconds on average so as not to interfere with other unscheduled transmissions.

17 Type 2: Objects for station management

17.1 General

This protocol specification is quite flexible. It can provide deterministic and synchronized I/O transfer at cyclic intervals up to 1 ms and node separations up to 25 km. This performance is adjustable on-line by configuring the link parameters. These parameters, which govern the access to the link, can be tuned as required to match different applications.

Station Management allows these parameters to be changed on-line, as the network is working, it also allows the link to continue functioning while connections to new nodes are added and removed.

The functions of Station Management allow

- access to variables and events within each of the Layers;
- a common user interface;
- coordinating the change of link parameters;
- non-disruptively adding nodes to the link;
- tuning of link parameters;
- clock synchronization between nodes.

The access to variables and events within each of the Layers is accomplished by defining object interfaces for these parameters.

The ControlNet object provides a consistent interface to the Physical and Data Link Layers.

The Keeper object holds the link attributes for all devices on a link and shall be responsible for distributing those attributes to those devices in an orderly fashion. It also holds (for link scheduling software) a copy of the connection originator data for all connection originator devices using a network. If there are multiple Keeper objects on a link, they perform negotiations to determine which Keeper is the Master Keeper and which Keeper(s) perform backup Keeper responsibilities. The Master Keeper is the Keeper actively distributing link parameters and attributes to the nodes on the network. A backup Keeper is one that monitors Keeper related network activity and can transition into the role of Master Keeper should the Master Keeper fail to perform.

The Keeper object also contains support for obtaining, holding, and releasing the Network Resource, a network semaphore that is used to eliminate conflicts when modifying the attribute data held by the Keeper object(s) on a link.

The Scheduling object shall exist in connection originator (CO) devices. The Scheduling object shall be used by link scheduling software (LSS) to:

- read scheduled connection information;
- write schedule information;
- provide the CO with a signature (software key) which shall be used to authenticate the CO device's access to a specific link.

The TCP/IP interface object provides the mechanism to configure the TCP/IP interface of a device.

The Ethernet link object maintains link-specific counters and status information for a physical Ethernet ISO/IEC 8802-3 port.

17.2 ControlNet object

17.2.1 Overview

The ControlNet object shall provide a consistent Station Management interface to the Physical and Data Link Layers. This object shall make diagnostic information from these Layers available to client applications. Each node shall support one ControlNet object per link.

17.2.2 Attributes

17.2.2.1 Class attributes

The ControlNet object shall support the class attributes as specified in Table 145.

Table 145 – ControlNet object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x01	Required	Get	revision	UINT	revision of this object	first revision, value = 1
0x02	Required	Get	max instance	UDINT	maximum instance number	Value determined by node specifics

17.2.2.2 Instance attributes

17.2.2.2.1 General

The ControlNet object shall support the instance attributes as specified in Table 146.

Table 146 – ControlNet object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x80	Optional	Get	Pending_Link_Config	STRUCT of: 34 octets	pending link configuration parameters	see 16.9.2
			Link_Config	STRUCT of: 12 octets	pending link parameters	
			NUT_length	UINT	DLL NUT_length	in 10 µs ticks
			smax	USINT	DLL smax	0 to 99
			umax	USINT	DLL umax	1 to 99
			slotTime	USINT	DLL slotTime	in 1 µs ticks
			blanking	USINT	DLL blanking	in 1,6 µs ticks
			gb_start	USINT	DLL gb_start	in 10 µs ticks
			gb_center	USINT	DLL gb_center	in 10 µs ticks
			reserved	UINT	reserved	
			modulus	USINT	DLL modulus	127 required
			gb_prestart	USINT	DLL gb_prestart	in 10 µs ticks
			TUI	STRUCT of 22 octets	Keeper TUI	see 17.2.2.2.3
			unique_ID	UDINT	Keeper CRC	see 17.2.2.2.4

Table 146 (continued)

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			status_flag	UINT	TUI flag	see 17.2.2.2.5
			reserved	USINT [16]	reserved	
0x81	Required	Get	current_link_config	STRUCT of 34 octets	current link configuration parameters	same as attribute 0x80 see 16.9.2
0x82	Required	Get/Get_and_Clear	diagnostic_counters	STRUCT of 42 octets	diagnostic counters	see 17.2.2.2.7
			buffer_errors	UINT	buffer event counter	see 17.2.2.2.8
			error_log	OCTET[8]	bad DLPDU log	see 17.2.2.2.9
			event_counters	STRUCT of 32 octets	diagnostic counters	see 17.2.2.2.10
			good_frames_transmitted	OCTET[3]	good DLPDUs transmitted (LSB first)	
			good_frames_received	OCTET[3]	good DLPDUs received (LSB first)	
			selected_channel_frame_errors	USINT	framing errors detected on active receive channel	
			channel_A_frame_errors	USINT	framing errors detected on channel A	
			channel_B_frame_errors	USINT	framing errors detected on channel B	
			aborted_frames_transmitted	USINT	DLPDUs aborted during transmission (transmit underflows)	
			highwaters	USINT	LLC transmit underflow and LLC receive overflow	
			NUT_overloads	USINT	no unscheduled time in NUT (all time used for scheduled transmissions)	
			slot_overloads	USINT	more scheduled data queued for one NUT than allowed by sched_max_frame parameter	
			blockages	USINT	single Lpacket size exceeds sched_max_frame parameter	
			non_concurrence	USINT	two or more nodes could not agree whose turn it is to transmit	
			aborted_frames_received	USINT	incomplete DLPDUs received	

Table 146 (continued)

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			lonely_counter	USINT	number of times nothing heard on network for 8 or more NUTs	
			duplicate_node	USINT	DLPDU received from node with local node's MAC ID	
			noise_hits	USINT	noise detected that locked the modem rx PLL	
			collisions	USINT	Rx data heard just as we were going to transmit	
			mod_MAC_ID	USINT	MAC ID of the current moderator node	
			non_lowman_mods	USINT	Moderator DLPDUs heard from non lowman nodes	
			rogue_count	USINT	rogue events detected	
			unheard_moderator	USINT	DLPDUs being heard but no moderators being heard	
			vendor_specific	USINT	vendor specific	
			reserved	OCTET[4]	reserved	
			vendor_specific	USINT	vendor specific	
			vendor_specific	USINT	vendor specific	
			reserved	OCTET	reserved	
0x83	Required	Get	station_status	STRUCT of 6 octets	station status	See 17.2.2.2.11
			smac_ver	USINT	MAC implementation	
			vendor_specific	OCTET [4]	vendor specific	
			channel_state	OCTET	channel state LEDs, redundancy warning, and active channel bits	See 17.2.2.2.12
0x84	Get required, Set-(One Time Only) Optional	Set/Get	MAC_ID	STRUCT of: 4 octets	MAC ID switch and current settings	
			MAC_ID_current	USINT	current MAC ID	Range 0 to 99 See 17.2.2.2.13
			MAC_ID_switches	USINT	MAC ID switch settings	0 to 99, 0xFF See 17.2.2.2.14
			MAC_ID_changed	BOOL	MAC ID switches changed since reset	See 17.2.2.2.15
			Reserved	USINT	reserved	
0x85	Optional	Set/Get	Sched_max_frame	STRUCT of: 2 octets	scheduled data limit	

Table 146 (continued)

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			Sched_max_frame	USINT	scheduled maximum DLPDU	in octet pairs See 17.2.2.2.16
			Reserved	USINT	reserved	
0x86	Required	Get	error_log	STRUCT of: 10 octets	driver firmware buffer error counts and troublesome node list	See 17.2.2.2.17
			buffer_errors	UINT	buffer event counter	
			error_log	OCTET[8]	bad DLPDU log	
0x87	Optional	Get / Get_and_Clear	extended_diagnostic_counters	STRUCT of: 264 octets	additional diagnostic counters	
			unsched_transmitted	UDINT	number of unscheduled Lpackets transmitted	
			sched_highwater	UDINT	max number of shared/unscheduled Lpackets in transmit queue	
			sched_data	128 STRUCTS of: 2 octets (256 octets)	schedule information	
			words_in_use	USINT	number of scheduled words in use by this node this NUT	
			Lpkt_in_use	USINT	number of scheduled Lpackets in use by this node this NUT	
0x88	Optional	Get	Active_node_table	ARRAY of: 32 octets	one bit for each MAC ID 1 = node present 0 = node missing	see 17.2.2.2.18
0x89	Optional	Get	New_node_table	ARRAY of: 32 octets	one bit for each MAC ID 1 = node has joined link recently 0 = node has not joined link recently	see 17.2.2.2.19
LEGEND: LSB = least significant octet						

17.2.2.2.2 pending_link_config

The values within the attribute allow the pending link parameters to be read. This attribute shall change based on the Station Management function called synchronized parameter change, and shall not be set via the services of the ControlNet object.

17.2.2.2.3 Keeper TUI

The ControlNet object shall maintain separate copies of the TUI sub-attribute for pending_link_config and current_link_config attributes. Changes to the TUI data in the pending_link_config attribute shall occur at the reception of a fixed tag 0x81. Changes to the

TUI data in the `current_link_config` attribute shall occur at the completion of the synchronized parameter change.

17.2.2.2.4 unique_ID

The `unique_ID` field shall consist of a 32 bit CRC value calculated from the contents of the network configuration table maintained by the Keeper. This value shall be copied from the fixed tag 0x81 Lpacket.

17.2.2.2.5 status_flag

The `status_flag` field of the TUI sub-attribute shall consist of a 16 bit value. Unused bits shall be reserved and set to zero.

As shown in Table 147, the Keeper State bit (bit 4) shall indicate whether this node has ever heard a TUI from a master Keeper. The ControlNet object shall clear this bit at initialization. Since all TUI Lpackets sent by the master Keeper have this bit set, the ControlNet object need not manipulate this bit.

Reception of TUI Lpackets shall be enabled at power up using the `DLL_enable_fixed_request(id, 0x84)` service. The status field of the TUI Lpackets shall be checked. If bit3 = 0, bit4 = 1, and bit5 = 1, then the TUI Lpacket shall be copied into attribute 0x81 and TUI reception shall be disabled using the `DLL_disable_fixed_request(id, 0x84)`. Bits 0 and 1 of the TUI status field shall be used to override the default network redundancy settings for the node. They shall indicate if the network is being operated single channel or redundant channel mode. Only network redundancy information from TUI Lpackets with both the Keeper State and Keeper configured bits set, shall be used.

Table 147 – TUI status flag bits

Bit	Meaning
0 - 1	Network Redundancy Bit 1 = 0, bit 0 = 0; Illegal combination Bit 1 = 0, bit 0 = 1; Channel B only Bit 1 = 1, bit 0 = 0; Channel A only Bit 1 = 1, bit 0 = 1; Redundant
2	Holding Network Resource bit 0 = Network Resource not being held 1 = Network Resource being held for the object identified in the TUI
3	Network change in progress bit 0 = No network change in progress 1 = Network change in progress
4	Keeper State bit 0 = TUI transmitted by Keeper not in MASTER state 1 = TUI transmitted by Keeper in MASTER state.
5	Keeper configured bit 0 = No master Keeper heard since joining link 1 = Heard master Keeper since joining link
6 - 15	reserved (set to 0)

17.2.2.2.6 current_link_config

The values within the `current_link_config` attribute correspond to the link parameters currently used. The format of this attribute shall be identical to that of the `pending_link_config` attribute.

17.2.2.2.7 diagnostic_counters

The `diagnostic_counters` attribute shall maintain counts of events in the Physical and Data Link Layers.

NOTE The `diagnostic_counters` attribute may be read without effecting their values by using one of the `Get_Attribute` service requests. The information may be read and the values set to zero by using the `Get_and_Clear` service request. See 17.2.3.5.

The event information maintained by the firmware shall also be gettable (but not clearable) via the `error_log` attribute. This is done for efficiency. The information maintained by the firmware shall be immediately and directly accessible.

17.2.2.2.8 buffer_errors

The value of the `buffer_errors` sub-attribute shall increment on every receive overflow event and transmit underflow event. If an implementation never overflows or underflows, `buffer_errors` shall be zero.

17.2.2.2.9 error_log

If `DLL_event_indication` returns a value of `DLL_EV_badFrame`, the `error_log` sub-attribute records the last eight such indications. The `mac_id` parameter of this indication shall be recorded. The order of MAC IDs shall be most recent error first to oldest error last. Values of zero represent unused positions in the error log.

17.2.2.2.10 event_counters

The `event_counters` sub-attribute shall contain counts of link events. The counter shall rollover when their values exceed their limits.

17.2.2.2.11 station_status

This attribute shall be only gettable.

17.2.2.2.12 channel_state

The `channel_state` octet shall represent the current state of the network status indicators, if implemented. The bits of the `channel_state` octet are described in Table 148.

NOTE The network status indicators are described in Annex D.

Table 148 – Channel state bits

Bits	Description
0,1,2	Channel A LED State 0 = Off 1 = Solid green 2 = Flashing green-off 3 = Flashing red-off 4 = Flashing red-green 5 = Railroad red-off 6 = Railroad red-green 7 = Solid red
3,4,5	Channel B LED State Indications same as channel A LED.
6	Redundancy Warning – When warning, the non-active channel in a redundant configuration is unusable by the controller. 0 = normal (no warning) 1 = warning
7	Active Channel – Indicates which of two channels the receiver is currently listening to. 0 = Channel B 1 = Channel A

17.2.2.2.13 MAC_ID_current

The MAC_ID_current value shall be accessible using the Get_Attribute services. Setting shall be required on devices without address switches. Only the first set, of the MAC_ID_current value, after a ControlNet object reset shall be accepted. Setting shall be optional on devices with address switches.

The range of allowable values for MAC IDs shall be 1 to 99.

17.2.2.2.14 MAC_ID_switches

The MAC_ID_switches value shall be gettable and not settable on all devices. The returned value shall be 0xFF for auto-address nodes and nodes without address switches.

17.2.2.2.15 MAC_ID_changed

The MAC_ID_changed value shall be cleared when the device is reset and set whenever the device detects that the address switches have been changed. Once set, it stays set until the device is reset, even if the address switches are returned to their original settings. The frequency at which the address switches are checked shall be device dependent.

When a change in the network address switches is detected, this flag shall be set and a minor error shall be reported to the host.

17.2.2.2.16 sched_max_frame

The sched_max_frame attribute shall be used to limit the maximum size of the node's scheduled transmissions. A value of 0 shall indicate that a node shall not transmit during the scheduled portion of the NUT.

17.2.2.2.17 error_log

The instance attribute 0x86 shall be a copy of the error_log sub-attribute contained in instance attribute 0x82.

NOTE The information in this attribute is a copy of the information found in the diagnostic_counters attribute. This copy is only gettable. The values in this attribute may be cleared only by using the Get_and_Clear service request on the diagnostic_counters attribute. See 17.2.3.5 for more details.

This copy of the error_log shall only be accessible via the Get_Attribute services. This attribute shall only be cleared when the master copy in 0x82 is cleared.

17.2.2.2.18 Active node table (instance attribute = 0x88)

Instance attribute 0x88 of the ControlNet object shall consist of an array of 256 bits, one per MAC ID. The least significant bit shall correspond to MAC ID = 0; the most significant bit shall correspond to MAC ID = 255. The bit for a specific MAC ID shall be set (1) for a node within 1 second of a node transmitting on the link. The bit shall be cleared (0) within 20 seconds of the node leaving the link. A node shall be considered to have left the link if it misses three consecutive transmit opportunities.

17.2.2.2.19 New node table (instance attribute = 0x89)

Instance attribute 0x89 of the ControlNet object shall consist of an array of 256 bits, one per MAC ID. The least significant bit shall correspond to MAC ID = 0; the most significant bit shall correspond to MAC ID = 255. The bit for a specific MAC ID shall be set (1) for a node within 1 second of a node transmitting on the link. The bit shall be cleared (0) between 10 and 20 seconds after the node has joined the link.

17.2.3 Services

17.2.3.1 Common services

The ControlNet object shall support the common services as specified in Table 149.

Table 149 – ControlNet object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x05	N/A	Required	Reset	Return ControlNet object to power up state
0x10	N/A	Conditional	Set_Attribute_Single	Set network or node specific configuration information
0x0E	Required	Required	Get_Attribute_Single	Get network or node specific configuration information
0x01	N/A	Optional	Get_Attributes_All	Get network and node specific configuration information
0x02	N/A	Optional	Set_Attributes_All	Set network and node specific configuration information
0x03	N/A	Optional	Get_Attributes_List	Get network and/or node specific configuration information
0x04	N/A	Optional	Set_Attributes_List	Set network and/or node specific configuration information

17.2.3.2 Reset service

The Reset service shall reinitialize the Network Attachment Monitor (NAM) causing it to monitor the link activity and then attempt to join the link. MAC ID switches shall not be re-evaluated due to the reset.

The ControlNet object need not respond to a Reset request before actually performing the reset operation.

17.2.3.3 Set_Attribute_Single

If any attribute has been implemented as settable, the Set_Attribute_Single service shall be required.

17.2.3.4 Get_Attribute_All response

The Object/Service specific reply data portion for a successful Get_Attribute response is shown in the Instance Attributes for the ControlNet object table. The size of this response is dependent on the attribute instance(s) being accessed.

Because of the large size of the Get_Attribute_All response, devices with limited resources need not support this service.

The data portion of a Get_Attribute_All service request shall contain the following attributes in the following order:

0x81 - current_net_config

0x82 - diagnostic_counters

0x83 - station_status

0x84 - MAC_ID

0x86 - error_log

17.2.3.5 Class specific services

The ControlNet object shall support the class specific services as specified in Table 150.

Table 150 – ControlNet object class specific services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x4C	N/A	Required	Get_and_Clear	Get then clear the diagnostic counters.
0x4D	N/A	Required	Enter_Listen_Only	Force the node to enter and stay in listen only mode until reset.
0x4E	N/A	Required (See 17.2.3.8)	Where_am_I	Determine the MAC ID of the device this node is attached via the NAP.
0x4F	N/A	Conditional	Auto_Address	Required in all auto-address nodes. Select a new MAC ID using the auto address sequence of the NAM.

17.2.3.6 Get_and_Clear service

This class specific service shall be evoke an identical response to the Get_Attribute_Single service; however, after the response Lpacket is built, the value of the attribute shall be set to zero. Only instance attributes 0x82 and 0x87 shall support this service.

17.2.3.7 Enter_Listen_Only service

The Enter_Listen_Only service shall cause this object to send a DLL_listen_only_request(FALSE) to the DLL associated with this instance of the ControlNet object. This service need not generate a response since the node may stop transmitting before the response is sent.

NOTE 1 The node should remain in the listen-only state until a reset request is sent to either the ControlNet or Identity object.

NOTE 2 The Enter_Listen_Only service is intended to be used for diagnostic purposes to force nodes to drop off the link until they are specifically requested to rejoin. This allows a node to effectively be removed from the link without having to physically disconnect the network cable.

17.2.3.8 Where_am_I service

All transient nodes shall implement the Where_am_I service. This service shall determine the MAC ID of the node to which the transient node is connected via its network access port (NAP). The service may be implemented by using the WAMI server functionality of the Station Management entity.

NOTE A transient node may use this service to identify the MAC ID of the node into whose NAP it is plugged.

The Where_am_I response shall be a single USINT containing the MAC ID of the permanent node to which the transient node is connected. If the ControlNet object is unable to complete the WAMI query, it shall return a MAC ID of 255.

17.2.3.9 Auto_Address service

For nodes that support automatically selecting a MAC ID, this service shall reinitialize the NAM causing it to monitor the link activity and then attempt to join the link possibly with a new MAC ID. Nodes that do not support automatically selecting a MAC ID shall not implement this service.

The Auto_Address service need not respond prior searching for a new MAC_ID.

17.2.4 Behavior

17.2.4.1 Effect of DLL_tminus_zero_indication

Upon receipt of this indication, the object shall copy the contents of instance attribute 0x80 (pending_link_config) into instance attribute 0x81 (current_link_config). An internal copy of instance attribute 0x80 shall be kept even if the implementation has not made it accessible via one of the Get_Attribute services.

17.2.4.2 Duplicate Node

If a DLL_event_indication(DLL_EV_dupNode) is received from the Data Link Layer, the ControlNet object shall force the DLL into a listen-only state. The ControlNet object shall latch this state until a reset service is received. A reset to the Identity object shall also end the listen only state.

17.2.4.3 Redundancy

The network redundancy setting for a node shall be initially set at start up to the best capability of the device. (Channel A only for single channel devices, both channels for network redundant devices). The power up redundancy setting shall be reflected in the TUI contained in the instance attribute 0x82.

The network redundancy setting for a node shall be updated to the current network redundancy setting by the Keeper in one of two ways:

- The node receives a TUI Lpacket from the wire with the master Keeper and configured Keeper bits set and the net change in progress bit reset in the Status_Flags word;
- The node receives a fixed tag 0x81 Lpacket.

In either case, the received TUI data shall be copied into attribute 0x80.

If a single channel node joins a redundant link, it shall behave as if it were a redundant node with a faulty channel.

17.2.5 Module status indicator

The module status indicator, if implemented, shall provide the following status indications:

- a) solid green when connections are active in the *on line* state;
- b) flashing green/off in the *check for moderator listen only* state;
- c) flashing green/off when rogue conditions are detected;
- d) flashing red/off for a recoverable error;
- e) flashing red/off when in the DUP_LISTEN_ONLY state;
- f) solid red for a irrecoverable error.

17.3 Keeper object

17.3.1 Overview

The Keeper object shall hold the link attributes for all devices on a link and shall be responsible for distributing those attributes to those devices in an orderly fashion. It also holds (for link scheduling software) a copy of the connection originator data for all connection originator devices using a network. If there are multiple Keeper objects on a link, they perform negotiations to determine which Keeper is the Master Keeper and which Keeper(s) perform backup Keeper responsibilities. The Master Keeper is the Keeper actively distributing attributes to the nodes on the network. A backup Keeper is one that monitors Keeper related network activity and can transition into the role of Master Keeper should the Master Keeper fail to perform.

The Keeper object also contains support for obtaining, holding, and releasing the Network Resource, a network semaphore that is used to eliminate conflicts when modifying the attribute data held by the Keeper object(s) on a link.

17.3.2 Revision history

The revision history of the Keeper object is described in Table 151.

Table 151 – Keeper Object revision history

Class revision	Description of changes
01	Initial Release
02	Release for IEC 61158.

17.3.3 Attributes

17.3.3.1 Class attributes

The Keeper object shall support the class attributes as specified in Table 152.

Table 152 – Keeper object class attributes

Number	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x1	Required	Get	Revision	UINT	Revision of this object	2 (revision 2) (See Note)
NOTE 1 Revision 1 Keeper devices are restricted to only operate at MAC ID = 1. These devices perform the functions of the Keeper as described in this specification as long as no other Keeper is present on the Network. At any other MAC ID, these devices containing revision 1 Keepers perform as if no Keeper objects were present.						
NOTE 2 Revision 2 Keepers devices operate at any legal MAC ID value. These devices perform the functions of the Keeper as described in this specification in the presence of any other Revision 2 Keepers on the Network.						

17.3.3.2 Instance attributes

The Keeper object shall support the instance attributes as specified in Table 153. Only one instance of the Keeper object is allowed. That instance must be sensitive to the port on which requests are received. This is especially important in the case of a router device where one Keeper instance receives requests from two links.

Table 153 – Keeper object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x00	Required	Get	status	STRUCT of 2 octets	Keeper object status	
			state	USINT	current Keeper operating state	
			reserved	USINT	reserved for data alignment	
0x01 through 0x63	Required	Set/Get	port_status	STRUCT of 10 octets	port status for node 1 through node 99	
			port_status	UINT	port status	
			ID	STRUCT of 8 octets	node type identification	
			vendor	UINT	vendor code	
			type	UINT	product type	
			code	UINT	product code	
			major	USINT	major revision	
			minor	USINT	minor revision	
0x64 – 0xFE	Reserved					reserved for future expansion
0xFF	Required	Set/Get	net_config	STRUCT of 12 octets	current network parameters	
			NUT	UINT	Network Update Time	in 10 μ s ticks
			smax	USINT	Scheduled max node ID	0 to 99
			umax	USINT	Unscheduled max node ID	0 to 99
			Slot_Time	USINT	Slot Time	in 1 μ s ticks
			Blank_Time	USINT	Blanking Time	in octets
			Gb_Start	USINT	Guard Band Start	in 10 μ s ticks
			Gb_Center	USINT	Guard Band Center	in 10 μ s ticks
			Reserved	UINT	Reserved for Data Alignment	
			Int_Cnt mod	USINT	Interval Count Modulus (Macrocycle length)	127
			Gb_Prestart	USINT	Guard Band Prestart	in 10 μ s ticks
0x100	Required	Set/Get	name	UINT [33]	current name of this link	32 unicode characters and a unicode NULL
0x101	Required	Set/Get	RT_TUI	STRUCT of 22 octets	Table Unique Identifier	
			unique_ID	UDINT	attribute CRC	
			status_flag	UINT	TUI flag bits	See 17.2.2.2.3

Table 153 (continued)

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			reserved	USINT [16]	reserved to allow common format with attribute 0x102	
0x102	Required	Get	Link_TUI	STRUCT of 22 octets	Table Unique Identifier (current link only)	
			unique_ID	UDINT	attribute CRC	
			status_flag	UINT	TUI flag bits	See 17.2.2.2.3.
			Keeper_MAC_ID	USINT	MAC ID of node broadcasting the TUI	Any legal MAC ID
			reserved	USINT	reserved for data alignment	
			Net_Resource_Vendor_Id	UINT	Vendor ID of object holding Net Resource	
			Net_Resource_Serial_Number	UDINT	Serial number of object holding Net Resource	
			Net_Resource_Class	UDINT	Class of object holding Net Resource	
			Net_Resource_Instance	UDINT	Instance of object holding Net Resource	
0x103	Required	Set/Get	Cable_Config	STRUCT of up to 100 octets	Current cable configuration	
			Id	USINT	Id value	Always 0xFF
			Num_Elements	USINT	Number of cable configuration elements in network configuration	Range 1 to 24
			Propagation_time	UINT	Number of 100 nanosecond ticks	
			Physical element	Phy element [24]		
			Phy_element	STRUCT of 4 octets		
			Vendor_id	UINT	Vendor code	
			Product_code	USINT	Product code	
			How_many	USINT		
0x104	Required	Set/Get	CO_summary	STRUCT of 204 octets	General information about the <i>co_data</i> attribute	
			data_size	UINT	Size of <i>co_data</i> attribute	In words
			connection_info_revision	USINT		Used by programming tool 0 = Format 1 1 = Format 2 2 - 255 reserved

Table 153 (continued)

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			tool_keeper_revision	USINT	Highest level co_data parse rule supported	Level supported is $\leq$ Keeper object revision 0 = Format 1 1 - 255 reserved See 17.3.5.7
			Offsets	UINT[100]	Array of word offsets into the co_data attribute, by MAC ID	In words FFFFhex = no data for this MAC ID node 0 based index
0x105	Required	Set/Get fragment	CO_data	STRUCT of up to 14988 octets	connection originator information	Tool Keeper Revision Format 1
			Branch	STRUCT of variable size	Information relative to all routers or connection originators on this link	
			num_devices	UINT	number of devices on this link	
			device	STRUCT of variable size	details of this device	
			type	USINT	types of device	1 = router 2 = connection originator
			Path_Size	USINT	size of path to node	In 16 bit words
			Path	array of UINT	path to device	
			CO_data	STRUCT of variable size	Details of CO data	present only for CO devices (type = 2)
			COP	UDINT	CO password, as provided to the CO at the end of a scheduling session (see 17.4.3.10)	
			Size	UINT	Size of Connection data	In 16 bit words
			Connection	STRUCT of variable size	Connection data	Connection Info Revision in the format specified by connection_info_revision of attribute 0x104 One per connection for this CO device See 17.3.3.8.

17.3.3.3 Operating state definitions

The values for the Keeper operating state shall be as defined in Table 154.

Table 154 – Keeper operating state definitions

Value	Description
0	Power up - Not on line
1	Power up - TUI wait
2	Power up - TUI poll
3	Backup
4	Master verify
5	Master
6	Faulted Backup
7	Faulted Master verify
8	Faulted Master
9	Net Change - Backup
10	Net Change - Master
11	Net Change - Faulted Backup
12	Net Change - Faulted Master

17.3.3.4 Port status flag bit definitions

The values for the Port status flag shall be as defined in Table 155.

Table 155 – Port status flag bit definitions

Bit	Meaning
0	Node accepts scheduled connections bit 0 = Node does not accept scheduled connections 1 = Node accepts scheduled connections
1 – 15	reserved (set to 0)

17.3.3.5 status_flag

The Keeper configured bit cleared in the TUI status word shall identify a Keeper with an invalid configuration or cleared memory.

Referring to Table 156, the status_flag field of the TUI sub-attribute shall consist of a 16 bit value. Unused flag bits shall be reserved and set to 0.

The holding network resource bit shall only be used by the Keeper object.

The master Keeper bit shall indicate whether or not this node has ever heard from a master Keeper. The ControlNet object shall clear this bit at initialization. All TUIs sent by the master Keeper shall have this bit set. If a TUI Lpacket is received with the master Keeper bit not set, the Lpacket shall be retained as a received TUI, but the redundancy information shall not be used.

The redundancy bits shall be used to override the default redundancy settings for the node. They shall indicate if the link is being operated in the single channel mode or redundant. The redundancy information shall not be used if a TUI Lpacket is received with the master Keeper bit clear.

The net change in progress bit shall indicate that a synchronized network change sequence is in progress and that new nodes shall delay going on-line until the change sequence is completed and the bit is cleared. This bit shall be cleared in the case of a network resource timeout situation or if the network resource is released.

Table 156 – TUI status flag bits

Bit	Meaning	Used in:		
		RT_TUI	Network_TUI	ControlNet object TUI
0 - 1	Network Redundancy Bit 1 = 0, bit 0 = 0; Illegal combination Bit 1 = 0, bit 0 = 1; Channel B only Bit 1 = 1, bit 0 = 0; Channel A only Bit 1 = 1, bit 0 = 1; Redundant	X	X	X
2	Holding Network Resource bit 0 = Network Resource not being held 1 = Network Resource being held for the object identified in the TUI		X	
3	Network change in progress bit 0 = No network change in progress 1 = Network change in progress	X	X	X
4	Keeper State bit 0 = TUI transmitted by Keeper not in MASTER state 1 = TUI transmitted by Keeper in MASTER state.		X	X
5	Keeper configured bit 0 = Keeper attributes not configured (RT_TUI) 1 = Keeper attributes configured (RT_TUI)	X	X	X
6 - 15	reserved (set to 0)			

17.3.3.6 unique_ID

The unique_ID field of the TUI attribute shall consist of a 32-bit value calculated from the contents of the Keeper attribute table (excluding the TUI attribute) by the software configuration tool. The following attributes shall be used, in the given order to calculate the TUI unique_ID:

0x1 through 0x63 (port parameters for nodes 1 through 99)

0xFF (link parameters)

0x100 (link name);

0x101 (RT TUI) redundancy bits only;

0x103 (cable configuration);

0x105 (CO device password data only);

The TUI unique identifier shall be calculated as follows:

- the TUI unique identifier shall be initialized to 0;
- the data for attributes 0x1 through 0x63, attribute 0xFF, attribute 0x100, redundancy setting (attribute 0x0101 RT_TUI status flag bits 0–1), 0x103 and each CO password (in address order) shall be run through the CRC polynomial.

The polynomial used to calculate the CRC shall be $x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x^1 + 1$ as shown in Figure 40.

NOTE The mathematics of this polynomial is similar to that given in 5.1. However, the following software computation neither presets the accumulation to all ones nor complements the result. Therefore the term $L(X)(X^k + 1)$ is not included in the computation of $F(X)$ equivalent to that given in Figure 40.

```

UDINT CRC( // (r) the updated CRC
    unsigned long CRC, // (i) the CRC to start with
    OCTET* buffer, // (i) pointer to buffer of octets / octets
    WORD size) // (i) the number of octets in the buffer

# define KEEPER_CRC_POLY 0xEDB88320L
WORD octetIndex;
WORD bitIndex;
OCTET oneOctet;

// loop through the octets passed, including them in the CRC
for (octetIndex=0; octetIndex < size; octetIndex++)
{
    oneOctet = buffer[octetIndex];
// loop through the bits passed, including them in the CRC
for (bitIndex = 0; bitIndex < 8; bitIndex++)
{
    if ((oneOctet & 0x1) ^ (CRC & 0x1))
    {
        CRC = (CRC >> 1) ^ KEEPER_CRC_POLY;
    } else {
        CRC = (CRC >> 1);
    }
    oneOctet >>= 1;
}
}

// return the new CRC
return(CRC);
}

```

Figure 40 – Keeper CRC algorithm

17.3.3.7 Keeper MAC_ID

The Keeper_MAC_ID should be the MAC ID of the current master Keeper that is broadcasting the TUI. If no TUI has been received in the last 12 NUTS, a value of 0xFF shall be returned.

17.3.3.8 Connection

Various formats exist for connection data. Each format is described below. The entire collection of connection data entries shall be an integral number of words.

Format 1 does not specify the size of the path; therefore, its path[] shall be one of two forms:

- class segment, instance segment, then two connection point segments; or
- an ANSI extended symbol segment.

```

// Format 1
USINT  O2T_Size      // in 16 bit words
UINT    O2T_Rpi      // in 1 ms ticks from 2 to 32767
USINT  T20_Size      // in 16 bit words
UINT    T20_RPI      // bit 15 = connection point, 0 = point to point, 1 = multipoint
                // bit 14 = API in milliseconds from 2 to 32767
USINT  O2T_Nui_Api   // The highest bit set indicates the API;
                // bit 7 = 128*NUT, bit 6 = 64*NUT, etc.
                // Remaining bits indicate the O2T_NUI. If API = 1, NUI = 0
USINT  T_Node        // Range 1 - 99
USINT  T20_Nui_Api   // The highest bit set indicates the API;
                // bit 7 = 128*NUT, bit 6 = 64*NUT, etc.
                // Remaining bits indicate the O2T_NUI. If API = 1, NUI = 0
UINT    path[]       // forward open connection path from this

```

```
// Format 2
UINT 02T_net_params // copied from forward open
UINT T20_net_params // copied from forward open
UINT 02T_RPI // RPI in floating point format
UINT T20_RPI // RPI in floating point format
USINT 02T_schedule // from schedule segment in forward open
USINT MAC_ID
USINT T20_schedule // from schedule segment in forward open
USINT path_size // in 16-bit words
UINT path[] // forward open connection path from this
```

```
UDINT convert_to_10us_ticks (UINT RPI)
{
    return (RPI <= 0x4000) ? RPI :
           (RPI <= 0xE000) ? (RPI & 0x1FFF) + 0x2000 << (RPI>>13) - 1 :
                           : (RPI & 0x0FFF) + 0x1000 << (RPI>>12 & 1) + 7;
}

UINT convert_to_Keeper_format (UDINT ticks)
{
    if (ticks < 0x00004000) return RPI;
    if (ticks < 0x00008000) return RPI>>1 & 0x1FFF | 0x4000; // 0 - 16383 by 1
    if (ticks < 0x00008000) return RPI>>1 & 0x1FFF | 0x4000; // 16384 - 32766 by 2
    if (ticks < 0x00010000) return RPI>>2 & 0x1FFF | 0x6000; // 32768 - 65532 by 4
    if (ticks < 0x00020000) return RPI>>3 & 0x1FFF | 0x8000; // 65536 - 131064 by 8
    if (ticks < 0x00040000) return RPI>>4 & 0x1FFF | 0xA000; // 131072 - 262128 by 16
    if (ticks < 0x00080000) return RPI>>5 & 0x1FFF | 0xC000; // 262144 - 524256 by 32
    if (ticks < 0x00100000) return RPI>>7 & 0x0FFF | 0xD000; // 524288 - 1048448 by 128
    if (ticks < 0x00200000) return RPI>>8 & 0x0FFF | 0xE000; // 1048576 - 2096896 by 256
    return 0xFFFF;
}
```

17.3.3.9 Attribute data

A Keeper maintains the link and node configuration information for its link in an attribute data structure. The Keeper shall keep two copies of the attribute structure: pending copy in RAM and a current copy in non-volatile memory. When new attributes are set in the Keeper, they shall be set in the pending RAM copy. The RAM copy shall be written to the currently active copy in non-volatile storage only upon receipt of a special request to do so.

The attributes shall use the following rules for getting and setting:

- Set service requests set the pending copy in RAM;
- Get service requests get the current copy in non-volatile storage.

Keeper shall maintain the collection of attributes specified in Table 157:

Table 157 – Keeper attributes

Attribute number	Keeper attributes
0x01 - 0x63	(port parameters for nodes 1 through 99);
0xFF	(link parameters);
0x100	(link name);
0x101	(Keeper TUI);
0x103	(cable configuration);
0x104	(node offset information into CO data);
0x105	(CO path and password information);

All Keeper attribute definitions assume a maximum of 99 nodes on the link. The Keeper object shall have memory to store the CO_summary and CO_data attributes.

Memory requirements (in octets) for the Keeper attributes are as specified in Table 158.

Table 158 – Memory requirements (in octets) for the Keeper attributes

Octets	Keeper attributes
2	Keeper status
990	Port parameters (99 nodes at 10 octets each)
12	link parameters
66	link name
22	TUI
100	Cable configuration
204	CO summary
14988	CO data (7494 words)
16384	Total (Exactly 16K octets)

The sequence of events required to properly and safely update the Keeper attributes shall be as follows:

- Obtain the Network Resource;
- Issue a Change_Start service request to the Keeper;
- Issue the appropriate Set_Attribute service request(s) to update the pending copy of the Keeper attributes;
- Issue a Change_Complete service request to the Keeper. Doing this shall initiate a synchronized parameter change;
- Release the Network Resource.

17.3.4 Common services

The Keeper common services shall all operate with an internal time out to prevent a hardware error condition from indefinitely stalling operations. The actual time out values shall be device dependent.

The Keeper object shall support the common services as specified in Table 159.

Table 159 – Keeper object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x03	N/A	Required	Get_Attributes_List	Get network and node specific configuration information
0x04	N/A	Required	Set_Attributes_List	Set network and node specific configuration information
0x0E	Required	Required	Get_Attribute_Single	Get network and node specific configuration information.
0x10	N/A	Required	Set_Attribute_Single	Set network and node specific configuration information.

The Keeper object is capable of receiving these service requests via fixed tags 83 and 88. When received via fixed tag 83, the request is processed and responded to as required independent of Keeper state. When received on fixed tag 88, the Keeper will only respond to the request if it is in the Master, Net Change Master, Faulted Master or Net Change Faulted Master states. Keeper nodes in other states must not respond to the request.

17.3.5 Class specific services

Any device that has implemented the Keeper object shall also implement the Keeper UCMM.

17.3.5.1 All class specific services

The Keeper object shall support the following class specific as specified in Table 160.

Table 160 – Keeper class specific services

Service code	Need in implementation		Service name	Description of service	Parameter(s)	
	Class	Instance				
0x4B	N/A	Required	Obtain_Network_Resource (ONR)	Attempt to obtain the Network Resource for this link. If successful, hold for 60 seconds.	UINT UDINT UDINT UDINT	vendor ID serial number class instance
0x4C	N/A	Required	Hold_Network_Resource (HNR)	Continue to hold the previously obtained Network Resource for this link for 60 seconds.	UINT UDINT UDINT UDINT	vendor ID serial number class instance
0x4D	N/A	Required	Release_Network_Resource (RNR)	Release the Network Resource for this link.	UINT UDINT UDINT UDINT	vendor ID serial number class instance
0x4E	N/A	Required	Change_Start (CS)	Copy current attributes in non-volatile storage to pending attributes in RAM. Enter the appropriate <i>net change</i> operating state.	<none>	
0x4F	N/A	Required	Change_Complete (CC)	Copy pending attribute data in RAM to current data in non-volatile storage. Restart normal Keeper operation. Initiate synchronized network change if net attributes changed. If number of sets does not match, do <i>change_abort</i> processing.	16 bit number of “sets” performed since <i>Change_Start</i>	
0x50	N/A	Required	Change_Abort (CA)	Discard changes made to the pending attribute data, release the network resource and return to normal Keeper operation.	<none>	
0x51	N/A	Required	Get_Signature (GS)	Get signature value for a specific connection originator (CO password)	Request: path size (octet) (size is in words) path Response: 32 bit cop	
0x52	N/A	Required	Get_Attribute_Fragment (GAF)	Get a portion of an attribute specifically designed to be fragmented, that is too large to fit in a single Lpacket. For use only on the co_data attribute	UINT size; //words UINT offset; //offset	
0x53	N/A	Required	Set_Attribute_Fragment (SAF)	Set a portion of an attribute specifically designed to be fragmented, that is too large to fit in a single Lpacket. For use only on the CO_data attribute.	UINT size; //words UINT offset; //offset UINT data [];	

The error codes for these services shall be as specified in Table 161, where the service codes are the abbreviations for the service names specified in Table 160.

Table 161 – Service error codes

Error code	Meaning	Error code use by service								
		ONR	HNR	RNR	CS	CC	CA	GS	GAF	SAF
0x02	resource_unavailable_err	X	X	X	X					
0x03	invalid_parameter_err					X		X	X	X
0x04	path_segment_err							X	X	X
0x05	path_dest_unknown_err	X	X	X	X	X	X	X	X	X
0x08	unimplemented_service_err	X	X	X	X	X	X	X	X	X
0x09	invalid_attr_err								X	X
0x0C	wrong_object_state_err					X	X			X
0x0D	already_exists_err	X			X					
0x0F	no_permission_err		X	X						
0x11	reply_too_large_err							X		
0x13	not_enough_data_err	X	X	X		X		X	X	X
0x14	undefined_attr_err		X							
0x15	too_much_data_err							X		X
0x16	nonexistent_object_err	X	X	X	X	X	X	X	X	X
0x26	invalid_path_size_err	X	X	X	X	X	X	X	X	X

17.3.5.2 Network resource

The Network Resource (NR) shall be a link semaphore that is used to eliminate conflicts when modifying attribute information in Keeper objects.

In operation, the NR shall appear as a status flag bit and other data fields in the TUI Lpacket that shall be repetitively broadcast by the master Keeper.

A device that wishes to update the Keeper attribute information first shall submit a request to the master Keeper to obtain the NR on its behalf (via an *Obtain_Network_Resource* service). This request shall be broadcast since the identity of the master Keeper is not generally known. Once obtained, the device shall periodically ask (every 15 seconds) the master Keeper to continue holding the NR via a *Hold_Network_Resource* service request. The master Keeper shall maintain a 60 second timer that is started when the NR is initially obtained, and retriggered when a subsequent *Hold_Network_Resource* request from the same node is received. If the timer ever times out, the master Keeper shall automatically release the NR.

A backup Keeper shall respond to the Network Resource service requests as follows. When an *Obtain_Network_Resource* service request is received, it shall start a 60 second timer that shall be retriggered when a *Hold_Network_Resource* service request is received. The timer shall also be started when a Keeper powers up on an existing network and senses that the network resource is currently being held. The timer shall be stopped when a *Release_Network_Resource* service is received or the timer expires.

When a backup Keeper becomes the master Keeper, it shall use the data from the most recently received TUI Lpacket to hand off the holding of the NR from the old master Keeper.

In both master and backup Keepers, the 60 second timer shall be used to determine if the node requesting the NR has failed or has been prematurely removed from the network. When a Keeper's 60 second NR timer expires, the Keeper shall

- a) abort any attribute changes in progress;
- b) reset the net change in progress TUI status bit;
- c) exit the net change operating state;
- d) return to normal operation.

Since the NR can be held over a fairly long period if the user is performing substantial edits, the NR shall be cleanly handed off to the new master Keeper if the master Keeper changes. This handoff shall be invisible to the node that the master Keeper is holding the NR for. Since all information regarding the NR is contained in the TUI Lpacket being broadcast by the master Keeper, and since backup Keepers are receiving the broadcast TUI and are using the broadcast TUI for determining if and when they should take over as master Keeper, the backup Keepers shall have all the information needed to cleanly transition holding the NR when the appropriate backup Keeper takes over as master Keeper.

Faulted backup and master Keeper shall process Network Resource requests in the same way as non-faulted backup and master Keeper.

When the network resource is not being held, the related fields in the network TUI Lpackets shall be zeroed:

- NR_VENDOR_ID;
- NR_SERIAL_NUMBER;
- NR_CLASS;
- NR_INSTANCE.

17.3.5.3 Obtain_Network_Resource service

The Obtain_Network_Resource service request shall cause the master Keeper to hold the Network Resource for 60 seconds for itself or another node, if it is not already being held. The time period that the Network Resource is held may be extended indefinitely through use of the Hold_Network_Resource service request.

If the Network Resource is already being held for any node, an error status shall be returned.

The Obtain_Network_Resource service shall return the error codes as defined in Table 161, as designated in column ONR for this service.

17.3.5.4 Hold_Network_Resource service

The Hold_Network_Resource service request shall cause the master Keeper to continue holding the Network Resource for the next 60 seconds if the node requesting the hold is the same node for which the master Keeper is currently holding the Network Resource.

If the Network Resource is being held for another node or is not being held, an error status shall be returned.

The Hold_Network_Resource service shall return the error codes as defined in Table 161, as designated in column HNR for this service.

17.3.5.5 Release_Network_Resource service

The Release_Network_Resource service request shall cause the master Keeper to stop holding the Network Resource if the node requesting the release is the same node for which the master Keeper is currently holding the Network Resource. If a network change is in process and the network resource is released, the network change will be aborted.

Whether the Network Resource is being held for another node or is not being held, an error status shall be returned.

The Release_Network_Resource service shall return the error codes as described in Table 161, as designated in column RNR for this service.

17.3.5.6 Change_Start service

The Change_Start service request shall cause the Keeper object to:

- a) copy the current copy of the attributes in non-volatile storage into the pending copy in RAM;
- b) clear the count of Set_Attribute service requests (including set single, list and fragment) received by the ControlNet object;
- c) put the Keeper object into the appropriate net_change operating state.

This service request shall only be processed if the Network Resource is being held by the master Keeper as indicated by the net_resource bit in the broadcast TUI Lpacket.

The Change_Start service shall return the error codes as defined in Table 161, as designated in column CS for this service.

17.3.5.7 Change_Complete service

The Change_Complete service request shall cause the Keeper object to:

- a) copy its pending copy of the attributes from RAM to the current copy of the attributes in non-volatile storage;
- b) perform a synchronized network change;
- c) verify tool_keeper_revision in attribute 105 is $\leq$ Keeper object revision (if not true, mark attribute table invalid);
- d) exit whatever network change operating state it is in;
- e) return to normal Keeper operation.

If the Keeper object is not in one of the net change operating states, the Change_Complete service request shall be ignored and an incorrect state error response shall be generated.

If the number of Set_Attribute service requests (including: Set_Attribute_Single, Set_Attribute_List and Set_Attribute_Fragment) received by the Keeper object since the Change_Start service request does not match the number in the Change_complete parameter, a Change_Abort shall be executed instead and an error response shall be generated.

The reply to this service shall be returned after at least one TUI is broadcast with the net_change_in_progress bit cleared. A synchronized change shall take place prior to the TUI transmission with the bit reset.

The Change_Complete service shall return the error codes as defined in Table 161, as designated in column CC for this service.

17.3.5.8 Change_Abort service

The Change_Abort service shall cause the Keeper object to

- a) discard changes made to the pending attribute data;
- b) release the network resource;
- c) return to normal Keeper operation.

The reply to this service shall be returned after at least one TUI is broadcast with the net_change_in_progress bit and holding_network_resource bit cleared.

The Change_Abort service shall return the error codes as defined in Table 161, as designated in column CA for this service.

17.3.5.9 Get_Signature service

The Get_Signature service shall cause the master Keeper to look up the CO password values for a connection originator in the CO_data attribute. The path to the device shall be given as the attribute in the request. This path shall be parsed and used to search the tree structure of the CO_data one branch at a time, to locate the appropriate COP (CO password) value, which is returned as a parameter in the response.

The path shall be parsed one branch at a time since the path entries in the CO_data attribute only specify the portion of the path between one branch and the next, not the entire path to that branch.

The Get_Signature service shall return the error codes as specified in Table 161, as designated in column GS for this service.

17.3.5.10 Get_Attribute_Fragment service/response

The Get_Attribute_Fragment service shall collect and return some portion of the CO_data attribute data managed by the Keeper object. Any part of this attribute may be read. The CO_data attribute shall be the only Keeper attribute which responds to fragment services. Accessing other attributes shall cause an error to be returned.

NOTE When getting attribute data, the current attribute data from non-volatile storage, not the pending copy of the attribute should be returned.

The Get_Attribute_Fragment service shall return the error codes as defined in Table 161, as designated in column GAF for this service.

17.3.5.11 Set_Attribute_Fragment service

The Set_Attribute_Fragment service shall be used to set any portion of the CO_data attribute managed by the Keeper object. Only the CO_data attribute shall be accessed via this service. Accessing other attributes shall cause an error to be returned.

The Set_Attribute_Fragment service shall be processed only when the Keeper object is in one of the net change operating states. The Keeper object shall return an invalid mode error if it is not in one of the net change operating states when a Set_Attribute_Fragment service request is received.

Setting attribute data, shall set the pending copy of the attribute, not the current copy in non-volatile storage. The pending attributes shall be copied to the current attributes in non-volatile storage upon reception of a class specific Change_Complete service request.

The Keeper object shall not perform attribute value checking during any set operation.

The Set_Attribute_Fragment service shall return the error codes as defined in Table 161, as designated in column SAF for this service.

17.3.5.12 Table unique identifier (broadcast), fixed tag 0x84

When the Keeper object is in the master verify, faulted_master_verify, master, faulted_master, net change master or net_change_faulted_master operating state, it shall attempt to transmit a special Table Unique Identifier (TUI) Lpacket as scheduled data once every 4 NUTs (that is, on NUT numbers 0, 4, 8, 12, 16 ...). All Keeper devices shall reserve 26 octets of scheduled link transmit time once every 4 NUTs for transmitting this Lpacket. If sent unscheduled, it shall be the highest QoS priority unscheduled information.

When generating the transmitted TUI Lpacket, any reserved fields shall use the values as specified in the corresponding reserved fields of attribute 0x0101. The programming tool shall set any reserved fields in attribute 0x0101, to zero.

The format of the TUI Lpacket link data shall be as defined in Table 162.

Table 162 – Wire order format of the TUI Lpacket

Name	Data type	Description of parameter	Semantics of values
Size	USINT	Size of the Lpacket in words.	0x0D
Control	USINT	Link layer Lpacket control octet.	0x01 (Fixed tag Lpacket)
Fixed_Tag	USINT	Fixed tag value.	0x84 (TUI Lpacket)
Destination_Mac_Id	USINT	Who receives the Lpacket.	0xFF (Broadcast)
Unique_Id	UDINT	CRC of the Keeper's current attributes.	Least significant octet first.
Status_Flag	UINT	TUI flag values.	See the TUI attribute for details.
Keeper_Mac_Id	USINT	MAC ID of the Keeper broadcasting the TUI.	See the TUI attribute for details.
Reserved	USINT	Reserved for data alignment	
Net_Resource_Vendor_Id	UINT	Vendor ID of object holding Net Resource for exclusive use	
Net_Resource_Serial_Number	UDINT	Serial number of object holding Net Resource for exclusive use	
Net_Resource_Class	UDINT	Class number of object holding Net Resource for exclusive use	
Net_Resource_Instance	UDINT	Instance number of object holding Net Resource for exclusive use	

17.3.6 Service error codes

Table 163 lists possible error codes and the most likely condition under which the code would be returned.

Table 163 – Service error codes

Error code	Meaning	Return condition
0x02	resource_unavailable_err	The requested network resource is not available.
0x03	invalid_parameter_err	An invalid parameter was provided to the service.
0x04	path_segment_err	Whenever an path over and above the attribute number is specified.
0x05	path_dest_unknown_err	Object instance does not exist.
0x08	unimplemented_service_err	Whenever the service is not supported for an attribute.
0x09	invalid_attr_err	Whenever an attribute is specified which is not supported in this object.
0x0C	wrong_object_state_err	The object is in a state which does not allow the service to be performed.
0x0D	already_exists_err	The service has already been obtained by the requesting node.
0x0E	not_settable_err	Attribute is not settable.
0x0F	no_permission_err	Node requesting this service does not currently own the resource.
0x11	reply_too_large_err	Not enough room in the response buffer to reply.
0x13	not_enough_data_err	Whenever the request does not contain enough data
0x14	undefined_attr_err	attempts to access an undefined attribute
0x15	too_much_data_err	More data has been encountered than expected for this service request.
0x16	nonexistant_object_err	Keeper object not available.
0x26	invalid_path_size_err	Whenever an invalid path segment for this service is specified.

17.3.7 Behavior

The numeric indication of Keeper state shall be as defined in Table 164.

Table 164 – Keeper object operating states

State	Description
0,1,2 – Power up	The Keeper object is either waiting for the node to come on-line or is determining if it is a legitimate Keeper for this link.
3 – Backup	The Keeper object has determined that it is a legitimate Keeper for this link but either has determined it is a backup Keeper or has not yet determined if it is the master Keeper for this link.
4 – Master Verify	The Keeper object has determined that it is a legitimate Keeper for this link but has not heard another active Keeper on the link or has received a good TUI from a matching Keeper at a higher node number or has heard a TUI from a faulted Keeper. It starts broadcasting the TUI Lpacket but does not process synchronized parameter changes or respond to requests of the master Keeper.
5 – Master	The Keeper object has determined that it is a legitimate Keeper for this link and that it is the master Keeper for this link. It broadcasts the TUI Lpacket, processes synchronized link changes when required to change link parameters and must respond to requests of the master Keeper.
6 – Faulted Backup	The Keeper object has determined that it is not a legitimate Keeper for this link, and is not the Keeper if there are only faulted Keepers on the link.
7 – Faulted Master Verify	The Keeper object is faulted but has not heard another active Keeper on the network for 12 NUT times. It starts broadcasting the TUI Lpacket but does not process synchronized link changes or respond to requests of the master Keeper.
8 – Faulted Master	The Keeper object is faulted and has assumed faulted master Keeper duties. There are no unfaulted Keeper's on the network. It broadcasts the TUI Lpacket, processes synchronized link changes when required to change network parameters and must respond to requests of the master Keeper.
9,10,11,12 – Net Change	The Keeper is having its attributes updated. There is a <i>Net Change</i> substate for the <i>backup</i> , <i>master</i> , <i>fault</i> and <i>faulted master</i> states previously discussed. The separate substates are needed so the Keeper object can exit the <i>Net Change</i> state properly when updates complete normally and when updates are aborted. The pending attributes may be set only when the Keeper is in a <i>Net Change</i> state. Nodes in the <i>net change master</i> and <i>net change faulted master</i> must respond to requests of the master Keeper. Nodes in the <i>net change backup</i> or <i>net change faulted backup</i> states must not respond to requests of the master Keeper.

17.3.8 Miscellaneous notes

Keepers shall maintain parameters only for nodes on the same link.

Each Keeper shall be capable of distributing link parameters to all nodes on that link. The Master Keeper shall be the Keeper with the lowest MAC ID of all the legitimate Keepers on the link. A faulted master Keeper on a link may or may not be the lowest MAC ID.

The Keeper shall be responsible for configuring those devices that appear in its attributes. The Master Keeper shall be the only Keeper that issues responses to service requests via fixed tag 0x88 that are broadcast to all Keeper objects on a net.

All legitimate Keeper objects on a link shall have identical copies of the network attributes for all devices on the link. Keepers that do not agree with the Master Keeper shall stay in the faulted backup state until they agree (agree means their TUI *unique_id* parameters are identical).

Only one programming terminal may modify Keeper attributes at a time. The programming terminal shall acquire the Network Resource prior to starting the Keeper modification and shall retain possession of the Network Resource during the entire modification procedure. If a programming terminal makes a change to a Keeper, it shall update all of the other Keepers on

the link as well by broadcasting the Change_Start, Set_Attribute, and Change_Stop service requests.

The Keeper shall check the “tool Keeper revision” in attribute 0x0104 for a valid attribute table. If the Keeper is unable to understand that revision, the Keeper shall clear bit 5 in any TUI transmissions so that other more capable Keepers can assume the master Keeper responsibilities. Keeper revisions 0 and 1 shall be identical to revision 2.

The “connection info revision” value is independent on the Keeper revision, and shall only be utilized by programming tools.

The Keeper may, for network diagnostic purposes, broadcast to all nodes a debug fixed tag (0x90) Lpacket with three words of data. The first word identifies that the Keeper object is sending the diagnostic. The second word is the current “Keeper state” and the third word is the next “Keeper state”.

17.3.9 Keeper power up sequence

17.3.9.1 General

The Keeper power up sequence shall allow the Keeper object to begin operations under a variety of conditions, from a normal configured network power on state to one where many network devices are new and unconfigured.

The Keeper object power up sequence shall not begin until the NAM has brought the Keeper node on-line. The Keeper power up shall determine if this particular Keeper device is a legitimate Keeper for the local link and to bring the Keeper object into either the backup or faulted backup operating state.

If this device is a legitimate Keeper for the link, the Keeper object shall determine if it is the Master Keeper or a Backup Keeper for the link. If this device is not a legitimate Keeper for the link, the Keeper object enters a fault state. An unconfigured Keeper shall enter the fault state at power up.

The 12*NUT timer shall be retriggered whenever a TUI Lpacket is received that indicates the NR is being held. This shall indicate that the Keeper attributes are being updated.

The Keeper object’s power up sequence is illustrated in Figure 41.

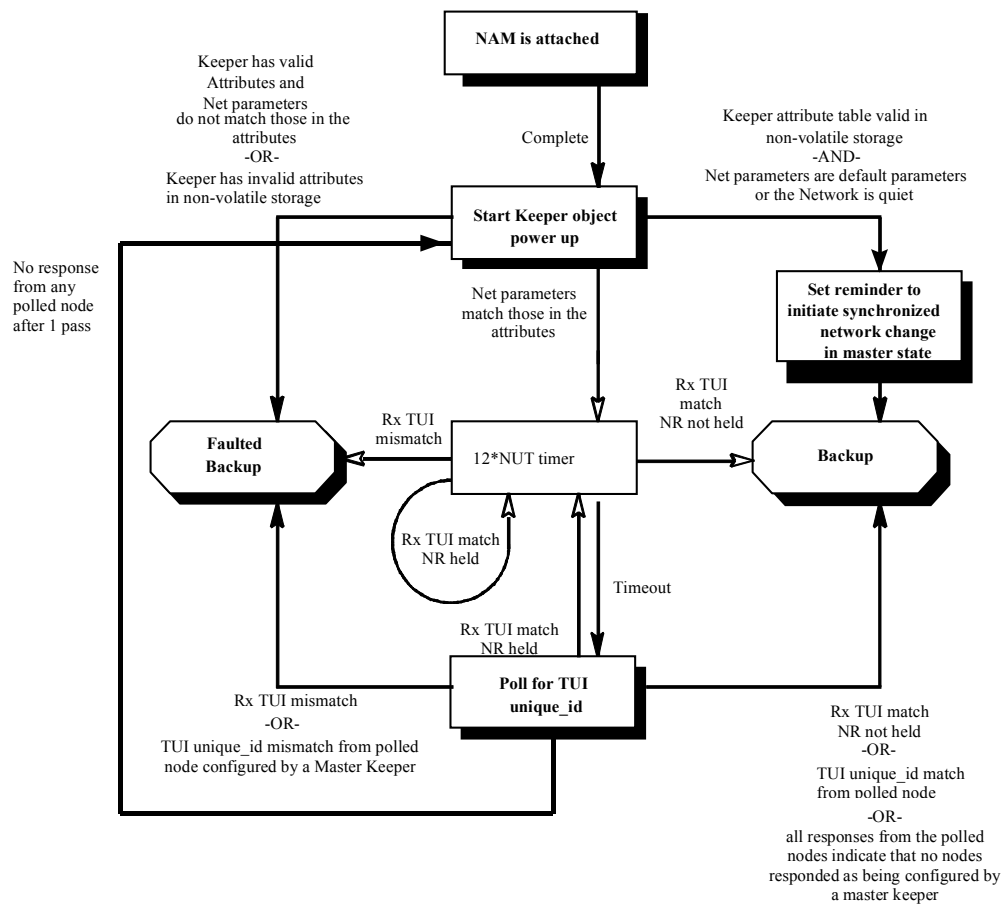


Figure 41 – Keeper object power-up state diagram

17.3.9.2 Poll for TUI unique_id

The Poll for TUI unique_id state in Figure 41 shall require the following operation be performed by the Keeper object:

- Issue a UCMM request to the addressed node's ControlNet object to read the object's current configuration parameters. Should the request fail, proceed on to the next MAC ID until UMAX is reached. Multiple concurrent requests may be permitted to reduce the poll time.
- Each MAC ID in the range 1 through UMAX (excluding the MAC ID of the Keeper node) may be polled singly or concurrently with a Get_Attribute_Single request on the current_link_config attribute of the ControlNet object of the node. This attribute contains a copy of the current TUI attribute for the node. Concurrent access to multiple nodes at one instant will serve to accelerate the polling process.
- If no response is heard from any polled node and UMAX has been reached, the poll is aborted and the Keeper object shall transition to the "Start Keeper object power up state", since at least one other node on a non-default network is present but unable to respond. This will allow the power up poll process to repeat until a node is capable of responding.
- If a response is received from a polled node and the "Keeper State" bit in the returned TUI status_flag was set, the polled node was configured by a Keeper in the "Master" operating state. Compare the TUI unique_id value returned by the polled node to the one in this Keeper's attribute table. If it matches, the poll is aborted and the Keeper object shall transition to the "backup" operating state. If it mismatches, the poll is aborted and the Keeper object shall transition to the "faulted backup" operating state.

- e) If a response is received from a polled node and the “Keeper State” bit in the returned TUI status_flag was not set, the polled node was not configured by a Keeper in the “Master” operating state. In this case, the information is discarded and the poll continued. If all nodes respond in this way, indicating that none of them were configured by a Keeper in the “Master” operating state, the poll is terminated and the Keeper shall transition to the “backup” operating state since in this situation, all nodes are either new or have lost their stored configuration.

17.3.9.3 Operating states

The Keeper object operating states shall be defined as in Figure 42, Table 165, and the following sections.

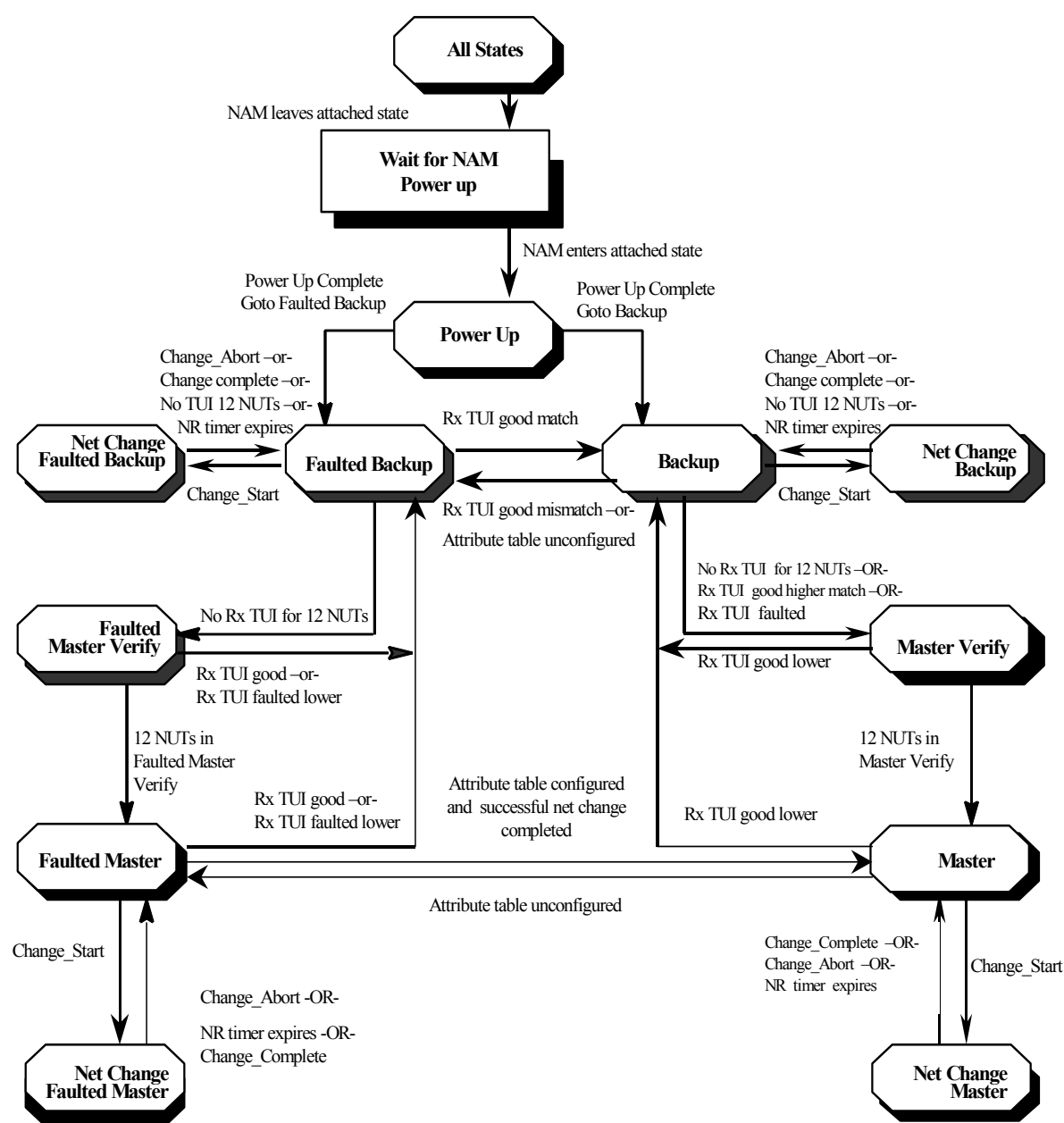


Figure 42 – Keeper object operating state diagram

Table 165 – Keeper object state event matrix

Event	State										
	Power Up	Backup	Master Verify	Master	Faulted Backup	Faulted Master Verify	Faulted Master	Net Change (Back-up)	Net Change (Master)	Net Change (Faulted Backup)	Net Change (Faulted Master)
Rx TUI from a good node						Goto Faulted Backup	Goto Faulted Backup				
Rx TUI from a good lower node			Goto Backup								
Rx TUI from a good higher node (match)		Goto Master verify									
Rx TUI from a good node (mismatch)		Goto Faulted Backup									
Rx TUI from a good node (match)					Goto Backup						
Rx TUI from a faulted node		Goto Master Verify									
Rx TUI from a faulted lower node						Goto Faulted Backup					
No TUI Rx for 12*NUT		Goto Master Verify			Goto Faulted Master Verify			Abort change. Goto Backup		Abort change. Goto Faulted Backup	
12 Nut times elapse			Goto Master			Goto Faulted Master					
Change_Start		Goto Net Change Backup		Goto Net Change Master	Goto Net Change Faulted Backup		Goto Net Change Faulted Master				
Change_Complete								Goto Backup	Goto Master	Goto Backup	Goto Master verify
Change_Abort								Do Change Abort. Goto Backup	Do Change Abort. Goto Master	Do Change Abort. Goto Faulted Backup	Do Change Abort. Goto Faulted Master
60 second Network Resource timer expires								Abort change. Goto Backup	Abort change. Goto Master	Abort change. Goto Faulted Backup	Abort change. Goto Faulted Master
NAM leaves attached state	Goto wait for NAM powerup										
Power up complete to Backup	Goto Backup										
Power up complete to Faulted Backup	Goto Faulted Backup										
NOTE Blank cells indicate that this event cannot occur, or if it does occur, no action shall be taken by the Keeper object.											

17.3.9.4 Network resource

Access to changing Keepers shall be controlled through the use of the Network Resource. Only one device can acquire the Network Resource for exclusive access at a time.

The Network Resource shall be held by the master Keeper for the node performing the attribute updates. If the master Keeper does not hear from the node performing the update at least once every 60 seconds, the master Keeper shall automatically release the Network Resource, abort the change in progress, and return to normal operation.

17.3.9.5 Faulted Keeper

Any Keeper whose *TUI unique_id* attribute does not match that of the broadcast TUI from a Keeper shall enter a fault state unless a net change operation is in progress.

17.3.9.6 MAC ID chooses Master Keeper

The Keeper node MAC ID included in the TUI message shall be used by the Keeper to determine which Keeper on a link is the Master Keeper.

17.3.9.7 Master Verify

The Keeper shall wait for $12 \times \text{NUT}$ after the first TUI broadcast begins before assuming master Keeper status. This shall be sufficient time for a Keeper with a lower MAC ID to become Master Keeper. Other Keepers become Backup Keepers.

17.3.9.8 Rogue

If the node containing the Keeper rogues during a synchronized network change sequence, the "Net Change in Progress" bit shall be cleared and control shall be returned back to the start of the synchronized network change processing immediately after the NAM brings the node back on-line. The normal Keeper object power-up sequence shall be bypassed.

17.3.9.9 Synchronized network change processing

17.3.9.9.1 Algorithm

Although the moderator is used by the link to distribute operating parameters, it shall be the responsibility of the Keeper object to initiate changes to the link parameters.

All link parameter changes shall be accomplished through the use of a synchronized network change algorithm. This shall provide a means for all nodes on the network to change their current link parameters in unison without disrupting network operation.

The algorithm for the synchronized network change operation shall be as defined in Figure 43.

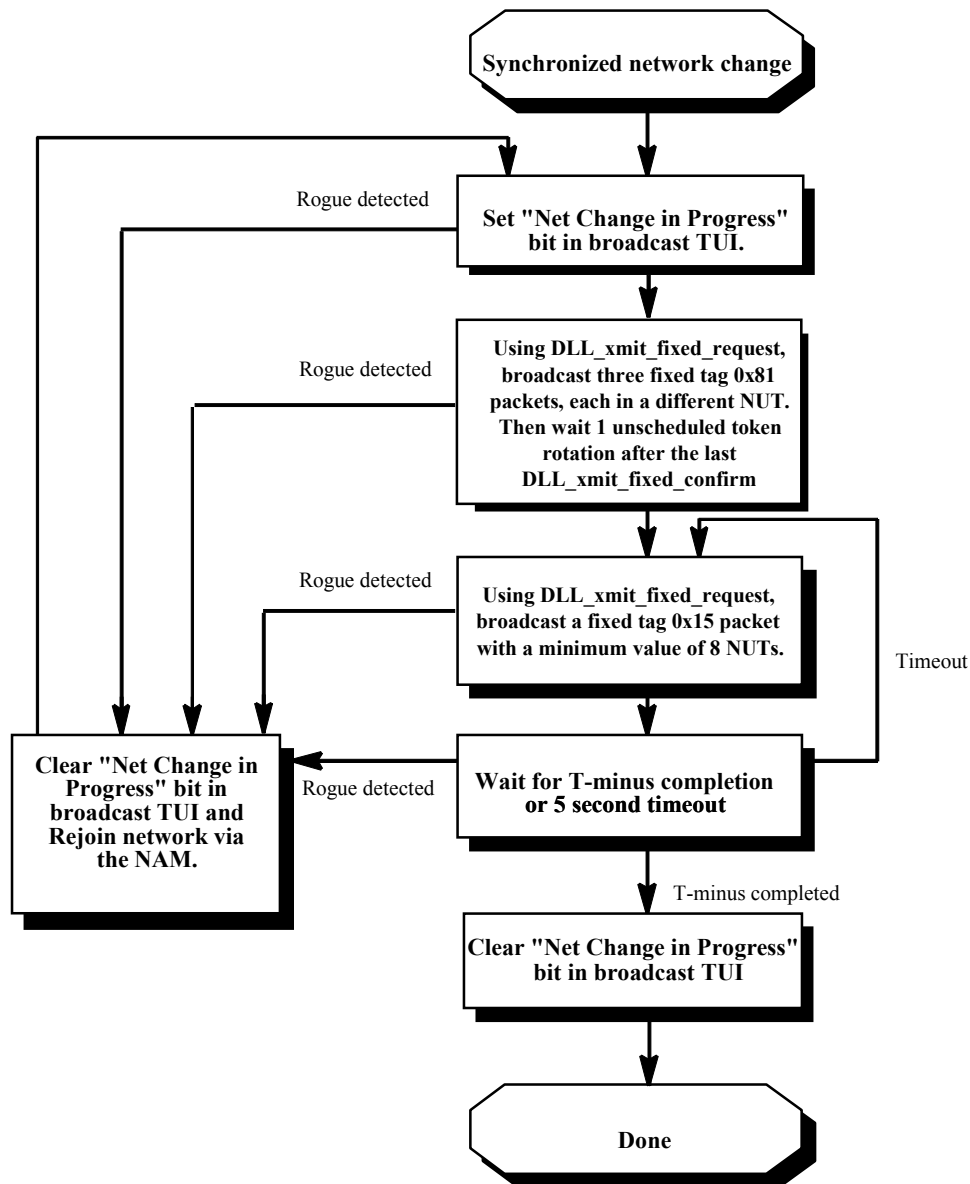


Figure 43 – Synchronized network change processing

NOTE 1 Broadcasting a TUI with the net change in progress bit set forces nodes that are just powering up to hold off from going on-line until after the network change completes and the net change in progress bit in the broadcast TUI is cleared. This reduces the chances of a rogue event occurring.

NOTE 2 The tMinus delay count provides noise immunity and insures that all nodes will hear at least one moderator with the new tMinus count. It also allows nodes a reasonable amount of time to process the Set_Single packets with new network attributes. The delay count is determined by taking the maximum of 25 or the result of dividing 500 ms by the current nut time in ms.

NOTE 3 The rogue condition will occur only if the moderator node and at least one other node did not receive one of the three broadcast set network attribute service request packets. This would only occur if there was a serious noise event (lasting at least 3 NUTs in duration).

17.4 Scheduling object

17.4.1 Overview

The Scheduling object shall exist in connection originator (CO) devices. The Scheduling object shall be used by link scheduling software (LSS) to:

- a) read scheduled connection information;
- b) write schedule information;
- c) provide the CO with a password (software key) which shall be used to authenticate the CO device's access to a specific link.

A LSS session with the Scheduling object schedules a particular link. If a CO device has connections on multiple links, multiple LSS sessions shall be needed to schedule all the connections. The results of the scheduling session shall be saved by the CO device (including the CO password computed by the LSS).

Since the LSS may be integrated with the programming software (PS) for a CO device, the Scheduling object shall provide commands which support integrated or a separate PS. Specifically, when the LSS writes Scheduling data to the Scheduling object, it may write conditionally or it may force the write. A conditional write of Scheduling data shall be used if the PS and LSS were not integrated and the Scheduling object shall only use the provided Scheduling data if a separate PS has not changed the connection information during the LSS session. A forced write may be used with an integrated LSS/PS if the integrated tool can guarantee that no other PS has changed connection information since it was last read during the current Scheduling session.

All communication to the Scheduling object shall use the Unconnected Message Manager (UCMM) fixed service, or an active Generic tag connection service to the Message Router object.

17.4.2 Attributes

17.4.2.1 Class attributes

The Scheduling object shall support the class attributes as specified in Table 166. The Get_Attribute_All service shall return the class level attributes in numerical order, smallest to largest.

Table 166 – Scheduling object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute
0x01	Required	Get Only	Revision	UINT	First revision, value = 1
0x02	Required	Get Only	MaxInstances	UDINT	maximum number of Scheduling objects supported
0x03	Required	Get Only	NumInstances	UDINT	number of instantiated Scheduling objects

17.4.2.2 Instance attributes

The Scheduling object shall support the instance attributes as specified in Table 167. The Get_Attribute_All service shall return the attribute level attributes in numerical order, smallest to largest.

Table 167 – Scheduling object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute
0x01	Required	Get/Set	Route	STRUCT of	route from CO to the link being scheduled
			NumSegments	UINT	number of segments in path
			Segments	ARRAY of UINT	array of segments
0x02	Required	Get/Set	TimeOut	UDINT	watchdog time-out μ s (default = 60 seconds)
0x03	Optional	Get/Set	Controller State	UINT	bit 0 (run/program) = 0, may be programmed = 1, currently controlling I/O bit 1 (remote/local) = 0, bit 0 may not be changed remotely = 1, bit 0 may be changed remotely bit 2-15 reserved and shall be zero

17.4.3 Services

17.4.3.1 Services supported

In addition to the services described in this subclause, the Scheduling object shall support the standard Get_Attribute_All service addressed to the class instance (instance zero). A specific Scheduling object implementation may optionally support other standard attribute services. The required services shall be as specified in Table 168.

An implementation may support setting the route instance-level attribute, but if it does, the internal behavior shall be indistinguishable from a delete service followed by a create service containing the new route.

Table 168 – Services at class level and instance level

Service	Code	Needed at class level	Needed at instance level
Get_Attribute_All	0x01	Required	Optional
Set_Attribute_All	0x02	N/A	Optional
Get_Attribute_List	0x03	Optional	Optional
Set_Attribute_List	0x04	N/A	Optional
Create	0x08	Required	Optional
Delete	0x09	N/A	Required
Get_Attribute_Single	0x0E	Optional	Optional
Set_Attribute_Single	0x10	N/A	Optional
Kick Timer	0x4B	N/A	Required
Read	0x4C	N/A	Required
Conditional Write	0x4D	N/A	Required
Forced Write	0x4E	N/A	Optional
Change Start	0x4F	N/A	Required
Break Connections	0x50	N/A	Required
Change Complete	0x51	N/A	Required
Restart Connections	0x52	Required	N/A

17.4.3.2 Create (0x08)

The create service shall be used to instantiate a Scheduling object on the CO. Within the create message, the client (LSS) shall include a route pointing from the CO to the link to be scheduled. The route to the link shall consist of port segments and shall not contain network segments. The Scheduling object shall reply with an instance number to which all further communication shall be addressed for this Scheduling session, and with the currently stored CO password and path.

The new instance shall delete itself if communications to its client are severed. Communications to an instance shall be monitored via a watchdog timer whose time-out value (default = 60 s) can be set via one of the optional set attribute services. The watchdog timer for an instance shall be reset when it receives any message. If the timer expires, the instance shall be deleted aborting all pending changes. The time-out value shall be a 32-bit integer with units of microseconds.

If the create service is addressed to the class level (instance zero), the instance number of the newly created instance shall be chosen automatically by the Scheduling object. For debug purposes, the Scheduling object may optionally allow the create service to be addressed to a specific instance number.

The Scheduling object shall support at least one instance. If the Scheduling object allows more than one instance to exist, it shall ensure the integrity of all its instances. The value of instance attribute 0x01 shall be provided with the create service.

```
class Scheduling_Create_Request : public MR_Request
{
    UINT    number_of_attributes;    // set to 1
    InstanceAttribute1 attribute1;
}

class Scheduling_Create_Response : public MR_Response
{
    UDINT    instance_number;
    UDINT    cop;                // Connection Originator Password (COP)
    UINT     path_size;          // in words, range 0 to 255
    UINT     path[];             // array of port segments from the scheduled link
                                // to the connection originator
                                // no key segments, no network segments
                                // include the MAC ID of the link hop
};
```

The response shall include one of the status codes shown in Table 169.

Table 169 – Status error descriptions for Create

General status	Extended status	Error description
0x02	0x0001	insufficient resource — maximum number of Scheduling object instances already exist
	0x0002	insufficient resource — not enough memory on CO device
0x04	N/A	path syntax error
0x08	N/A	unimplemented service (since non-zero instances are optional)
0x10	N/A	cannot open another instance due to internal conflicts
0x13	N/A	insufficient request data — request was too short or truncated
0x0D	N/A	object already exists (when non-zero instance specified)
0x1C	N/A	attribute list shortage — required attribute was missing
0xD0	N/A	no scheduled connections on this link

17.4.3.3 Delete (0x09)

The delete service shall be used to conclude a LSS Scheduling session. It deallocates all resources for a specified instance of the Scheduling object. Any pending changes which have not yet been committed shall be lost. Connections which have been broken by commands earlier in the LSS session shall not necessarily be restored. The delete service shall not be supported at the class level. The response of this service shall be as specified in Table 170.

Table 170 – Status error descriptions for Delete and Kick_Timer

General status	Extended status	Error description
0x00	N/A	success
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)

17.4.3.4 Kick_Timer (0x4B)

The kick_timer service shall reset the watchdog timer within an instance of the Scheduling object without otherwise affecting the state of the Scheduling object. The LSS shall issue a command at a rate of four per time-out period. For example, the LSS shall issue a service request every 15 seconds if the time-out value is set to the default 60 seconds. All other Scheduling object services, directed to this Scheduling instance, shall also reset the watchdog timer so the kick timer service need only be used if the client (LSS) wants to keep a Scheduling object instance alive but has nothing to say.

The watchdog timer for an instance shall be reset when it receives any message; however, it shall not start its countdown until the corresponding response is sent. This shall be to allow for a single task implementation. The response of this service shall be as specified in Table 170.

17.4.3.5 Read (0x4C)

The read service shall be used by the LSS to read information about the scheduled connections that the CO device desires to create on a specific link. The connection information shall come from an application object within the CO device.

A read service request shall contain a required UDINT parameter that specifies which connection index on which to start the read. The Scheduling object shall reply to a read service with as much connection information as can be fit into a reply Lpacket. The connection indexes within an Lpacket shall be ordered from smallest to largest (gaps shall be allowed). The extended status field within the reply shall indicate whether connections with higher indexes exist in the CO device. A client (LSS) shall continue to submit **read** requests with higher starting indexes until it reads all the connection information for the chosen link. An extended status of 0 shall indicate that more connections exist in the device. An extended status of 1 shall indicate that all connections have been read.

The reply shall consist of a UINT indicating the number of connections described in this response followed by a series of entries each describing a connection that meets the connection request criteria (the route to the link). Each entry shall have a unique Scheduling data index that is chosen by and shall primarily be used by the CO. The last one of these can be incremented by the LSS to continue the request if it is too long for a reply.

Other parameters provided in the connection entry shall describe the connection sizes, connection types (multi-cast/point to point) and connection update rates (RPIs). The connection entry route information shall consist of the route from the link to the target module for the connection including the connection points within the target module. Port information need not actually be used by the LSS.

This route shall also contain the current values assigned for the connection in the network segments. The network segments shall contain both API (Lpacket interval) and starting NUT information.

```
class Scheduling_Read_Request : public MR_Request
{
    UDINT first_connection_index;
}

class Scheduling_Read_Response : public MR_Response
{
    UINT number_of_connections;
    Scheduling_Connection_Description desc[];
}

class Scheduling_Connection_Description
{
    UDINT connection_index;
    UINT O2T_parameters;
    UINT T2O_parameters;
    UDINT O2T_RPI;
    UDINT T2O_RPI;
    UINT path_size;           // in words, range 0 to 255
    UINT path[];              // array of path segments
                                // first segment is always 0=>T schedule segment
                                // second is always port segment onto the link
                                // third is always T=>O schedule segment
                                // remaining path including logical segments
                                // and other port segments if multihop connection
};
```

The scheduling tool, LSS, shall use the logical segments of the target path to determine whether more than one connection request actually uses one multipoint producer. The rule for all objects except the Assembly object shall be that a match of class, instance, T⇒O connection point, and any finer granularity (for example attribute or member) constitutes a unique producer and it shall be scheduled once for both connections. Since the Assembly object equates instance and connection point, a match on connection path instance shall not determine whether the producer is unique.

For example, these two connection paths specify the same producer (instance 0x88 of the assembly object):

"20 04 24 03 2C 99 2C 88";

"20 04 24 AA 2C 77 2C 88".

These two connection paths specify different producers since the logical class segment do not match:

"20 77 24 03 2C 99 2C 88";

"20 77 24 AA 2C 99 2C 88".

The status response of this service shall be as specified in Table 171.

Table 171 – Status error descriptions for Read

General status	Extended status	Error description
0x00	N/A	success
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)
0x13	N/A	insufficient request data — request was too short or truncated

17.4.3.6 Conditional_Write (0x4D)

Conditional_Write service shall be used by the LSS to write the schedule information back to the CO device. Receipt of this message shall indicate that specific indexes of the connection information are being rescheduled by the LSS. A CO device may start breaking the current connections associated with the connection indexes, or it may wait until the receipt of a subsequent break connections service request.

If the new schedule provided by the LSS exactly matches the current schedule for a given connection, the CO device may ignore that part of the write request. A Scheduling object implementation may reply with an out of resource error if it receives updates for more connections than it has been queried about (via read requests). An implementation may also check for duplicate write requests; thereby, allowing retries — more write requests than read requests. The connection indexes may not be in order.

The difference between the conditional write service and the forced write service shall be that, for the conditional case, the CO device shall be required to verify the freshness of the connection information it previously sent to the LSS. If the data is not fresh, the CO device shall not use the Scheduling data for that connection index. In the forced case, the CO device need not check. “Fresh” connection information shall mean that nothing has changed the connection information since it was sent to the LSS via a read command.

The request shall be an array of connection schedules.

```
class Scheduling_Write_Request : public MR_Request
{
    UINT    number_of_connection_schedules;
    struct {
        UDINT connection_index;
        USINT  O2T_schedule;
        USINT  T20_schedule;
    } schedules[];
};
```

The response status of this service shall be as specified in Table 172.

Table 172 – Status error descriptions for Conditional_Write

General status	Extended status	Error description
0x00	N/A	success
0x03	N/A	invalid value — bad index contained in request
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)
0x0C	N/A	wrong state (change start not received)
0x10	N/A	device state conflict (device cannot break connection)
0x13	N/A	insufficient request data — request was too short or truncated
0x15	N/A	instance is unable to receive any more write requests

17.4.3.7 Forced_Write (0x4E)

Forced_Write service shall be similar to the conditional write service except that the LSS shall guarantee that it has provided valid connection information. The request parameters shall be the same as for the Conditional_Write service. The response status of this service shall be as specified in Table 173.

Table 173 – Status error descriptions for Forced_Write

General status	Extended status	Error description
0x00	N/A	success
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)
0x0C	N/A	wrong state (change start not received)
0x10	N/A	device state conflict (device cannot break connection)
0x13	N/A	insufficient request data — request was too short or truncated
0x15	N/A	instance is unable to receive any more write requests

17.4.3.8 Change_Start (0x4F)

The Change_Start service shall be used to initiate a Scheduling data change in the controller. The controller shall allocate enough memory for the new data and for the CO password and path. The **conditional write** or **forced write** service shall be then used to write the data to the controller.

The CO password shall be used when the controller first is powered up and begins to remake the connections. The CO shall check this CO password with the one stored in the Keeper object of the link prior to making any connections on that link.

The Change_Start request shall have only one parameter. The parameter shall be a UINT in the range 0 to 255 that specifies the size of the CO password and path to be provided in the Change_Complete service. Implementations may use this to allocate a receive buffer for the CO password and path.

The response status of this service shall be as specified in Table 174.

Table 174 – Status error descriptions for Change_Start

General status	Extended status	Error description
0x00	N/A	success
0x02	0x0002	insufficient resource — not enough memory on CO device
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)
0x10	N/A	device state conflict (for example, device cannot allocate sufficient memory)
0x13	N/A	insufficient request data — request was too short or truncated

17.4.3.9 Break_Connections (0x50)

Break_Connections service shall be used after the new Scheduling data has been written to the controller to break all the affected connections. The service shall reply only after all the connections have been broken. If any connections are in the process of being connected, they shall also be closed before replying to this service.

The Scheduling object break connections service shall break all connections that were specified in the Scheduling data write services. An LSS may always write Scheduling data for all connections (causing all connections over the link to be broken) but another LSS may be optimized to minimize the number of connections that get broken.

When the Scheduling object break connections service is received, in addition to breaking the changed connections, the CO shall NULL out (zero out) the currently active scheduled data values for the connections it breaks to prevent connections from being re-established until the Scheduling data change complete service is received. If the change complete service is never received (due to loss of communications with LSS) these connections shall not be able to be opened until they are rescheduled. The response of this service shall be as specified in Table 175.

Table 175 – Status error descriptions for Break_Connections

General status	Extended status	Error description
0x00	N/A	success
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)
0x0C	N/A	object state conflict
0x10	N/A	device state conflict (device cannot break connections)
0x13	N/A	insufficient request data — request was too short or truncated

17.4.3.10 Change_Complete (0x51)

Change_Complete service shall be used to indicate that all connections have been broken and it is permitted to start making the new connections. The request contains the new CO password and path to be used for the link that has been scheduled. The CO shall check this CO password with the one stored in the Keeper object of the link prior to making any connection on that link.

```
class Scheduling_Change_Complete_Request : public MR_Request
{
    UDINT  cop;           // Connection Originator Password (COP)
    UINT   path_size;     // in words, range 0 to 255
    UINT   path[];        // array of port segments from the scheduled link
                        // to the connection originator
                        // no key segments, no network segments
                        // include the MAC ID of the link hop
};
```

NOTE The reverse path included in this request allows the LSS to detect changes in the network position of the scheduled CO during later scheduling sessions.

The response status of this service shall be as specified in Table 176.

Table 176 – Status error descriptions for Change_Complete

General status	Extended status	Error description
0x00	N/A	success
0x03	N/A	invalid value (new CO password and path is not formatted correctly)
0x04	N/A	path syntax error
0x05	N/A	instance undefined
0x08	N/A	unimplemented service (if directed to instance 0)
0x10	N/A	device state conflict
0x13	N/A	insufficient request data — request was too short or truncated
0x0C	N/A	wrong state

17.4.3.11 Restart_Connections (0x52)

The Restart_Connections service may only be sent to the class instance of the Scheduling object (instance zero). It shall be used to break all connections which use a specific link. After all the connections have been broken, the CO device shall attempt to re-establish them. The request parameter for this service shall be structured as is instance attribute 0x01. The response status of this service shall be as specified in Table 177.

Table 177 – Status error descriptions for Restart_Connections

General status	Extended status	Error description
0x00	N/A	success
0x04	N/A	path syntax error
0x08	N/A	unimplemented service (if addressed to a non-zero instance)
0x10	N/A	device state conflict (cannot break connections)
0x13	N/A	insufficient request data — request was too short or truncated
0xD0	N/A	no scheduled connections on this link

17.4.4 Typical Scheduling session

A typical scheduling session shall use these services:

- a) create;
- b) read;
- c) write;
- d) change_start;
- e) break_connections;
- f) conditional_write;
- g) forced_write;
- h) change_complete.

The following steps describe the usage of services in a typical Scheduling session:

- 1) The LSS shall begin by sending a **create** message to instance zero of the Scheduling object. Contained in this message shall be a path from the CO to the link. This path shall determine which connections are of interest to the LSS.

- 2) The Scheduling object shall create an instance of itself to service this LSS and reply with the instance number to which all future communications in this session shall be sent.
- 3) The newly created instance shall reset a 60 second timer upon receipt of any message. If this timer ever expires, the instance shall delete itself aborting any change in progress. In normal operation, the LSS shall be required to send a command to the Scheduling object at least once every 15 seconds allowing a few dropped messages without unnecessarily deleting an instance.
- 4) The LSS may optionally send **read** commands to the newly created instance to transfer the connection information from the CO device. The transmitted connection information shall contain indexes which are used to uniquely identify the connections internally within the CO device.
- 5) The LSS shall reflect back these indexes on subsequent **write** commands so that the CO may determine for which connections the **write** applies. The issuing of **read** commands shall be optional since the connection information can be obtained for the LSS by other means. For example, in an integrated LSS/PS, the PS shall provide this information to the LSS.
- 6) Once the connection information is acquired from each CO desiring scheduled data on the link, the LSS shall calculate a tentative schedule.
- 7) The LSS then shall issue the following sequence of commands to each of the CO devices: **change start**, one or more **writes**, and **break connections**. This sequence can be done in parallel to each of the CO devices. The type of **write** command (forced or conditional) shall depend on how the connection information was obtained. If the LSS has independently ensured that the connection information is valid, it may use the **forced write** command; otherwise, it shall use a **conditional write** command. The LSS can know that the connection information is valid by obtaining the edit resource for the CO before the connection information.
- 8) Upon receipt of a **conditional write** command, the Scheduling object shall check that the parameters of the connection have not changed since they were last read. If the connection has been changed, the Scheduling object shall not use the Scheduling information for the index whose connection parameters have changed. The Scheduling object within the CO can do this through any means it chooses.
- 9) The Scheduling object shall be absolved of responsibility to check the validity of **forced writes**. This responsibility shall be assumed by the LSS.
- 10) Once the **break connection** commands to each of the CO devices has completed successfully, the LSS shall issue a **change complete** to each of the CO devices indicating that the schedule transmitted earlier via **writes** is now valid. The **change complete** command shall include the new CO password and path.
- 11) The session shall end by deleting the instance of the Scheduling object.

17.5 TCP/IP interface object

17.5.1 Overview

The TCP/IP interface object provides the mechanism to configure the TCP/IP interface of a device.

NOTE 1 Examples of configurable items include the device's IP Address, Network Mask, and Gateway Address.

The physical port associated with the TCP/IP interface object shall be any port supporting the TCP/IP protocol.

NOTE 2 For example, a TCP/IP interface object may be associated any of the following : an Ethernet 802.3 port, an ATM port, a serial port running SLIP, a serial port running PPP.

The TCP/IP interface object provides an attribute that identifies the link-specific object for the associated physical port. The link-specific object is generally expected to provide link-specific counters as well as any link-specific configuration attributes.

Each device shall support exactly one instance of the TCP/IP interface object for each TCP/IP-capable port on the device. A request to access instance 1 of the TCP/IP interface object shall always refer to the instance associated with the port over which the request was received.

17.5.2 Attributes

17.5.2.1 Class attributes

The TCP/IP interface object shall support the class attributes as specified in Table 178.

Table 178 – TCP/IP interface object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x01	Optional	Get	revision	UINT	revision of this object	first revision, value = 1

17.5.2.2 Instance attributes

17.5.2.2.1 General

The TCP/IP interface object shall support the instance attributes as specified in Table 179.

Table 179 – TCP/IP interface object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x01	Required	Get	status	DWORD	Interface status	See 17.5.2.2.2
0x02	Required	Get	configuration capability	DWORD	Interface capability flags	Bit map of capability flags. See 17.5.2.2.3
0x03	Required	Get/Set	configuration control	DWORD	Interface control flags	Bit map of control flags. See 17.5.2.2.4
0x04	Required	Get	Physical link object	STRUCT of:	Path to physical link object	See 17.5.2.2.5
			path size	UINT	Size of path	Number of UINTs in Path
			path	ARRAY of UINT	Logical segments identifying the physical link object	Class Segment and Instance Segment. Maximum length is 6 UINTs (if 32 bit logical segments are used).
0x05	Required	Get/Set	interface configuration	STRUCT of:	TCP/IP network interface configuration	See 17.5.2.2.6
			IP Address	UDINT	Device's IP address.	Value of 0 indicates no IP address has been configured. Otherwise, the IP address shall be set to a valid class A, B, or C address and shall not be set to the loopback address (127.0.0.1).

Attribute ID 0x0.5 <i>continued</i>	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			network mask	UDINT	Device's network mask	Value of 0 indicates no network mask address has been configured.
			gateway address	UDINT	gateway address	Value of 0 indicates no IP address has been configured. Otherwise, the IP address shall be set to a valid class A, B, or C address and shall not be set to the loopback address (127.0.0.1).
			name server	UDINT	Primary name server	Value of 0 indicates no name server address has been configured. Otherwise, the name server address shall be set to a valid class A, B, or C address.
			name server 2	UDINT	Secondary name server	Value of 0 indicates no secondary name server address has been configured. Otherwise, the name server address shall be set to a valid class A, B, or C address.
			domain name	STRING	Default domain name	ASCII characters. Maximum length is 48 characters. Shall be padded to an even number of characters (pad not included in length). A length of 0 shall indicate no Domain Name is configured.
0x06	Required	Get/Set	host Name	STRING	Host name	ASCII characters. Maximum length is 64 characters. Shall be padded to an even number of characters (pad not included in length). A length of 0 shall indicate no Host Name is configured. See 17.5.2.2.7

17.5.2.2.2 Status

The status attribute shall indicate the status of the TCP/IP network interface, as specified in Table 180. Refer to the state diagram in Figure 44 for a description of object states.

Table 180 – Status values

Value	Meaning
0x00000000	TCP/IP network interface is not configured
0x00000001	TCP/IP network interface is configured, and interface configuration attribute fields are valid.
0x00000002 – 0xFFFFFFFF	Reserved for future use

17.5.2.2.3 Configuration capability

The configuration capability attribute is a bitmap that indicates the device's support for optional network configuration capability. Possible configuration capability items are listed in Table 181. Devices are not required to support any one particular item, however they shall support at least one method of obtaining an initial IP address.

Table 181 – Configuration capability bits

Bit	Name	Definition
0	BOOTP client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via BOOTP.
1	DNS client	1 (TRUE) shall indicate the device is capable of resolving host names by querying a DNS server.
2	DHCP client	1 (TRUE) shall indicate the device is capable of obtaining its network configuration via DHCP.
3	DHCP-DNS update	1 (TRUE) shall indicate the device is capable of sending its host name in the DHCP request as documented in Internet draft <draft-ietf-dhc-fqdn-option-01.txt>.
4	Configuration settable	1 (TRUE) shall indicate the interface configuration attribute is settable. Some devices, for example a PC or workstation, may not allow the interface configuration to be set via the TCP/IP interface object.
5-31	Reserved	Reserved for future use and shall be set to zero.

17.5.2.2.4 Configuration control

The configuration control attribute is a bitmap used to control network configuration options, as specified in Table 182.

Table 182 – Configuration control bits

Bit	Name	Definition
0-3	Startup Configuration	Determines how the device shall obtain its initial configuration at start up. 0 = The device shall use the network configuration previously stored (for example, in non-volatile memory or via hardware switches, etc). 1 = The device shall obtain its network configuration via BOOTP. 2 = The device shall obtain its network configuration via DHCP upon start-up. 3-15 = Reserved for future use.
4	DNS Enable	If 1 (TRUE), the device shall resolve host names by querying a DNS server.
5-31	Reserved	Reserved for future use and shall be set to zero.

Devices are not required to support any of the particular values of the Startup Configuration bits, however a device must support at least one method of obtaining an initial TCP/IP interface configuration. For example, some low-end devices may choose to obtain network interface configuration via BOOTP or DHCP only. Other devices may wish to only allow initial configuration over a serial (or other) port, then store the configuration in non-volatile memory.

The use of BOOTP and/or DHCP may not be appropriate for some devices. DHCP in particular supports dynamically allocated IP addresses, which could result in a device getting a different IP address each time it powers up. This behavior is not appropriate for devices that need to have static IP addresses.

Out of the box, a device may wish to obtain its initial configuration via some method other than BOOTP or DHCP. For example, the device may wish to obtain its initial configuration over an attached serial port, or via another communications port compatible with the corresponding Type 2 Application Layer protocol (assuming the device has another communications port). In this case, the device shall have its startup configuration bits set to 0, and its interface configuration attribute fields to all 0s. The device shall then wait to be configured over another communications port.

Once a device is up and running, when the value of the startup configuration bits is 0, a request to set the interface configuration attribute shall cause the device to store the contents of the interface configuration attribute in non-volatile storage if supported by the device. The startup configuration bits shall not be set to 0 unless the interface configuration attribute has previously been set. Otherwise the device could be rendered unable to communicate on the network.

17.5.2.2.5 Physical link object

This attribute identifies the object associated with the underlying physical port. There are two components to the attribute : a path size (in UINTs) and a path. The path shall contain a class segment and an instance segment that identify the physical port object. The maximum path size is 6 (assuming a 32 bit logical segment for each of the class and instance). An example path is shown in Table 183.

Table 183 – Example path

Path	Meaning
[20][f6][24][01]	[20] = 8 bit class segment type; [f6] = Ethernet link object class; [24] = 8 bit instance segment type; [01] = instance 1.

The physical link object itself typically maintains link-specific counters as well as any link-specific configuration attributes. The most likely physical link object is the Ethernet link object (see 17.6).

17.5.2.2.6 Interface configuration

This attribute contains the configuration parameters required to operate as a TCP/IP node. In order to prevent incomplete or incompatible configuration, the parameters making up the interface configuration attribute cannot be set individually. To modify the Interface configuration attribute, the user should first get the interface configuration attribute, change the desired parameters, then set the attribute.

The TCP/IP interface object shall apply the new configuration upon completion of the Set service. If the value of the startup configuration bits (configuration control attribute) is 0, the new configuration shall be stored in non-volatile memory. The device shall not reply to the set service until the values are safely stored to non-volatile storage.

If initial configuration is to be obtained via BOOTP or DHCP, the interface configuration attribute components shall be all zeros until the BOOTP or DHCP reply is received. Upon receipt of the BOOTP or DHCP reply, the interface configuration attribute shall show the configuration obtained via BOOTP/DHCP.

Devices are not required to support the Set service. Some implementations, for example those running on a PC or a Workstation, need not support setting the network interface configuration via the TCP/IP interface object.

Components of the interface configuration attributes are described in Table 184.

Table 184 – Interface configuration components

Name	Meaning
IP address	The device's IP address.
Network mask	The device's network mask. The network mask is used when the IP network has been partitioned into subnets. The network mask is used to determine whether an IP address is located on another subnet.
Gateway address	The IP address of the device's default gateway. When a destination IP address is on a different subnet, packets are forwarded to the default gateway for routing to the destination subnet.
Name server	The IP address of the primary name server. The name server is used to resolve host names. For example, that might be contained in a fieldbusconnection path.
Name server 2	The IP address of the secondary name server. The secondary name server is used when the primary name server is not available, or is unable to resolve a host name.
Domain name	The default domain name. The default domain name is used when resolving host names that are not fully qualified. For example, if the default domain name is "network.org", and the device needs to resolve a host name of "plc", then the device will attempt to resolve the host name as "plc.network.org".

17.5.2.2.7 Host Name

The Host Name attribute contains the device's host name. The host name attribute is used when the device supports the DHCP-DNS Update capability and has been configured to use DHCP upon start up. The mechanism allows the DHCP client to transmit its host name to the DHCP server. The DHCP server then updates the DNS records on behalf of the client.

NOTE The DHCP-DNS Update mechanism is specified in Internet draft <draft-ietf-dhc-fqdn-option-01.txt>.

The host name attribute does not need to be set for the device to operate normally. The value of the Host Name attribute, if it is configured, shall be used for the value of the FQDN option in the DHCP request. If the Host Name attribute has not been configured then the device shall not include the FQDN option in the DHCP request.

For devices that do not support the DHCP-DNS capability, or are not configured to use DHCP, then the host name may be used for informational purposes.

17.5.3 Services

17.5.3.1 Common services

The TCP/IP interface object shall support the common services as specified in Table 185.

Table 185 – TCP/IP interface object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x01	Optional	Optional	Get_Attributes_All	Returns a predefined listing of this objects attributes (see the Get_Attribute_All response definition in 17.5.3.2)
0x02	N/A	Optional	Set_Attributes_All	Modifies all settable attributes.
0x0E	Optional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.
0x10	N/A	Required	Set_Attribute_Single	Modifies a single attribute.

17.5.3.2 Get_Attribute_All response

For class attributes, since there is only one class attribute, class attribute #1 shall be returned.

For instance attributes, attributes shall be returned in numerical order. The Get_Attribute_All reply shall be as specified in Table 186.

Table 186 – Get_Attribute_All reply format

Attribute ID	Size (octets)	Contents
1	4	Status
2	4	Configuration capability
3	4	Configuration control
4	2	Physical link object, path size
	variable, 12 octets max	Physical link object, path (if path size is non-zero)
5	4	IP address
	4	Network mask
	4	Gateway address
	4	Name server
	4	Secondary name server
	2	Domain name length
	variable, equal to domain name length	Domain name
	1	Pad octet only if Domain Name Length is odd
6	2	Host name length
	variable, equal to host name length	Host name
	1	Pad octet only if Host Name Length is odd

The lengths of the physical link object path, domain name, and host name are not known before issuing the Get_Attribute_All service request. Implementers shall be prepared to accept a response containing the maximum sizes of the physical link object path (6 UINTs), the domain name (48 USINTs), and host name (64 USINTs).

17.5.3.3 Set_Attribute_All request

The instance Set_Attribute_All request shall contain the configuration control attribute, followed by the interface configuration attribute.

17.5.4 Behavior

The behavior of the TCP/IP interface object shall be as illustrated in the State Transition Diagram shown in Figure 44.

Note that the act of obtaining an initial executable image via BOOTP/TFTP shall not be considered within the scope of the TCP/IP interface object behavior. Devices are free to implement such behavior, however it shall be considered to have occurred in the “Non-existent” state.

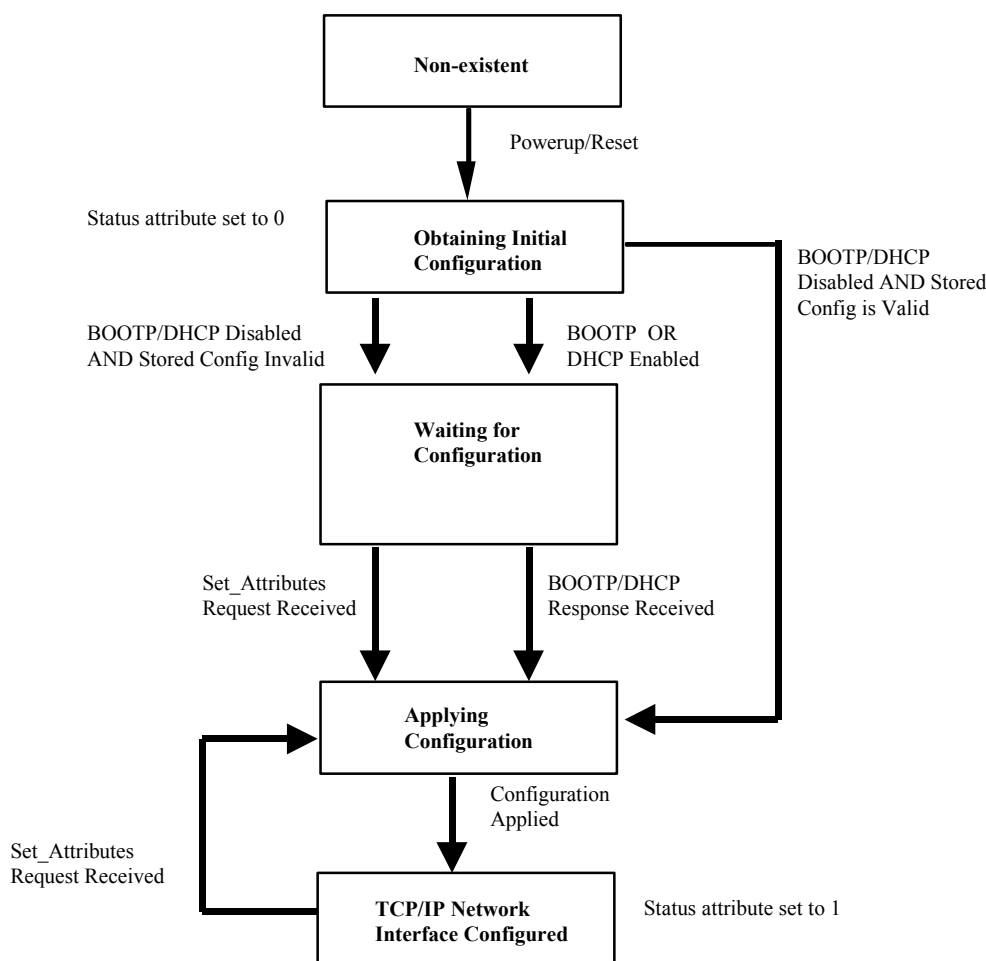


Figure 44 – State transition diagram for TCP/IP interface object

17.6 Ethernet link object

17.6.1 Overview

The Ethernet link object maintains link-specific counters and status information for a physical Ethernet ISO/IEC 8802-3 port. Each device shall support exactly one instance of the Ethernet link object for each Ethernet port on the module. A request to access instance 1 of the Ethernet link object shall always refer to the instance associated with the port over which the request was received.

17.6.2 Attributes

17.6.2.1 Class attributes

The Ethernet link object shall support the class attributes as specified in Table 187.

Table 187 – Ethernet link object class attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x01	Optional	Get	revision	UINT	revision of this object	first revision, value = 1

17.6.2.2 Instance attributes

17.6.2.2.1 General

The Ethernet link object shall support the instance attributes as specified in Table 188.

Table 188 – Ethernet link object instance attributes

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
0x01	Required	Get	Interface speed	UDINT	Speed of the interface	Speed in Megabits per second (10, 100, 1000, etc.). See 17.6.2.2.2
0x02	Required	Get	Interface flags	DWORD	Interface status flags	Bit map of interface flags. See 17.6.2.2.3
0x03	Required	Get	Physical address	ARRAY of 6 USINTs	MAC layer address	See 17.6.2.2.4
0x04	Conditional	Get/ Get_and _Clear	Interface counters	STRUCT of:		See 17.6.2.2.5
			In Octets	UDINT	Octets received on the interface	
			In Ucast Packets	UDINT	Unicast packets received on the interface	
			In NUcast Packets	UDINT	Non-unicast packets received on the interface	
			In Discards	UDINT	Inbound packets received on the interface but discarded	
			In Errors	UDINT	Inbound packets that contain errors (does not include In Discards)	
			In Unknown Protos	UDINT	Inbound packets with unknown protocol	
			Out Octets	UDINT	Octets sent on the interface	
			Out Ucast Packets	UDINT	Unicast packets sent on the interface	

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			Out NUCast Packets	UDINT	Non-unicast packets sent on the interface	
			Out Discards	UDINT	Outbound packets discarded	
			Out Errors	UDINT	Outbound packets that contain errors	
0x05	Conditional	Get/Set	Media counters	STRUCT of:	Media-specific counters	See 17.6.2.2.6
			Alignment errors	UDINT	frames received that are not an integral number of octets in length	
			FCS errors	UDINT	DLPDUs received that do not pass the FCS check	
			Single collisions	UDINT	Successfully transmitted DLPDUs which experienced exactly one collision	
			Multiple collisions	UDINT	Successfully transmitted DLPDUs which experienced more than one collision	
			SQE test errors	UDINT	Number of times SQE test error message is generated	
			Deferred transmissions	UDINT	DLPDUs for which first transmission attempt is delayed because the medium is busy	
			Late collisions	UDINT	Number of times a collision is detected later than 512 bit-times into the transmission of a packet	
			Excessive collisions	UDINT	DLPDUs for which transmission fails due to excessive collisions	
			MAC transmit errors	UDINT	DLPDUs for which transmission fails due to an internal MAC sublayer transmit error	
			Carrier sense errors	UDINT	Times that the carrier sense condition was lost or never asserted when attempting to transmit a frame	
			Frame too long	UDINT	DLPDUs received that exceed the maximum permitted DLPDU size	

Attribute ID	Need in implementation	Access rule	Name	Data type	Description of attribute	Semantics of values
			MAC receive errors	UDINT	DLPDUs for which reception on an interface fails due to an internal MAC sublayer receive error	
NOTE The interface counters is an optional attribute, however it shall be implemented if the media counters attribute is implemented.						

17.6.2.2.2 Interface speed

The interface speed attribute shall indicate whether the interface is running at 10 Mbit/s, 100 Mbit/s, 1 Gb/s, or other speeds. The scale of the attribute is in Mbit/s, so if the interface is running at 100 Mbit/s then the value of the interface speed attribute shall be 100. The interface speed is intended to represent the available link transmit time ; the attribute shall not be doubled if the interface is running in full-duplex mode.

17.6.2.2.3 Interface flags

The interface flags attribute contains status and configuration information about the physical interface as specified in Table 189.

Table 189 – Interface flags bits

Bit	Name	Definition
0	Link status	Indicates whether or not the port is connected to an active network : - 0 indicates an inactive link, - 1 indicates an active link. The determination of link status is implementation specific. In some cases devices can tell whether the link is active via hardware/driver support. In other cases, the device may only be able to tell whether the link is active by the presence of incoming packets.
1	Half/Full duplex	0 indicates the port is running half duplex, 1 indicates full duplex. If the Link status flag is 0, then the value of the Half/Full duplex flag is indeterminate.
2-31	Reserved	Reserved and shall be set to zero.

17.6.2.2.4 Physical address

The physical address attribute contains the interface's MAC layer address. The physical address is an array of octets. The recommended display format is "XX-XX-XX-XX-XX-XX", starting with the first octet. The physical address is not a settable attribute : the Ethernet address shall be assigned by the manufacturer, and shall be unique per ISO/IEC 8802-3 requirements.

17.6.2.2.5 Interface counters

The interface counters attribute contains counters relevant to the receipt of packets on the interface. These counters shall be as defined in RFC 1213 "MIB-II Management Information Base". The interface counters is an optional attribute, however it shall be implemented if the media counters attribute is implemented.

17.6.2.2.6 Media counters

The media counters attribute contains counters specific to Ethernet media. These counters shall be as defined by RFC 1643, “Definitions of Managed Objects for Ethernet-Like Interface Types”. The media counters is an optional attribute, however if it is implemented the interface counters attribute shall also be implemented.

17.6.3 Services

17.6.3.1 Common services

The Ethernet link object shall support the common services as specified in Table 190.

Table 190 – Ethernet link object common services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x01	Optional	Optional	Get_Attributes_All	Returns a predefined listing of this objects attributes (see the Get_Attribute_All response definition in 17.6.3.2)
0x0E	Conditional	Required	Get_Attribute_Single	Returns the contents of the specified attribute.

The Get_Attribute_Single shall be implemented for the class attribute if the class attribute is implemented.

17.6.3.2 Get_Attribute_All response

For class attributes, since there is only one possible attribute, the Get_Attribute_All response is the same as the Get_Attribute_Single response. If no class attributes are implemented, then no data is returned in the data portion of the reply.

For instance attributes, attributes shall be returned in numerical order, up to the last implemented attribute.

17.6.3.3 Class specific services

The Ethernet link object shall support the class specific services as specified in Table 191.

Table 191 – Ethernet link object class specific services

Service code	Need in implementation		Service name	Description of service
	Class	Instance		
0x4C	N/A	Conditional	Get_and_Clear	Gets then clears the specified attribute (interface counters or media counters).

The Get_and_Clear service shall only be implemented if the interface counters and media counters attributes are implemented.

17.6.3.4 Get_and_Clear service

The Get_and_Clear service is a class specific service. It is only supported for the interface counters and media counters attributes. The Get_and_Clear response shall be the same as the Get_Attribute_Single response for the specified attribute. After the response is built, the value of the attribute shall be set to zero.

18 Type 2: Other DLE elements of procedure

18.1 Network Attachment Monitor (NAM)

18.1.1 General

The network attachment monitor (NAM) shall control the DLL to allow new nodes to non-disruptively join a working link.

The NAM controls attaching the DLL to an existing link without causing disruption to transmissions on that link. It accomplishes this by monitoring and controlling the DLL via the Station Management and the normal send/receive interfaces to the DLL as summarized in Table 192 and Figure 45.

Table 192 – NAM states

State	Actions
Check for Cable	1) auto-address nodes shall find an address before entering (see 18.1.3) 2) use default parameters (see 18.1.2) 3) transmit DLPDUs, but no moderators 4) if receive rogue moderator, go to “Check for Moderator” 5) if receive any good DLPDU, transmit one more DLPDU, then go to “Wait to Rogue”.
Wait to Rogue	1) use default parameters (see 18.1.2) 2) transmit no DLPDUs 3) if hear rogue moderator, go to “Check for Moderator” 4) if hear no rogue moderator for 400 – 600 ms, go to “I’m Alive”
Check for Moderator	1) transmit no DLPDUs 2) if hear 9 identical valid moderators in successive NUTs, go to “I’m Alive” using the parameters in those moderators 3) if have not heard a moderator for 1,0 – 3,0 seconds, go to “Check for Cable” 4) if receive moderator with invalid link parameters, stay in this state, but change network status indicator to invalid link configuration (flashing red / green)
I’m Alive	1) auto-address nodes shall find an address before entering (see 18.1.3) 2) transmit DLPDUs, including moderators if this node has the lowest MAC ID 3) transmit three fixed tag 0x80 Lpackets 4) after sending them go to “Attached” 5) if receive rogue moderator, go to “Check for Moderator”
Attached	1) normal network operation 2) transmit DLPDUs, including moderators if this node has the lowest MAC ID 3) if receive rogue moderator, go to “Check for Moderator” 4) if no other nodes on link for 8 NUTs, go to “Check for Cable”

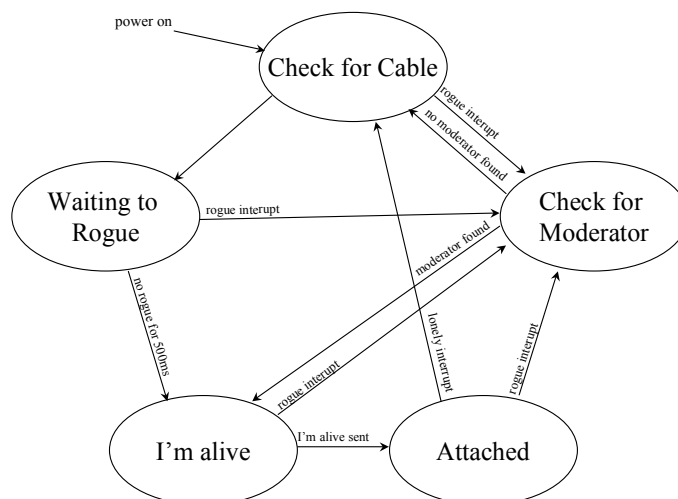


Figure 45 – NAM state machine

18.1.2 Default parameters

The default link parameters for a node shall be as shown in Table 193:

Table 193 – Default link parameters

Parameter	Data type	Value
NUT_length	UINT	100 ms (10000 intervals of 10 µs each)
smax	USINT	0
umax	USINT	99
slotTime	USINT	254 (255 µs)
blanking	USINT	6 (octet times)
gb_start	USINT	610 µs (61 intervals of 10 µs each)
gb_center	USINT	450 µs (45 intervals of 10 µs each)
modulus	USINT	127
gb_prestart	USINT	920 µs (92 intervals of 10 µs each)

18.1.3 Auto-addressing

Some nodes may have no pre-assigned MAC ID. These nodes, known as auto-address nodes, shall search for an unused MAC ID by observing the link. The auto-address node shall not transmit until a free MAC ID is found.

NOTE Typically, auto-address nodes are transient nodes such as hand-held terminals.

18.1.4 Valid MAC IDs

On a link with default configuration parameters, an auto-address node shall search in the range 92 through 99. On a link with any other configuration parameters, an auto-address node shall search in the range SMAX+1 through UMAX. A free MAC ID shall be declared found if at least three sequential unscheduled transmit opportunities for that MAC ID have passed without receiving any DLPDUs from that MAC ID.

NOTE The ControlNet Object provides an interface to determine the MAC ID which has been chosen.

18.1.5 State machine description

The following state machine description shall define the behavior of the network attachment monitor:

```
// Network Attachment Monitor state machine description
//
// This state machine monitors and controls the DLL to make sure that it goes on-line
// in a fashion that does not disrupt a running network.
//
//
// Lpacket constants: masks for ctl octet
//
#define FIXEDSCREEN 1
#define TAGPAD 2

#define DATAPAD 4
#define OCTETWORD 8
//
// moderator Lpacket constants
//
#define MODERATOR_SIZE 9      // moderator Lpacket size octet
#define MODERATOR_CTL 1      // moderator Lpacket control octet
#define MODERATOR_TAG 0      // moderator Lpacket tag octet

//
// Lpacket class
//
class Lpacket
{
public:
    USINT size;          // return the size octet
    USINT ctl;           // return the control octet

    // return the value of the tag pad bit
    int tag_pad(void)
    {
        return (ctl&TAGPAD) >>1;
    }
    // return the value of the data pad bit
    int data_pad(void)
    {
        return (ctl&DATAPAD) >>2;
    }
    // return the number of octets in the Lpacket
    int wire_size(void)
    {
        return size*2 - tag_pad() - data_pad();
    }
    // get the next octet to be transmitted
    USINT get_next_octet(void);

    // is this Lpacket aborted?
    BOOL abort(void);
};

class fixedLpacket : public Lpacket
{
public:
    USINT tag;
    USINT dest;
};

class moderatorLpacket : public fixedLpacket
{
public:
    USINT NUT_length;
    USINT smax;
    USINT umax;
    USINT slotTime;
    USINT blanking;
    USINT gb_start;
    USINT gb_center;
    USINT usr;
    USINT interval_count;
    USINT modulus;
    USINT tMinus;
    USINT gb_prestart;
    USINT spare;
};
```

```

    BOOL isValid(void);
        // this checks for umax<myaddr, and similar parameter
        // error that would prevent a node from successfully
        // joining a network

    BOOL operator==(moderatorLpacket &);
        // the equality test for moderator Lpackets excludes
        // the interval count, and usr, but includes the
        // tMinus counter
};

//
// interface to station management
//
BOOL SM_powerup;           // input indication: powerup has occurred
void SM_LEDS(LED_STATE);  // set the LEDS to a specified pattern
#define LED_bad_network_parameters // flashes the network status indicator

class DLL_config_data
{
public:
    BOOL listen_only;
    USINT myaddr;
    UINT NUT_length;
    USINT smax;
    USINT umax;
    USINT slotTime;
    USINT blanking;
    USINT gb_start;
    USINT gb_center;
    USINT interval_count;
    USINT modulus;
    USINT gb_prestart;
};

//
// interface to DLL
//
void DLL_xmit_fixed_request(char *comment);
void DLL_xmit_fixed_confirm(void);
void DLL_recv_fixed_indication();

//
// These timer objects are as used internally to the ACM of the DLL.
//
// various behaviors for timers
enum {ONCE_UP, ONCE_DOWN, REPEAT_UP, REPEAT_DOWN} timer_mode;

```

```

class timer
{
public:

float count;           // number of ticks left
float preset;          // initialize ticks, or cycle length
BOOL expire;           // latched flag when timer expires
timer_mode mode;

// constructor: defines mode
timer(timer_mode m)
{
    mode = m;
}

void restart(void)
{
    if(mode == REPEAT_DOWN
        || mode == ONCE_DOWN) // if counting down, start with preset
    {
        count = preset;
    }
    else
    {
        count = 0;           // else start from zero
    }
    // start counting
}

void start(float interval) // start counting down from interval towards zero
{
    mode = ONCE_DOWN;
    preset = interval;      // set interval
    restart();              // start counting
}

void begin_counting(void) // start counting up from zero
{
    mode = ONCE_UP;
    restart();
}

bool expired(void)        // returns true if the timer has expired; clears flag
{
    BOOL tmp;
    tmp = expire;
    expire = 0;
    return tmp;
}
};

//
// time conversion factors
//
#define usec
#define msec *1000
#define sec *1000000

class timer timer(ONCE_DOWN); // general purpose timer
int counter;                  // general purpose counter
moderatorLpacket new_mod;     // buffer for getting new moderator Lpackets
moderatorLpacket old_mod;     // buffer for saving moderator Lpackets

DLL_config_data default_config; // DLL config parameters for default mode
DLL_config_data DLL_config;     // scratch DLL config buffer

// Wait for powerup.
// Bring the DLL on-line.
// It is assumed that the node's MAC ID is known before this machine is allowed to
// power-up.
// If this node is auto-addressed, a free MAC ID shall be determined
// before enabling this machine.

```

```

state: powerup0
    event:    SM_powerup
    destination: checkForCable0
    action:
        DLL_set_current_request(default_config); // set default parameters

state: checkForCable0
    event:    DLL_set_current_confirm() // default parameters were set
    destination: checkForCable1
    action:
        DLL_enable_moderator_request(FALSE); // disable moderators
        DLL_online_request(TRUE); // send DLL on-line

state: checkForCable1
    event:    DLL_online_confirm() // DLL is now on-line, but can't moderate
    destination: checkForCable2

state: checkForCable2
    // rogue -> checkForModerator
    event:    DLL_SM_report(rogue)
    destination: checkForModerator0
    action:
        DLL_listen_only_request(TRUE); // send DLL to listen only
        DLL_enable_fixed_request(MODERATOR_TAG); // enable reception of moderators

    event:
        || DLL_rcv_gen_indication // any Lpacket received
    destination: waitForRogue
    action:
        wait until transmit once more;
        DLL_listen_only_request(TRUE); // send DLL to listen only
        DLL_enable_moderator_request(FALSE); // re-enable moderators
        timer.start(500 msec);

state: waitForRogue
    // rogue -> checkForModerator
    event:    DLL_SM_report(rogue)
    destination: checkForModerator0
    action:
        DLL_listen_only_request(TRUE); // send DLL to listen only
        DLL_enable_fixed_request(MODERATOR_TAG); // enable reception of moderators

    event:    timer.expired()
    destination: alive0
    action:
        DLL_listen_only_request(FALSE); // send DLL to listen only
        DLL_xmit_fixed_request("send an I'm alive message");

//
// in the "alive" states, the DLL is on-line
// three "I'm alive" message are sent
// if lonely is detected, do nothing
//
state: alive0
    // rogue -> checkForModerator
    event:    DLL_SM_report(rogue)
    destination: checkForModerator0
    action:
        DLL_listen_only_request(TRUE); // send DLL to listen only
        DLL_enable_fixed_request(MODERATOR_TAG); // enable reception of moderators

    // first message sent, send the second
    event:    DLL_xmit_fixed_confirm();
    destination: alive1
    action:
        DLL_xmit_fixed_request("send an I'm alive message");

```

```

state: alive1
    // rogue -> checkForModerator
    event:    DLL_SM_report(rogue)
    destination: checkForModerator0
    action:
        DLL_listen_only_request(TRUE);        // send DLL to listen only
        DLL_enable_fixed_request(MODERATOR_TAG); // enable reception of moderators

    // second message sent, send the third
    event:    DLL_xmit_fixed_confirm();
    destination: alive2
    action:
        DLL_xmit_fixed_request("send an I'm alive message");

state: alive2
    // rogue -> checkForModerator
    event:    DLL_SM_report(rogue)
    destination: checkForModerator0
    action:
        DLL_listen_only_request(TRUE);        // send DLL to listen only
        DLL_enable_fixed_request(MODERATOR_TAG); // enable reception of moderators

    // third message send, consider attachment to be complete
    event:    DLL_xmit_fixed_confirm();
    destination: attached

//
// the DLL is on-line and fully functional
//
state: attached
    // rogue -> checkForModerator
    event:    DLL_SM_report(rogue)
    destination: checkForModerator0
    action:
        DLL_listen_only_request(TRUE);        // send DLL to listen only
        DLL_enable_fixed_request(MODERATOR_TAG); // enable reception of moderators

    event:    DLL_SM_report(lonely)
    destination: checkForCable
    action:
        DLL_enable_moderator_request(FALSE);    // disable moderators

//
// a rogue event was detected
// the DLL instantly disables itself (it goes to its rogue state)
//
state: checkForModerator0
    event:    DLL_enable_fixed_confirm();        // receive moderators enabled
    destination: checkForModerator1
    action:
        counter = 0;                // init moderator counter
        old_mod = NULL;
        timer.start(3 sec);

state: checkForModerator1
    // timeout
    event:    timer.expired
    destination: checkForCable0
    action:
        DLL_set_current_request(default_params); // set default parameters

    // ignore irrelevant received DLPDUS
    event:    DLL_rcv_fixed_indication(new_mod) // got a moderator Lpacket
    condition: received Lpacket is not mod
        || received Lpacket is not TUI
        || TUI.net_change == false
    destination: checkForModerator1

    // received a TUI indicating a net change in progress -- reset counter
    event:    DLL_rcv_fixed_indication(new_mod) // got a moderator Lpacket
    condition: received Lpacket is TUI
        && TUI.net_change == TRUE // the new moderator is different
    destination: checkForModerator1
    action:
        counter = 0;

    // received a moderator that would not permit this node
    // to operate normally (for example, - myaddr>umax)
    // reset counter and flash bad_net_parameter indicator
    event:    DLL_rcv_fixed_indication(new_mod) // got a moderator Lpacket
    condition: !isValid(new_mod) // the new moderator is no good for us
    destination: checkForModerator1
    action:

```

```

SM_LEDS(LED_bad_network_parameters);      // flash the indicator
old_mod = new_mod;
counter = 0;

// received moderator that doesn't match the last one -- reset counter
event:   DLL_rcv_fixed_indication(new_mod) // got a moderator Lpacket
condition: received Lpacket is mod
          && new_mod != old_mod           // the new moderator is different
destination: checkForModerator1
action:
    old_mod = new_mod;
    counter = 0;

// received moderator that matches, but not the ninth match -- just increment the
counter
event:   DLL_rcv_fixed_indication(new_mod) // got a moderator Lpacket
condition: received Lpacket is mod
          && isValid(new_mod)
          && new_mod==old_mod
          && counter<8
destination: checkForModerator1
action:
    counter++;

// have received 9 successive identical moderators
// adopt these parameters and proceed to go back on-line
event:   DLL_rcv_fixed_indication(new_mod) // got a moderator Lpacket
condition: received Lpacket is mod
          && isValid(new_mod)
          && new_mod==old_mod
          && counter==8
destination: checkForModerator2
action:
    DLL_online_request(FALSE);           // send DLL off-line

//
// second step to going back on-line
//
state: checkForModerator2

    event:   DLL_online_confirm()          // now off-line
    destination: check ForModerator3
    action:
        copy parameters from new_mod to DLL_config; // get net configuration from
moderator
        DLL_set_current_request(DLL_config);        // adopt those parameters

//
// third step to going back on-line
//
state: checkForModerator3

    event:   DLL_set_current_confirm()      // have now adopted new parameters
    destination: alive0
    action:
        DLL_listen_only_request(FALSE);    // send DLL to listen only
        DLL_xmit_fixed_request("send an I'm alive message");

```

18.2 Calculating link parameters

18.2.1 Link parameters

This subclause describes the requirements governing the selection of values for the following configuration variables:

- NUT_length;
- gb_prestart;
- gb_start;
- gb_center;
- slotTime;
- blanking.

NOTE The definitions of the link parameters include NUT_length, gb_center, gb_prestart, gb_start, blanking, and slotTime.

18.2.2 Conditions affecting link parameters

Parameter value selection shall allow for worst case conditions of these factors:

- signal propagation time between nodes;
- the timing reference accuracy at various nodes;
- moderator changes resulting in a change in the cable distance or signal propagation time between moderator and non-moderator nodes.

NOTE 1 A typical moderator change occurs as follows: At some random time the previous moderator is disconnected from the cable, loses power, or stops transmitting moderators for some other reason. Two NUTs (sometimes three) pass during which no moderator DLPDUs are transmitted. The new moderator transmits its first moderator DLPDU in the guardband of the third NUT since it last received a valid moderator DLPDU.

NOTE 2 In the event that the previous moderator was lost near the time that the moderator DLPDU was being transmitted it may be possible that the new moderator received the last moderator DLPDU from the previous moderator, while other nodes may not have received the last moderator DLPDU. It is therefore required that nodes tolerate an interval of 4 NUTs between reception of valid moderator DLPDUs.

18.2.3 Moderator change

In the requirements which follow, the term “moderator change” shall be interpreted by a non-moderator node as:

- reception of a valid moderator DLPDU from the original moderator node
- a time interval of up to 4 NUTs during which no valid moderator DLPDUs are received
- reception of a valid moderator DLPDU from a second moderator node

18.2.4 NUT timing

18.2.4.1 Tone

A network update time (NUT) for a node shall begin at a point in time called the tone, and shall end at the next tone.

NOTE The times specified in this subclause assume a perfectly accurate clock. Any permitted inaccuracy is explicitly included in the requirements. Sources of inaccuracy may include firmware delay, hardware delay, and local crystal inaccuracy. Since some of the time range requirements do not explicitly specify the clock accuracy tolerance, implementations should provide that internal timers are set such that the requirements are met.

18.2.4.2 Clock accuracy

The timing specifications for PhL Signalling to support the tone and NUT precision shall be as defined in Table 194.

Table 194 – PhL timing characteristics

Specification	Limits / characteristics	Comments
Data Rate	5 Mbit/s $\pm$ CA	also called M_Symbol rate, <i>data “zero” or “one”</i>
Bit Time	200 ns $\pm$ CA	also called M_Symbol time, <i>data “zero” or “one”</i>
PhL symbol time	100 ns $\pm$ CA	also called Phy_Symbol time, see data encoding rules
Clock Accuracy (CA)	$\pm$ 150 parts/million maximum	including temperature, long term, and short term stability

18.2.4.3 Restriction on NUT_length

The maximum value of NUT_length shall be 10000 (100 ms). The minimum value NUT_length shall ensure that each NUT allows a maximum length unscheduled Lpacket to be sent after the scheduled Lpackets have been sent.

NOTE For example, with no scheduled connections, the minimum value of NUT_length is about 1 ms.

18.2.4.4 Moderator node timing

The moderator node shall begin a new NUT at $(\text{NUT\_length} \times 10 \mu\text{s}) \pm \text{CA}$ after the previous tone. The reception of a valid or invalid DLPDU shall not affect the timing of the transmission of the moderator DLPDU by the moderator node.

The moderator node shall begin transmission of the moderator DLPDU at $(\text{NUT\_length} - \text{gb\_center}) \times 10 \mu\text{s} \pm \text{CA}$ after the tone.

18.2.4.5 Non-moderator node timing

A non-moderator node shall begin a new NUT at $(\text{gb\_center} \times 10 - 40 \pm 2) \mu\text{s}$ after the last M_Symbol of the end delimiter of any valid received moderator DLPDU. A non-moderator node shall begin a new NUT at $(\text{NUT\_length} \times 10) \mu\text{s} \pm \text{CA}$ after the previous tone if a valid moderator DLPDU is not received in the current NUT.

Every non-moderator node shall receive and process any valid moderator DLPDU which begins between $(\text{tone} + 20) \mu\text{s}$ and $(\text{tone} + (\text{NUT\_length} \times 10 - 30) \mu\text{s} \pm \text{CA})$.

18.2.4.6 Post-tone silence

No DLPDUs shall be transmitted between the tone and $20 \mu\text{s}$ after the tone.

18.2.4.7 Pre-tone silence

No DLPDU shall be transmitted, except the moderator DLPDU, that does not terminate before $((\text{NUT\_length} - \text{gb\_prestart}) \times 10 + 11,2) \mu\text{s} \pm \text{CA}$ after the tone.

The value of `gb_prestart` shall provide that the blanking time after the last non-moderator DLPDU transmitted by any node and received by any other node expires before the start of the guardband in the receiving node. This condition shall also be met during a moderator change.

18.2.4.8 Guardband start

The guardband shall start at $(\text{NUT\_length} - \text{gb\_start}) \times 10 \mu\text{s} \pm \text{CA}$ after the tone.

The value of `gb_start` shall provide that any node shall observe the start of the moderator DLPDU no earlier than $((\text{NUT\_length} - \text{gb\_start}) \times 10 + 20) \mu\text{s} \pm \text{CA}$ after the previous tone in that node. This condition shall be met during a moderator change.

18.2.4.9 Guardband center

The value of `gb_center` shall provide that any node shall observe the end of the moderator DLPDU no later than $(\text{NUT\_length} \times 10 - 40) \mu\text{s} \pm \text{CA}$ after the tone. This condition shall also be met during a moderator change.

18.2.4.10 Useable time

Every node shall receive and process any valid DLPDU that begins between $(\text{tone} + 20 \mu\text{s})$ and $(\text{tone} + (\text{NUT\_length} - \text{gb\_start}) \times 10 \mu\text{s} \pm \text{CA})$, except that reception of a valid or invalid DLPDU shall not affect the timing of the transmission of the moderator DLPDU by the moderator node.

18.2.5 Slot timing

18.2.5.1 Slot time

In the following description, the slots shall be numbered sequentially. The value of `slotTime` shall provide that a DLPDU transmitted in slot N is received in slot N at all other nodes assuming worst case conditions of

- crystal drift;
- signal propagation delay between nodes;
- MAC ID assignment;

but excluding noise events.

18.2.5.2 First slot

Scheduled slot 0 shall begin at 20 µs after the tone.

18.2.5.3 Missing node

If no DLPDU is sent or received during slot N-1, slot N shall begin at $\text{slotTime} \times 1 \mu\text{s} \pm \text{CA}$ after the beginning of slot N-1.

18.2.5.4 Restarting the slot timer

If a DLPDU is sent or received during slot N-1, slot N shall begin no later than 6,0 µs after the last M_Symbol of the end delimiter of the DLPDU. If a damaged DLPDU, with `ph_status_indication` not equal to Normal, is received during slot N-1, slot N shall begin no later than 6,0 µs after `ph_lock_indication` goes FALSE.

18.2.5.5 Node turn around time

If both node N-1 and node N transmit in their respective slots, node N shall begin transmitting between 11,2 µs and 12,8 µs after the last M_Symbol of the end delimiter of the DLPDU transmitted by node N-1.

18.2.5.6 Beginning transmission after missed node

If node N-1 does not transmit in its slot, and node N is enabled to transmit in its slot, node N shall begin transmitting no earlier than the start of slot N, and no later than 6,8 µs after the start of slot N.

NOTE The major factor to consider here is crystal drift. In the worst case, where there are two nodes attached to the network at MAC IDs 0 and 99, SMAX=98, UMAX=99, and USR=1, there can be a silence of 196 slot times between the scheduled DLPDU transmitted by node 0 and the unscheduled DLPDU transmitted by node 99. If node 99 has a fast crystal and node 0 has a slow crystal, or vice versa, the value of `slotTime` should be adequate to guarantee that, when node 99 transmits, node 0 agrees that the current slot is 99.

18.2.6 Blanking

The value of blanking shall be 6.

18.2.7 Example implementation (informative)

The requirements in 18.2.4, 18.2.5 and 18.2.6 are met (not optimally) in the following example program which calculates the link parameters:

```
#include <stdio.h>
#include <stdlib.h>
#include <stdint.h>

#define roundup(x,y) (((x) - 1L) / (y) + 1L) /* Round up integer division */

#define F_TURN      16060L /* ns for firmware turnaround and detect */
#define B_TURN      5720L /* ns extra turnaround with large blanking value */
#define NULLFRAME   7L /* octets for length of a null Lpacket transmission */
#define DELAY       4110L /* ns per 1000 meters cable delay */
#define RPT_DLY     815L /* ns per repeater delay */
#define PPMT        1500L /* Max crystal error in tenths of PPM = +/- 150,0 ppm */

#define MODBEFORE    2 /* min moderator overhead before Lpacket (housekeeping) */
#define MODRECOV     3 /* worst case min moderator overhead after any Lpacket */
#define MODTIME      5 /* ticks for the moderator plus error recovery if bad */
#define MINUNSCHED  850 /* minimum unscheduled time */
```

```

int main(int argc, char **argv)
{
    /* Default values for command line input parameters. These are
     * arbitrarily chosen for our typical convenient network sizes.
     */
    int dest = 0xff;
    int board = -1; /* If -1 use driver, else use XT version TX */
    int redund = 3; /* Unspecified. Later code defaults to 3 */
    long NUT_time = 10000L;
    int length = 1000;
    int smax = 7;
    int umax = -1;
    int repeater = 0;
    int blank = 6;
    int maxfrm = 255;
    int intmod = 256;
    int terse = 0;
    int listenonly = 0;
    int noleds = 0;

    /* Other derived variables. */

    int gap;
    long drift;
    long prop;
    long turn;
    long bslot;
    long slot;
    long picodev;
    int center;
    int start;
    int prestart;
    long unscheduled;
    long nuttenth;
    int nuthigh;
    int nutlow;
    char txhead[7];
    char *a;

    /* Parse arglist */

    argc--;
    argv++;
    if(argc==0)
    {
        argc=1;
        argv[0]="-h";
    }

    while(argc && (a=&argv[0][0],*a++=='-'))
    {
        while(*a)switch(*a++)
        {
            case 'x' :
                board = atoi(a);
                if ((board < 0) || (board > 3))
                {
                    printf("%%ERROR -- 0 <= board number <= 3.\n");
                    return(1);
                }
                goto nextarg;

            case 'l':
                listenonly = 128;
                break;
            case 'n':
                noleds = 64;
                break;
            case 'b':
                redund = 1;
                break;
            case 'a':
                redund = 2;
                break;
            case 'r':
                redund = 3;
                break;
            case 't':
                terse = 1;
                break;
        }
    }
}

```

```

        case 'h' :
        default:
            printf("Compute and print a net config Lpacket.\n\n");
            printf("NETCONF [-abdhlnrt] [-x<board>] [0x<dest>] "
                "[<NUT time> [<length> [<smax>\n"]);
            printf("          [<umax> [<repeaters> [<blanking> [<maxframe> "
                "[<int modulus>]]]]]]\n");
            return(1);
        }
nextarg:  argc--;
        argv++;
    }

    /* Check for network address option */
    if (argc && argv[0][0] == '0' && argv[0][1] == 'x')
    {
        sscanf(argv[0], "%i", &dest);
        if ((dest < 0) || (dest > 255))
        {
            printf("%%ERROR -- 0x00 <= destination address <= 0xff.\n");
            return(1);
        }
        argc--;
        argv++;
    }

    /* Parse remaining arguments.
    */
    switch (argc)
    {
        case 8:  intmod = atoi(argv[7]);
        case 7:  maxfrm = atoi(argv[6]);
        case 6:  blank = atoi(argv[5]);
        case 5:  repeater = atoi(argv[4]);
        case 4:  umax = atoi(argv[3]);
        case 3:  smax = atoi(argv[2]);
        case 2:  length = atoi(argv[1]);
        case 1:  NUT_time = atol(argv[0]);
    }

    /* check the validity of all the arguments */
    if ((intmod < 1) || (intmod > 256))
    {
        printf("%%ERROR -- 1 <= interval count modulus <= 256.\n");
        return(1);
    }
    if ((maxfrm < 0) || (maxfrm > 255))
    {
        printf("%%ERROR -- 0 <= max scheduled frame <= 255.\n");
        return(1);
    }
    if ((blank < 0) || (blank > 255))
    {
        printf("%%ERROR -- 0 <= blanking <= 255.\n");
        return(1);
    }
    if (repeater < 0)
    {
        printf("%%ERROR -- number repeaters >= 0.\n");
        return(1);
    }
    if ((smax < 0) || (smax > 254))
    {
        printf("%%ERROR -- 0 <= max scheduled addr <= 254.\n");
        return(1);
    }
    if (length < 0)
    {
        printf("%%ERROR -- cable length >= 0.\n");
        return(1);
    }
    if ((NUT_time < 350L) || (NUT_time > 100000L))
    {
        printf("%%ERROR -- 350 <= nut time <= 100,000.\n");
        return(1);
    }

    if (umax == -1) umax = smax;
    if ((umax < smax) || (umax > 254))
    {
        printf("%%ERROR -- smax <= umax <= 254.\n");
        return(1);
    }
}

```

```

if (NUT_time%10!=0)
{
    printf("%%ERROR -- NUT length is specified in 10 µs intervals.\n");
    return(1);
}

/* Gap is the number of slot times that can pass in the normal
protocol between transmissions.

When  $s_{max} < u_{max}$ , then the maximum silence equals  $s_{max} + u_{max} - 1$ .
The maximum silence occurs on the network with  $s_{max} < u_{max}$  when
the first node is at 0, the only other is at  $u_{max}$  with the  $USR==1$ .

To illustrate a transmit sequence example when  $s_{max} < u_{max}$ ,
set  $s_{max}=4$  and  $u_{max}=5$ ,  $USR==1$ , with two nodes at 0 and 5.
Slot times -> ssssuuuu -> ( $s_{max} + u_{max} - 1$ )
01234123450

When  $s_{max}=u_{max}$ , then the maximum silence equals  $s_{max} + u_{max} - 2$ .

The maximum silence occurs on the network with  $s_{max} = u_{max}$  when
the first node is at 0, the only other is at 1 with the  $USR==2$ .

To illustrate a transmit sequence example when  $s_{max}=u_{max}$ ,
set  $s_{max}=u_{max}=5$ ,  $USR==2$ , with two nodes at 0 and 1.
Slot times -> ssssuuuu -> ( $s_{max} + u_{max} - 2$ )
012345234501

Units: slot times
*/

gap = ( $s_{max} < u_{max}$ ) :  $s_{max} + u_{max} - 1$  :  $s_{max} + u_{max} - 2$ ;
if (gap < 0) gap = 0;

/* the drift is a correction factor needed to compensate for
the amount of time that the fastest and slowest nodes drift
apart during the longest silence due to crystal inaccuracy.

Actual drift factor =  $(1 - (2 * gap + 1) (ppm)) / (1 - ppm^2)$ 
however,  $ppm^2$  is so small it may be ignored.

Units: unitless factor scaled by micro/pico
*/

drift = roundup( $100000000L - (long)(2 * gap + 1) * PPMT$ , 10L);

/* Prop is the time the signal takes to travel down the cable and
work its way through all the repeaters and modems. It assumed
on even a cable system that has no repeater boxes, that two
two repeater exist to allow nodes connected via the NAP.

Units: picoseconds
*/

prop = (long)(length)*DELAY + (long)(repeater+2)*RPT_DLY*1000L;

/* turn is:

- the amount of time it takes for the DLL to respond to a
transmission, that is, - the time from the end of the end delimiter
to the start of the preamble

plus

- the time it takes another DLL to detect that the first *has*
responded, that is, - the time from the start of the preamble to the
end of the start delimiter.

Units: nanoseconds
*/

/* turnaround could be */
turn = (long)(blank)*1600L + B_TURN; /* <-- blanking constrained */
if (turn < F_TURN) turn = F_TURN; /* <-- or firmware constrained */

/* bslot is the ideal slot time - which is usually defined as
2 props + turnaround + detect (note that in this case, the
detect time and the turn time are combined)
Units: picoseconds
*/

bslot = 2*prop + (turn*1000L);

/* slot is the ideal slot time adjusted to account for variances in
crystal frequency.
Units: microseconds
*/

slot = roundup(bslot, drift);

```

```

if ((slot < 1L) || (slot > 256L))
{
    printf("%sERROR -- Slot time is large, reduce network complexity. \n");
    return(1);
}

/* Calculate guardband parameters (round up to a multiple of a 10
 * microsecond tick clock).
 */

/* calc nut time - units: 10 usec ticks */
nuttenth = roundup(NUT_time,10L);

/* picodev is the amount the clocks of the slowest and fastest
 * node drifts apart during four NUTs
 * Units: picoseconds
 */
picodev = nuttenth * PPMT * 8L;

/* center is the time before the next tone that the moderator
 * begins transmitting the moderator DLPDU. This includes adjustments
 * for clock drift during up to 4 NUTs, plus a moderator switchover
 * from a near to a far node.
 * Unit: 10 usec ticks
 */
center = MODTIME + MODRECOV +
(int)roundup(picodev + 2*prop , 10000000L);

/* start is the time before the next tone that the guardband starts.
 * Unit: 10 usec ticks
 */
start = center + MODBEFORE +
(int)roundup(picodev , 10000000L);

/* prestart is the time before the next tone beyond which nodes may not
 * transmit. Due to clock skew and other factors a transmission which
 * is sent such that it ends just at prestart may actually be received
 * by another node such that the transmission ends just as the guardband
 * starts. The prestart margin is required in order to ensure that
 * transmissions from slow nodes do not cross over into the guardbands
 * of faster nodes.
 * Unit: 10 usec ticks
 */
prestart = (int)roundup((nuttenth*PPMT*2L) + ((NULLFRAME+blank)*1600000L)
+ bslot , 10000000L) + start;

/* Determine if nut parameters is practical and print warnings.
 * Initial calculations are conservative due to roundup of several
 * parameters. Usage of fixround helps correct it.
 */

/* calc the max length of unscheduled time assuming that all
 * scheduled nodes are silent.
 * Units: microseconds
 */
unscheduled = NUT_time - max(roundup(NULLFRAME*1600000L+prop+turn*1000L,
1000000000L),slot)*(long)(smax+1)
- (long)(prestart)*10L;

if (unscheduled < MINUNSCHED)
    printf("%sWARNING -- Not enough scheduled time available. Increase nut
time.\n");

/* Print statistics and the transmit command.
 */
nutlow = nuttenth & 0xff;
nuthigh = (nuttenth >> 8) & 0xff;
if(!terse)
{
    long deviation = roundup(picodev , 10000000L);

    printf("slot = %ld usec, max unscheduled time = %ld usec\n",
        slot,unscheduled);
    printf("dev = %ld usec, prestart = %d ticks, start = %d ticks, center = %d
ticks\n",
        deviation,prestart,start,center);
}

if (board == -1) sprintf(txhead,"tx");
else sprintf(txhead,"txt /%d",board);
printf(
    "%s m a 17 %02x %02x %02x %02x %02x %02x %02x "
    "%02x %02x %02x %02x %02x %02x 00 00 00\n",
    txhead,dest,nutlow,nuthigh,smax,umax,(int)slot-1,blank,
    start,center,(listenonly|noleds|redund),maxfrm,intmod-1,prestart);
return(0);
}

```

19 Type 2: Detailed specification of DL components

19.1 General

This clause specifies each main functional component of the DL internal architecture as shown in Figure 25 using the modeling language of 16.1.

19.2 Access Control Machine (ACM)

The Access Control Machine (ACM) shall control the sequencing of transmissions by this node.

```
// ACM State Machine Description
//
// constant and type definitions
//
// generic type and constant definitions
//
typedef enum {FALSE=0, TRUE=1} BOOL;

//
// protocol constants
//
#define LOWCOUNTINIT      2 // NUTs as lowman before switchover to moderator
#define MODERATOR_LENGTH 40 // length of the moderator DLPDU in usec
#define TXDUPCHECKS        3 // moderators that have to be heard before transmit at
// powerup
#define TXNOMOD            5 // number of missed moderators before disabling transmit
#define DEAFNESS           5 // number of missed moderators before adjusting phase
#define DEAFADJUST         5 // 10 usec ticks to slip NUT if deafness is detected
#define LONELYINIT         8 // NUTs that are tolerated without hearing anything
// before becoming lonely (should be longer than DEAFNESS)
// to ensure time for deaf recovery)
//
// Lpacket definition
//
//
// Lpacket constants: masks for control octet
//
#define FIXEDSCREEN        1
#define TAGPAD             2
#define DATAPAD            4

//
// moderator Lpacket constants
//
#define MODERATOR_SIZE     9 // moderator Lpacket size octet (in 16 bit words)
#define MODERATOR_CTL      1 // moderator Lpacket ctl octet (fixed tag)
#define MODERATOR_TAG      0 // moderator Lpacket service octet

//
// Lpacket class
//
class Lpacket
{
public:
    USINT size; // the size octet
    USINT ctl;  // the ctl octet

    // return the value of the tag pad bit
    int tag_pad(void)
    {
        return (ctl&TAGPAD) >>1;
    }

    // return the value of the data pad bit
    int data_pad(void)
    {
        return (ctl&DATAPAD) >>2;
    }

    // return the size of the Lpacket (in octets)
    int wire_size(void)
    {
        return size*2 - tag_pad() - data_pad();
    }

    // get the next octet to be transmitted
    USINT get_next_octet(void);
};
```

```

    // is this Lpacket aborted?
    BOOL abort(void);
};

//
// fixed tag Lpacket (two octet tag)
//
class fixedLpacket : public Lpacket
{
public:

    USINT service;
    USINT dest;
};

//
// moderator fixed tag Lpacket
//
class moderatorLpacket : public fixedLpacket
{
public:

    UINT NUT_length;
    USINT smax;
    USINT umax;
    USINT slotTime;
    USINT blanking;
    USINT gb_start;
    USINT gb_center;
    USINT usr;
    USINT interval_count;
    USINT modulus;
    USINT tMinus;
    USINT gb_prestart;
    USINT spare;
};

/////////////////////////////////////////////////////////////////
//
// internal VARIABLES
//
// unless otherwise specified, all variables are initialized to zero at powerup

int tmp;
USINT interval_count; // counts how many NUTs have passed (0 - modulus)
USINT itr;           // implicit token register - tracks which MAC should xmit
USINT usr;           // unscheduled start register -- first unscheduled token
USINT tMinus;        // countdown to synchronized change
USINT lowcount;      // how many times have I been lowman?
USINT modcount;      // counts moderators (or missed moderators) depends on netState
USINT modcountinit;  // how many NUTs to check for dupnode
USINT nglowman;      // optimistic lowman detector
USINT qlowman;       // pessimistic lowman detector
USINT lonely;        // how many NUTs have I been lonely
USINT deafcount;     // number of times moderator apparently occurred during deaf
period
BOOL in_guardband;    // TRUE during the guardband
BOOL dupflag;        // edge detector for dupnode
BOOL scheduled;      // scheduled/unscheduled part of the NUT
UINT octetsleft;     // octets / octets left before end of current transmitted DLPDU
USINT macSrce;       // source MAC ID for the current DLPDU
USINT currentMod;     // the current moderator
moderatorLpacket moderator; // buffer for moderator Lpackets

enum { OFFLINE,
      WATCH,
      DUPCHECK,
      NOTMOD,
      MOD,
      ROGUE,
      DUPNODE
} netState; // connection state

/////////////////////////////////////////////////////////////////
// timer definition
//
// various behaviors for timers
typedef enum {ONCE_UP, ONCE_DOWN, REPEAT_UP, REPEAT_DOWN} timer_mode;

class timer
{
public:
    float count; // number of ticks left
    float preset; // init ticks, or cycle length

```

```

    BOOL expire;          // latched flag when timer expires
    timer_mode mode;

    // constructor: defines mode
    timer(timer_mode m)
    {
        mode = m;
    }

    void restart(void)
    {
        // if counting down, start with preset
        if(mode == REPEAT_DOWN || mode == ONCE_DOWN)
        {
            count = preset;
        }
        else
        {
            count = 0;    // else start from zero
        }
        // start counting (implementation not specified)
    }

    void start(float interval) // start counting down from interval towards zero
    {
        mode = ONCE_DOWN;
        preset = interval; // set interval
        restart();         // start counting
    }

    void begin_counting(void) // start counting up from zero
    {
        mode = ONCE_UP;
        restart();
    }

    bool expired(void)      // returns true if the timer has expired; clears flag
    {
        BOOL tmp;

        tmp = expire;
        expire = 0;
        return tmp;
    }

};

//
// time conversion factors
//
#define usec
#define msec *1000
#define sec *1000000

//
// define the timers used by the ACM
//
class timer gen_timer(ONCE_DOWN); // general purpose, in several places
class timer slot_timer(REPEAT_DOWN); // used for response time-out between messages
class timer watch_timer(ONCE_DOWN); // used for watch time state
class timer NUT_timer(REPEAT_DOWN); // this generates the NUT timing
class timer holdoff_timer(ONCE_DOWN); // holds off transmit for a timer after
transmit or receive

////////////////////////////////////
//
// interface to PhL
//
BOOL ph_lock_indication; // optimistic DLPDU detect (clock recovery is tracking)
BOOL ph_frame_indication; // pessimistic DLPDU detect (Manchester valid since the SD)

////////////////////////////////////
//
// interface to Deserializer
//
BOOL RX_dataReady; // an octet of a DLPDU is available
USINT RX_rxData; // octet / octet most recently received from DLPDU (source MAC ID)
BOOL RX_endMAC; // the end delimiter has been detected
BOOL RX_FCSOK; // FCS on last DLPDU was OK
BOOL RX_abort; // abort received on last DLPDU

```



```

/////////////////////////////////////////////////////////////////
//
// interface to RxM
//
BOOL RX_receivedLpacket; // an Lpacket has been received
moderatorLpacket RX_Lpacket; // the Lpacket most recently received

/////////////////////////////////////////////////////////////////
//
// interface to transmitter (TxM)
//
void TX_sendHeader(USINT myaddr); // send preamble, SD, source MAC ID
void TX_sendLpacket(Lpacket &lp); // send an Lpacket
void TX_sendTrailer(void); // send FCS, ED
BOOL TX_headerComplete; // header has been sent
BOOL TX_LpacketComplete; // Lpacket has been sent
BOOL TX_trailerComplete; // trailer has been sent
BOOL TX_abort; // DLPDU was aborted

/////////////////////////////////////////////////////////////////
//
// interface to LLC
//
Lpacket LLC_Lpacket; // shared variable between LLC and ACM containing the
Lpacket
BOOL LLC_pickLpacket (BOOL scheduled,
                      UINT octetsLeft); // tell LLC to pick an Lpacket for transmission
LLC_LpacketSent(void); // tell LLC that the Lpacket was sent
/////////////////////////////////////////////////////////////////
//
// interface to station management
//
//
// various events that can be reported to SM
//
typedef enum
{
    DLL_EV_rxGoodFrame,
    DLL_EV_txGoodFrame,
    DLL_EV_badFrame,
    DLL_EV_errA,
    DLL_EV_errB,
    DLL_EV_txAbort,
    DLL_EV_NUT_overrun,
    DLL_EV_dribble,
    DLL_EV_nonconcurrency,
    DLL_EV_rxAbort,
    DLL_EV_lonely,
    DLL_EV_dupNode,
    DLL_EV_noisePulse,
    DLL_EV_collision,
    DLL_EV_invalidModAddress,
    DLL_EV_rogue,
    DLL_EV_deafness,
    DLL_EV_supernode,
}event;

extern void DLL_event(event ev, int opt=0); // report an event
extern void DLL_currentMod_indication(USINT macID);
extern void DLL_tminus_zero_indication(void);

extern void DLL_online_confirm(BOOL en);
extern void DLL_listen_only_confirm(BOOL en);
extern void DLL_tone_indication(USINT NUT_count);

class DLL_config_data
{
public:
    USINT myaddr;

    UINT NUT_length;
    USINT smax;
    USINT umax;
    USINT slotTime;
    USINT blanking;
    USINT gb_start;
    USINT gb_center;
    USINT interval_count;
    USINT modulus;
    USINT gb_prestart;
};

```

```

// interface to DLL management interface

extern DLL_config_data DLL_SM_pending;
extern DLL_config_data DLL_SM_current;

extern BOOL SM_powerup;

extern BOOL SM_mod_enable; // state of mod_enable
extern BOOL SM_online; // state of on-line/off-line
extern BOOL SM_online_req; // pending state of on-line/off-line
extern BOOL SM_listen_only; // listen only state
extern BOOL SM_listen_only_req; // pending listen only state

////////////////////////////////////
//
// subroutines
//
// take care of common processing for network parameter changes
//
void handle_net_change(void)
{
    if(SM_listen_only != SM_listen_only_req) // handle completion of listen_only
change
    {
        SM_listen_only = SM_listen_only_req;
        DLL_listen_only_confirm(SM_listen_only);
    }

    if(tMinus==1) // handle completion of synchronized change
    {
        tMinus = 0;
        DLL_tminus_zero_indication();
    }
}

//
// maintenance work done whenever the moderator is received
//
void processModerator(moderatorLpacket &moderator)
{
    //
    // copy data from the moderator
    // which cannot cause a rogue condition
    //
    interval_count = moderator.interval_count;
    usr = moderator.usr;
    tMinus = moderator.tMinus;

    lowcount = LOWCOUNTINIT; // re-initialize lowman counter

    // resynchronize the NUT timer to the moderator
    NUT_timer.count = DLL_SM_current.gb_center - MODERATOR_LENGTH/10;

    if(netState == MOD) // if this node is moderator
    {
        netState = NOTMOD; // and a moderator was received!
        modcount = 0; // drop moderatorship
        // init count of missing moderators
        DLL_event(DLL_EV_nonconcurrency); // indicate non-concurrency
    }
    else
    {
        if(!in_guardband) // unexpected moderator
        {
            DLL_event(DLL_EV_nonconcurrency); // indicate non-concurrency
        }

        if(netState == WATCH) // got moderator, there is a network
        {
            netState = DUPCHECK; // go check for dupnodes
            modcount = modcountinit; // this is how many NUTs to check
        }
        else if(netState == DUPCHECK) // if checking for duplicate MAC IDs
        {
            if(SM_listen_only == FALSE) // unless this node is required to
            {
                // remain in this state
                modcount--; // decrement NUT counter
                if(modcount == 0) // when done
                {
                    netState = NOTMOD; // it's OK to go transmit
                }
            }
        }
        else if(netState == ROGUE) // a rogue that sees a good moderator
    }
}

```

```

        {
            netState = DUPCHECK;          // can exit rogue state
            modcount = modcountinit;
        }
    else
    {
        modcount = 0;          // else just reset mod counter
    }
}

//
// recovery after a time-out while waiting for moderator
//
void missed_moderator(void)
{
    if(netState == NOTMOD)          // if supposed to be on-line and hearing moderators
    {
        modcount++;                // count consecutive missing moderator
        if(modcount >= TXNOMOD)    // if exceeded limit
        {
            netState = DUPCHECK;    // change to a quietly waiting mode
            modcount = 1;          // init mods required before return on-line
        }

        if(SM_listen_only == FALSE // if allowed to transmit
            && SM_mod_enable == TRUE // and allowed to be moderator
            && DLL_SM_current.myaddr <= nqlowman) // and apparently the lowman
        {
            lowcount--;            // count consecutive NUTs as lowman
            if(lowcount == 0)      // if this is the second time
            {
                netState = MOD;    // then activate the moderator on third NUT
            }
        }
    }
    else
    {
        lowcount = LOWCOUNTINIT; // otherwise, re-initialize lowman counter
    }
}

//
// housekeeping actions at the end of each NUT
//
void housekeeping(void)
{
    if(netState == WATCH)          // in watch state
    {
        lonely = LONELYINIT;        // hold the lonely counter reset
        if(watch_timer.count == 0)  // if timed out (no moderators)
        {
            // change to on-line -- try to build a link
            if (SM_listen_only == FALSE // if allowed to transmit
                && SM_mod_enable == TRUE // and allowed to moderate
                && nqlowman == DLL_SM_current.myaddr) // and lowman
            {
                netState = MOD; // activate moderator state
            }
        }
        else
        {
            netState = DUPCHECK; // else wait for another node to moderate
            modcount = modcountinit;
        }
    }

    else if (qlowman == 255          // no good FCS received in the last NUT
            && DLL_SM_current.myaddr != 0 // local node is not supernode
            && netState != WATCH)    // not already in the watch state
    {
        lonely--;
        if(lonely == 0 || netState == ROGUE || netState == DUPNODE)
        {
            DLL_event(DLL_EV_lonely);
            if(netState == ROGUE)
            {
                watch_timer.start(0,75 sec);
            }
            else
            {
                watch_timer.start(1,25 sec);
            }
            netState = WATCH;
        }
    }
}

```

```

    }
else
{
    lonely = LONELYINIT;
}

if(deafcount >= DEAFNESS)
{
    deafcount = 0;
    DLL_event(DLL_EV_deafness);
    NUT_timer.count += DEAFADJUST;
}

// this counter determines how many NUTS a node should
// check the link before deciding that it is not
// a dupnode.  if 0<myaddr<=SMAX a node is guaranteed to
// an access each NUT, so the check time is low.  On the
// other hand, if smax<myaddr<=umax, a node might have to wait
// a long time for its access slot come up, so use a long
// time.  If a DUPCHECKING unscheduled only node ever sees its slot
// go by, it sets modcount to a lower number immediately.

// nodes between SMAX and UMAX aren't guaranteed to get a slot every NUT
// if not an unscheduled only node, use a low number else use a huge number

if (DLL_SM_current.myaddr > DLL_SM_current.smax)
    modcountinit = TXDUPCHECKS;
else
    modcountinit = 255;

// update the interval count once per NUT
interval_count = (interval_count + 1) % DLL_SM_current.modulus;

// update the usr once per NUT
usr = (usr + 1) % (DLL_SM_current.umax + 1);
}

////////////////////////////////////
////////////////////////////////////
//
// The ACM State Machine
//
state: powerup

    event:      SM_powerup
    destination: offline

//
// be idle until told to go on-line
//
state: offline

    // normal case
    event:      SM_online == TRUE
    condition:   ph_frame_indication == FALSE
    destination: rxMod0
    action:
        in_guardband = TRUE;          // start up at the beginning of a guardband
        if(DLL_SM_current.myaddr == 0) // either local node is supernode
        {
            netState = MOD;
            SM_listen_only = FALSE;
        }
        else
            // or not a supernode
        {
            netState = WATCH;
            watch_timer.start(1,25 sec);
        }
        scheduled = FALSE;
        in_guardband = TRUE;
        DLL_online_confirm(SM_online);

// exception: there was an Lpacket on the wire at transition to on-line
event:      SM_online == TRUE
condition:   ph_frame_indication == TRUE
destination: badMod          // treat it as an error inside the guardband
action:
    DLL_event(DLL_EV_badFrame,macSrce);
    in_guardband = TRUE;
    if(DLL_SM_current.myaddr == 0)
    {
        netState = MOD;
        SM_listen_only = FALSE;
    }
    else
    {
        netState = WATCH;
        watch_timer.start(1,25 sec);
    }

```

```

    }
    DLL_online_confirm(SM_online);

//
// wait for either energy on the wire or a slot time-out
//
state: waitFrame

    event:    ph_lock_indication == TRUE
    destination: frEst1
    action:    gen_timer.begin_counting();

    event:    slot_timer.expired()
    destination: gap
    action:    gen_timer.start(0,6 usec);

//
// wait for frame start, or noise
//
state: frEst1
    event:    ph_frame_indication == TRUE
    destination: rxFrame

    event:    ph_lock_indication == FALSE      // short energy burst == noise blip
    destination: waitFrame
    action:    DLL_event(DLL_EV_noisePulse);    // ignore it

    event:    gen_timer.count >= 8,0 usec
    destination: frEst2

//
// continue waiting
//
state: frEst2

    event:    ph_frame_indication == TRUE
    destination: rxFrame

    event:    ph_lock_indication == FALSE      // longer burst treat as a damaged DLPDU
    destination: endFrame
    action:    DLL_event(DLL_EV_badFrame,macSrce);
               gen_timer.start(5,2 usec);

    event:    gen_timer.count >= 11,2 usec      // energy this long with no data start
    destination: endFrame                      // treat as bad DLPDU, no start delimiter
    action:    DLL_event(DLL_EV_badFrame,macSrce);
               gen_timer.start(5,2 usec);

//
// wait here for there to be no detectable energy on the wire (no clock lock)
//
state: endFrame

    event:    ph_lock_indication == FALSE
    destination: gap
    action:    slot_timer.restart();

//
// simulate first half of node turnaround time with this timing gap
//
state: gap

    event:    ph_frame_indication == TRUE      // DLPDU start overrides all
    destination: rxFrame

    event:    gen_timer.expired()              // process the rest of
    destination: nextSlot
    action:    gen_timer.start(6,1 usec);      // start second half of turnaround

    if(scheduled)                             // do implicit token rotation, scheduled
    {
        itr++;
        if(itr>DLL_SM_current.smax)           // after SMAX,
        {                                     // change from scheduled to unscheduled
            itr=usr;
            scheduled = FALSE;
        }
    }
    else                                       // implicit token rotation, unscheduled
    {

```

```

        itr = (itr + 1) % (DLL_SM_current.umax + 1);
    }

//
// decide what comes next
//
state: nextSlot

    event:    ph_frame_indication == TRUE        // detect a DLPDU start at all times
    destination: rxFrame

    // time for the guardband
    event:    gen_timer.expired()
    condition: NUT_timer.count < DLL_SM_current.gb_prestart
    destination: guardBand
    action:

        // scheduled time not completed -- log the error and proceed
        if(scheduled == TRUE)
        {
            DLL_event(DLL_EV_NUT_overrun);
            scheduled = FALSE;
        }
        in_guardband = TRUE;

    // not this node's slot, keep counting.
    // Note that the slot timer auto repeats to maintain regular slot intervals
    event:    gen_timer.expired()
    condition: NUT_timer.count >= DLL_SM_current.gb_prestart
                && itr != DLL_SM_current.myaddr
    destination: waitFrame

    // this node's slot, but not allowed to talk
    event:    gen_timer.expired()
    condition: NUT_timer.count >= DLL_SM_current.gb_prestart
                && itr == DLL_SM_current.myaddr
                && !(netState == MOD || netState == NOTMOD)
    destination: waitFrame
    action:    modcount = min(modcount, TXDUPCHECKS); // optimize dupcheck

    // this node's slot, and allowed to talk
    event:    gen_timer.expired()
    condition: NUT_timer.count >= DLL_SM_current.gb_prestart
                && itr == DLL_SM_current.myaddr
                && (netState==MOD || netState==NOTMOD)
    destination: waitForHold

//
// a DLPDU has started, this deals with the data
//
state: rxFrame

    // prematurely lost the DLPDU event
    event:    ph_frame_indication == FALSE
    destination: endFrame
    action:
        DLL_event(DLL_EV_badFrame,macSrce);
        gen_timer.start(5,2 usec);

        // start hold timer after line is quiet
        holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

    // duplicate node condition
    event:    RX_dataReady
    condition: RX_rxData == DLL_SM_current.myaddr
    destination: dupNode
    action:
        macSrce = RX_rxData;
        nqlowman = min(nqlowman,macSrce);

    // record the source MAC ID
    event:    RX_dataReady
    condition: RX_rxData != DLL_SM_current.myaddr
    destination: nextLpacket
    action:
        macSrce = RX_rxData;
        nqlowman = min(nqlowman,macSrce);

//
// start here to parse each new Lpacket
// the RxM delivers one complete Lpacket at a time
//
state: nextLpacket

    // received an out-of-sequence moderator -- attempt resync
    event:    RX_receivedLpacket
    condition: RX_Lpacket.service == MODERATOR_TAG
    destination: rxMod3                // unexpected moderator

```

```

// ignore most other received Lpackets
event:    RX_receivedLpacket
condition: RX_Lpacket.service != MODERATOR_TAG
destination: nextLpacket

// lost the DLPDU in the middle
event:    ph_frame_indication == FALSE
destination: endFrame
action:
    DLL_event(DLL_EV_badFrame, macSrce);
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

// the DLPDU was aborted
event:    RX_abort
destination: endFrame
action:
    DLL_event(DLL_EV_rxAbort);
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

// if bad FCS, or DLPDU extended into the guardband, treat as bad DLPDU
event:    RX_endMAC
condition: NUT_timer.count <= DLL_SM_current.gb_start || !RX_FCSOK
destination: endFrame
action:
    DLL_event(DLL_EV_badFrame, macSrce);
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

// good DLPDU, and not guardband time
// use the source ID
event:    RX_endMAC
condition: NUT_timer.count > DLL_SM_current.gb_start && RX_FCSOK
destination: endFrame
action:
    qlowman = min(qlowman, macSrce);
    DLL_event(DLL_EV_rxGoodFrame);
    if(itr != macSrce) DLL_event(DLL_EV_nonconcurrency);
    itr = macSrce;
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

//
// deal with a duplicate node indication, local MAC ID already in by another node
//
state: dupNode

// bad DLPDU so ignore indication
event:    ph_frame_indication == FALSE
destination: endFrame
action:
    DLL_event(DLL_EV_badFrame, macSrce);
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

// bad DLPDU so ignore indication
event:    RX_endMAC
condition: !RX_FCSOK
destination: endFrame
action:
    DLL_event(DLL_EV_badFrame, macSrce);
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

// good DLPDU so accept the duplicate indication
event:    RX_endMAC
condition: RX_FCSOK
destination: endFrame
action:
    netState = DUPNODE;
    if (!dupflag)
    {
        dupflag == TRUE;
        DLL_event(DLL_EV_dupNode);
    }
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

//
// Hold off transmission until blanking + 1 octet has expired

```

```
// since the last transmit or receive. It is possible that the
// hold timer expires before the gen_timer, dealing with turnaround time.
//
state: waitForHold

    // time to transmit, but line has activity, treat it as a collision, and back off.
    event: holdoff_timer.expired()
    condition: ph_lock_indication == TRUE
    destination: endFrame
    action:
        DLL_event(DLL_EV_collision);
        gen_timer.start(0,6 usec);

    // no problem, start transmission
    event: holdoff_timer.expired()
    condition: ph_lock_indication == FALSE
    destination: sendHeader
    action:
        // calculate time remaining in this NUT
        tmp = (unsigned)(NUT_timer.count-DLL_SM_current.gb_prestart);
        octetsleft = min(255, (tmp<<1) + tmp + (tmp>>3)) * 2;
        TX_sendHeader(DLL_SM_current.myaddr);

//
// when header has been sent, start first Lpacket
//
state: sendHeader

    // TxLLC has something, send a new Lpacket
    event: TX_headerComplete
    condition: LLC_pickLpacket(scheduled, octetsleft) == TRUE
    destination: sendNextLpacket
    action:
        TX_sendLpacket(LLC_Lpacket);
        octetsleft = octetsleft - LLC_Lpacket.wire_size();

    // nothing to send that fits in the time left, close the DLPDU
    event: TX_headerComplete
    condition: LLC_pickLpacket(scheduled,octetsleft) == FALSE
    destination: sendTrailer
    action:
        TX_sendTrailer();
        if(scheduled // if unable to send all scheduled Lpackets
            && LLC_pickLpacket(scheduled,octetsleft) == TRUE)
        {
            DLL_event(DLL_EV_dribble);
        }

//
// send subsequent Lpackets
//
state: sendNextLpacket

    // TxLLC has something, send a new Lpacket
    event: TX_LpacketComplete
    condition: LLC_pickLpacket(scheduled,octetsleft) == TRUE
    destination: sendNextLpacket
    action:
        TX_sendLpacket(LLC_Lpacket);
        octetsleft = octetsleft - LLC_Lpacket.wire_size();

    // nothing to send that fits in the time left, close the DLPDU
    event: TX_LpacketComplete
    condition: LLC_pickLpacket(scheduled,octetsleft) == FALSE
    destination: sendTrailer
    action:
        TX_sendTrailer();

//
// complete DLPDU termination
//
state: sendTrailer

    event: TX_trailerComplete
    destination: endFrame
    action:
        if(TX_abort == TRUE)DLL_event(DLL_EV_txAbort);
        else DLL_event(DLL_EV_txGoodFrame);
        gen_timer.start(5,2 usec);
        // start hold timer after line is quiet
        holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

//
// almost guardband start time, wait for guardband start then check lots of things
//
state: guardBand

    event: ph_frame_indication == TRUE // received DLPDU takes precedence
    destination: rxFrame // deal with it
```



```

// net change not in progress, this node is not moderator
event:    NUT_timer.count <= DLL_SM_current.gb_start // guardband start time NOW
condition: tMinus == 0
           && SM_listen_only == SM_listen_only_req
           && netState != MOD
destination: rxMod0

// net change not in progress, this node is moderator AND can remain moderator
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: tMinus == 0
           && SM_listen_only == SM_listen_only_req
           && netState == MOD
           && SM_mod_enable == TRUE
           && min(DLL_SM_current.myaddr, qlowman) == DLL_SM_current.myaddr
destination: txMod0

// net change not in progress, this node is moderator, but cannot remain moderator
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: tMinus == 0
           && SM_listen_only == SM_listen_only_req
           && netState == MOD
           && (SM_mod_enable == FALSE
              || min(DLL_SM_current.myaddr, qlowman) != DLL_SM_current.myaddr)
destination: rxMod0
action:
    netState = NOTMOD;
    modcount = 0;
    lowcount = LOWCOUNTINIT;

// net change in progress, this node is not moderator
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: tMinus > 1 && SM_listen_only == SM_listen_only_req && netState != MOD
destination: rxMod0
action:
    tMinus = tMinus - 1;

// net change in progress, this node is moderator, and can remain moderator
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: tMinus > 1
           && SM_listen_only == SM_listen_only_req
           && netState == MOD
           && SM_mod_enable == TRUE
           && min(DLL_SM_current.myaddr, qlowman) == DLL_SM_current.myaddr
destination: txMod0
action:
    tMinus = tMinus - 1;

// net change in progress, this node is moderator, but cannot remain moderator
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: tMinus > 1
           && SM_listen_only == SM_listen_only_req
           && netState == MOD
           && (SM_mod_enable == FALSE
              || min(DLL_SM_current.myaddr, qlowman) != DLL_SM_current.myaddr)
destination: rxMod0
action:
    tMinus = tMinus - 1;
    netState = NOTMOD;
    modcount = 0;
    lowcount = LOWCOUNTINIT;

// net change complete, this node becomes supernode
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: tMinus == 1 && DLL_SM_pending.myaddr == 0
destination: txMod0
action:
    handle_net_change();
    netState = MOD;
    SM_listen_only = FALSE;           // override listen_only

// net change complete, but shall stop transmitting for a time
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: ( tMinus == 1
           && DLL_SM_pending.myaddr != 0
           && DLL_SM_pending.myaddr != DLL_SM_current.myaddr
           || SM_listen_only != SM_listen_only_req
           && SM_listen_only)
destination: rxMod0
action:
    handle_net_change();
    netState = WATCH;               // go to WATCH state
    watch_timer.start(1,25 sec);

// net change complete, this node is not moderator, no significant changes
event:    NUT_timer.count <= DLL_SM_current.gb_start
condition: ( tMinus == 1
           && DLL_SM_pending.myaddr != 0
           && DLL_SM_pending.myaddr == DLL_SM_current.myaddr

```

```

        || SM_listen_only != SM_listen_only_req
        && !SM_listen_only)
    && netState != MOD
destination: rxMod0
action:
    handle_net_change();

// net change complete, this node is moderator, no major changes, and can remain
moderator
event: NUT_timer.count <= DLL_SM_current.gb_start
condition: (
    tMinus == 1
    && DLL_SM_pending.myaddr != 0
    && DLL_SM_pending.myaddr == DLL_SM_current.myaddr
    || SM_listen_only != SM_listen_only_req
    && !SM_listen_only)
    && netState == MOD
    && SM_mod_enable == TRUE
    && min(DLL_SM_pending.myaddr, qlowman) == DLL_SM_pending.myaddr
destination: txMod0
action:
    handle_net_change();

// net change complete, this node is moderator, no major changes, but can't stay
moderator
event: NUT_timer.count <= DLL_SM_current.gb_start
condition: (
    tMinus == 1
    && DLL_SM_pending.myaddr != 0
    && DLL_SM_pending.myaddr == DLL_SM_current.myaddr
    || SM_listen_only != SM_listen_only_req
    && !SM_listen_only)
    && netState == MOD
    && (SM_mod_enable == FALSE
    || min(DLL_SM_pending.myaddr, qlowman) != DLL_SM_pending.myaddr)
destination: rxMod0
action:
    handle_net_change();
    netState = NOTMOD;
    modcount = 0;
    lowcount = LOWCOUNTINIT;

//
// wait for moderator start
//
state: rxMod0

    // got it
    event: ph_frame_indication == TRUE
    destination: rxMod1

    // timed out waiting for moderator
    event: NUT_timer.count <= 20 usec
    destination: waitTone
    action:
        missed_moderator();
        housekeeping();

//
// receive the moderator
//
state: rxMod1

    // bad moderator
    event: ph_frame_indication == FALSE
    destination: badMod
    action:
        DLL_event(DLL_EV_badFrame, macSrce);

    // get MAC source ID
    event: RX_dataReady == TRUE
    destination: rxMod2
    action:
        macSrce = RX_rxData;
        nqlowman = min(nqlowman, macSrce);

//
// continue receiving moderator
//
state: rxMod2

    // bad DLPDU
    event: ph_frame_indication == FALSE
    destination: badMod
    action:
        DLL_event(DLL_EV_badFrame, macSrce);

    // bad moderator DLPDU if it's null
    event: RX_endMAC
    destination: badMod

```

```

    action:
        DLL_event(DLL_EV_badFrame,macSrce);

    // got an Lpacket, save the Lpacket for later, and go check the rest of the DLPDU
    event:   RX_receivedLpacket
    destination: rxMod3

//
// handle the end of a moderator DLPDU
//
state: rxMod3

    // bad DLPDU
    event:   ph_frame_indication == FALSE
    destination: badMod
    action:
        DLL_event(DLL_EV_badFrame,macSrce);

    // bad DLPDU
    event:   RX_abort
    destination: badMod
    action:
        DLL_event(DLL_EV_rxAbort);

    // bad DLPDU if another Lpacket is in the moderator DLPDU
    event:   RX_receivedLpacket
    destination: badMod
    action:
        DLL_event(DLL_EV_badFrame,macSrce);

    // bad DLPDU
    event:   RX_endMAC
    condition: !RX_FCSOK
    destination: badMod
    action:
        DLL_event(DLL_EV_badFrame,macSrce);

    // dupnode!
    event:   RX_endMAC
    condition: RX_FCSOK && macSrce == DLL_SM_current.myaddr
    destination: endFrame
    action:
        netState = DUPNODE;
        if(!dupflag)
        {
            dupflag = TRUE;
            DLL_event(DLL_EV_dupNode);
        }
        gen_timer.start(5,2 usec);
        // start hold timer after line is quiet
        holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

    // good DLPDU, but bad header
    event:   RX_endMAC
    condition: RX_FCSOK
        && macSrce != DLL_SM_current.myaddr
        && (
            | RX_Lpacket.size != MODERATOR_SIZE
            | RX_Lpacket.ctl != MODERATOR_CTL
            | RX_Lpacket.service != MODERATOR_TAG
            | RX_Lpacket.dest != 0xFF)
    destination: badMod

    // moderator, but from an incorrect address (not lowman)
    event:   RX_endMAC
    condition: RX_FCSOK
        && macSrce != DLL_SM_current.myaddr
        && RX_Lpacket.size == MODERATOR_SIZE
        && RX_Lpacket.ctl == MODERATOR_CTL
        && RX_Lpacket.service == MODERATOR_TAG
        && RX_Lpacket.dest == 0xFF
        && macSrce > qlowman
    destination: endFrame
    action:
        DLL_event(DLL_EV_invalidModAddress);
        gen_timer.start(5,2 usec);
        // start hold timer after line is quiet
        holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

    // this is a moderator from a supernode,
    // or this node is at 0xFF so that all others look like a supernode
    // adopt all the parameters from the moderator
    event:   RX_endMAC
    condition: RX_FCSOK
        && macSrce != DLL_SM_current.myaddr
        && RX_Lpacket.size == MODERATOR_SIZE
        && RX_Lpacket.ctl == MODERATOR_CTL
        && RX_Lpacket.service == MODERATOR_TAG
        && RX_Lpacket.dest == 0xFF
        && macSrce <= qlowman

```

```

        && (macSrce == 0 || DLL_SM_current.myaddr == 0xFF)
destination: waitTone
action:
    qlowman = min(qlowman, macSrce);
    if (macSrce == 0 && currentMod != 0) DLL_event(DLL_EV_supernode);
    currentMod = macSrce;
    DLL_currentMod_indication(currentMod);
    DLL_SM_current.NUT_length = moderator.NUT_length;
    DLL_SM_current.gb_prestart = moderator.gb_prestart;
    DLL_SM_current.gb_start = moderator.gb_start;
    DLL_SM_current.gb_center = moderator.gb_center;
    DLL_SM_current.modulus = moderator.modulus;
    DLL_SM_current.blanking = moderator.blanking;
    DLL_SM_current.slotTime = moderator.slotTime;
    DLL_SM_current.smax = moderator.smax;
    DLL_SM_current.umax = moderator.umax;
    processModerator(moderator);
    housekeeping();

// good moderator received, everything matches local copy
// resynchronize and continue
event:    RX_endMAC
condition: RX_FCSOK
    && macSrce != DLL_SM_current.myaddr
    && RX_Lpacket.size == MODERATOR_SIZE
    && RX_Lpacket.ctl == MODERATOR_CTL
    && RX_Lpacket.service == MODERATOR_TAG
    && RX_Lpacket.dest == 0xFF
    && macSrce <= qlowman
    && !(macSrce == 0 || DLL_SM_current.myaddr == 0xFF)
    && moderator.NUT_length == DLL_SM_current.NUT_length
    && moderator.gb_prestart == DLL_SM_current.gb_prestart
    && moderator.gb_start == DLL_SM_current.gb_start
    && moderator.gb_center == DLL_SM_current.gb_center
    && moderator.modulus == DLL_SM_current.modulus
    && moderator.blanking == DLL_SM_current.blanking
    && moderator.slotTime == DLL_SM_current.slotTime
    && moderator.smax == DLL_SM_current.smax
    && moderator.umax == DLL_SM_current.umax
destination: waitTone
action:
    qlowman = min(qlowman, macSrce);
    currentMod = macSrce;
    DLL_currentMod_indication(currentMod);
    processModerator(moderator);
    housekeeping();

// The moderator doesn't match, so this node is a rogue.
// Since in the guardband, go to rxmod for recovery
event:    RX_endMAC
condition: RX_FCSOK
    && macSrce != DLL_SM_current.myaddr
    && RX_Lpacket.size == MODERATOR_SIZE
    && RX_Lpacket.ctl == MODERATOR_CTL
    && RX_Lpacket.service == MODERATOR_TAG
    && RX_Lpacket.dest == 0xFF
    && macSrce <= qlowman
    && !(macSrce == 0 || DLL_SM_current.myaddr == 0xFF)
    && !(
        moderator.NUT_length == DLL_SM_current.NUT_length
        && moderator.gb_prestart == DLL_SM_current.gb_prestart
        && moderator.gb_start == DLL_SM_current.gb_start
        && moderator.gb_center == DLL_SM_current.gb_center
        && moderator.modulus == DLL_SM_current.modulus
        && moderator.blanking == DLL_SM_current.blanking
        && moderator.slotTime == DLL_SM_current.slotTime
        && moderator.smax == DLL_SM_current.smax
        && moderator.umax == DLL_SM_current.umax
    )
    && in_guardband == TRUE
destination: rxMod0
action:
    qlowman = min(qlowman, macSrce);
    currentMod = macSrce;
    DLL_currentMod_indication(currentMod);
    netState = ROGUE;
    DLL_event(DLL_EV_rogue);

// The moderator doesn't match, so this node is a rogue.
// Since currently not in the guardband, go to endframe for recovery.
event:    RX_endMAC
condition: RX_FCSOK
    && macSrce != DLL_SM_current.myaddr
    && RX_Lpacket.size == MODERATOR_SIZE
    && RX_Lpacket.ctl == MODERATOR_CTL
    && RX_Lpacket.service == MODERATOR_TAG
    && RX_Lpacket.dest == 0xFF
    && macSrce <= qlowman
    && !(macSrce == 0 || DLL_SM_current.myaddr == 0xFF)
    && !(
        moderator.NUT_length == DLL_SM_current.NUT_length
        && moderator.gb_prestart == DLL_SM_current.gb_prestart

```

```

        && moderator.gb_start == DLL_SM_current.gb_start
        && moderator.gb_center == DLL_SM_current.gb_center
        && moderator.modulus == DLL_SM_current.modulus
        && moderator.blanking == DLL_SM_current.blanking
        && moderator.slotTime == DLL_SM_current.slotTime
        && moderator.smax == DLL_SM_current.smax
        && moderator.umax == DLL_SM_current.umax)
    && in_guardband == FALSE
destination: endFrame
action:
    qlowman = min(qlowman, macSrce);
    currentMod = macSrce;
    DLL_currentMod_indication(currentMod);
    netState = ROGUE;
    DLL_event(DLL_EV_rogue);
    gen_timer.start(5,2 usec);
    // start hold timer after line is quiet
    holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

//
// come here to handle a bad moderator DLPDU
// wait for link to become quiet, or until tone generation is due
//
state: badMod

    // This state is reached when this node identifies a moderator DLPDU,
    // but the DLPDU is subsequently reported bad so the state transfers
    // back to endFrame if this node is not in the guardband.
    event: TRUE
    condition: in_guardband == FALSE
    destination: endFrame
    action:
        DLL_event(DLL_EV_badFrame,macSrce);
        gen_timer.start(5,2 usec);
        // start hold timer after line is quiet
        holdoff_timer.start((DLL_SM_current.blanking+1) * 1,6 usec);

    // the guardband should be ended NOW
    event: NUT_timer.count <= 30 usec
    destination: waitTone
    action:
        missed_moderator();
        housekeeping();

    // else wait until the link is quiet
    event: ph_lock_indication == FALSE
    destination: rxMod0

//
// send the moderator, wait for guardband center
//
state: txMod0

    event: NUT_timer.count <= DLL_SM_current.gb_center
    destination: txMod1
    action:
        TX_sendHeader(DLL_SM_current.myaddr); // send header

//
// send more
//
state: txMod1

    event: TX_headerComplete
    destination: txMod2
    action:
        moderator.size = MODERATOR_SIZE;
        moderator.ctl = MODERATOR_CTL;
        moderator.service = MODERATOR_TAG;
        moderator.dest = 0xff;
        moderator.NUT_length = DLL_SM_current.NUT_length;
        moderator.smax = DLL_SM_current.smax;
        moderator.umax = DLL_SM_current.umax;
        moderator.slotTime = DLL_SM_current.slotTime;
        moderator.blanking = DLL_SM_current.blanking;
        moderator.gb_start = DLL_SM_current.gb_start;
        moderator.gb_center = DLL_SM_current.gb_center;
        moderator.usr = usr;
        moderator.interval_count = interval_count;
        moderator.modulus = DLL_SM_current.modulus;
        moderator.tMinus = tMinus;
        moderator.gb_prestart = DLL_SM_current.gb_prestart;
        moderator.spare = 0;

        TX_sendLpacket(moderator); // send the Lpacket

//
// finish up

```

```
//
state: txMod2

    event:      TX_LpacketComplete
    destination: txMod3
    action:
        TX_sendTrailer();

//
// at completion, transferring to waiting for tone
//
state: txMod3

    event:      TX_trailerComplete
    destination: waitTone
    action:
        currentMod = DLL_SM_current.myaddr;
        DLL_currentMod_indication(currentMod);
        housekeeping();

//
// wait for tone
//
state: waitTone

    // end of this NUT, and transferring to off-line
    event:      NUT_timer.count == 0
    condition:   SM_online == FALSE
    destination: offline
    action:
        DLL_tone_indication(interval_count);
        DLL_online_confirm(SM_online);

    // end of this NUT, start of another
    event:      NUT_timer.count == 0
    condition:   SM_online_req == TRUE
    destination: waitSlotZero
    action:
        DLL_tone_indication(interval_count);
        NUT_timer.restart();
        scheduled = TRUE;
        in_guardband = FALSE;

        // housekeeping

        // If this node detects line activity, but not moderators, attempt to recover.
        // Maybe problem is that the local node is performing housekeeping (and is
        // therefore deaf) during the time the moderator DLPDU is transmitted.

        if ((netState != MOD || netState != NOTMOD)
            && qlowman != 255
            && ph_frame_indication == TRUE)
        {
            deafcount++;
        }
        else
        {
            deafcount = 0;
        }

        gen_timer.start(20 usec); // start timer for start of scheduled time

//
// wait for start of scheduled
//
state: waitSlotZero

    // start a new scheduled token pass
    event:      gen_timer.expired()
    destination: gap
    action:
        itr = -1;
        if (DLL_SM_current.myaddr == 0) // supernode doesn't try to detect lowman
        {
            nqlowman = 0;
            qlowman = 0;
        }
        else // else initialize the detector
        {
            nqlowman = 255;
            qlowman = 255;
        }
        slot_timer.restart();
        gen_timer.start(0,6 usec);
```

19.3 TxLLC

The TxLLC (transmit LLC) shall receive and buffer Lpackets from the upper layers. It shall pick the next Lpacket to be transmitted based on

- the order in which Lpackets were queued;
- attributes of the Lpacket;
- information provided by the Access Control Machine (ACM).

The TxLLC shall present the selection Lpacket to the ACM for transmission.

```
// DLL TX LLC State Machine Description

// This state machine accepts transmit requests from the DLS-user,
// queues and prioritizes them, and submits one Lpacket at a time
// to the ACM based on parameters received from the ACM.

/////////////////////////////////////////////////////////////////
//
// type and constant definitions
//
typedef enum {FALSE=0, TRUE=1} BOOL;
typedef void *IDENTIFIER;

typedef enum { M_0,
               M_1,
               M_ND_plus,
               M_ND_minus } M_SYMBOL;

// These are the three transmit priorities.
// hard assignments are so that the priority can be
// used as an index into an array of FIFOs
// note that HIGH and LOW are unscheduled

typedef enum { SCHEDULED=0,
               HIGH=1,
               LOW=2 } PRIORITY;

typedef enum { OK,
               TXABORT,
               FLUSHED } TXSTATUS; // describes the result of a transmit request.

//
// Lpacket class
//
class Lpacket
{
public:

    // return the size octet of the current Lpacket
    USINT size;

    // return the ctl octet of the current Lpacket
    USINT ctl;

// Lpacket constants: masks for ctl octet

#define FIXEDSCREEN 1
#define TAGPAD 2
#define DATAPAD 4

    // return the value of the tag pad bit
    int tag_pad(void)
    {
        return (ctl&TAGPAD) >>1;
    }

    // return the value of the data pad bit
    int data_pad(void)
    {
        return (ctl&DATAPAD) >>2;
    }

    // return the number of octets in the Lpacket
    int wire_size(void)
    {
        return size*2 - tag_pad() - data_pad();
    }

    // store next octet to the Lpacket
    void put_octet(USINT data);

    // get an octet from the Lpacket
```

```

USINT &operator[] (int index);

Lpacket(void *p)          // constructor
{
}

Lpacket(int size)          // constructor
{
}
};

//
// superclass of Lpacket that defines an Lpacket to be transmitted
//
class txLpacket : public Lpacket
{
public:

    IDENTIFIER id;
    BOOL fixed;

    // constructors

    txLpacket(void *p) : Lpacket(p)
    {
    }

    txLpacket(int size) : Lpacket(size) // size is size of PDU buffer
    {
        // (Lpacket plus pads, in octets)
    }
};

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
// interface to DLS-user
//
void DLL_xmit_fixed_request(
    IDENTIFIER id,
    USINT      Lpacket[],
    UINT       size,
    PRIORITY   priority,
    USINT      service,
    USINT      destID);

extern void DLL_xmit_fixed_confirm (IDENTIFIER id, TXSTATUS status);

void DLL_xmit_generic_request(
    IDENTIFIER id,
    USINT      Lpacket[],
    UINT       size,
    PRIORITY   priority,
    USINT      tag[3]);

extern void DLL_xmit_generic_confirm (IDENTIFIER id, TXSTATUS status);

void DLL_flush-requests-by-QoS_request (PRIORITY priority);
extern void DLL_flush-requests-by-QoS_confirm (PRIORITY priority);
DLL_flush_single_request( PRIORITY priority, IDENTIFIER xmit_id );

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
// interface to ACM
//
txLpacket *pickLpacket(BOOL scheduled, int octetsLeft); // pick an Lpacket to send next
void LpacketSent(TXSTATUS status);                     // the Lpacket was sent

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
// interface to station management
//
void SM_powerup(void); // input indication: powerup has occurred

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
// a class to represent message FIFOs
//
class FIFO
{
public:

    void put(txLpacket lp); // add an Lpacket to the FIFO
    txLpacket &get(void);   // remove an Lpacket from the FIFO
    txLpacket &peek(void);  // look at the first Lpacket in the FIFO
};

```



```

void flush(void);           // delete all Lpackets in the FIFO
void flush1(IDENTIFIER id); // delete one Lpacket in the FIFO, given it's ID
BOOL has_data();           // true if the FIFO is not empty
};

FIFO fifo[3];               // at least three priority levels shall be provided

////////////////////////////////////
//
// TxLLC implementation
//
//
// powerup initialization
//
void SM_powerup(void)
{
    fifo[SCHEDULED].flush();
    fifo[HIGH].flush();
    fifo[LOW].flush();
}

//
// flush a particular FIFO
//
void DLL_flush-requests-by-QoS-request (PRIORITY priority)
{
    fifo[priority].flush();
    DLL_flush-requests-by-QoS-confirm(priority);
}

//
// flush a particular Lpacket
//
void DLL_flush_single_request (PRIORITY priority, IDENTIFIER id)
{
    fifo[priority].flush1(id);
}

//
// requests from DLS-user to submit Lpackets for transmission
//
void DLL_xmit_fixed_request(
    IDENTIFIER id,
    USINT      Lpacket[],
    UINT       size,
    PRIORITY   priority,
    USINT      service,
    USINT      destID)
{
    int tmp;

    tmp = size + 4 + (size&1); // size of DLSdu plus 4 octet fixed Lpacket header
                                // plus optional data pad

    txLpacket &lp(new txLpacket(tmp)); // create a new Lpacket buffer

    // build the PDU (Lpacket)
    lp[0] = tmp/2; // size, in words
    lp[1] = 0x01 | ((size&1)<<2); // ctl octet, plus data pad bit, if needed
    lp[2] = service; // fixed tag service
    lp[3] = destID; // destination MAC ID
    memcpy(&lp[4], Lpacket, size); // copy the rest of the data
    // there may be a pad octet sent after the DLSdu
    lp.id = id; // store the user's id (for the confirmation)
    lp.fixed = TRUE; // store type of Lpacket (fixed)
    fifo[priority].put(lp); // queue the Lpacket
}

void DLL_xmit_generic_request(
    IDENTIFIER id,
    USINT      Lpacket[],
    UINT       size,
    PRIORITY   priority,
    USINT      tag[3])
{
    int tmp;
    int pad;

    tmp = size + 6 + (size&1); // octet size of DLSdu plus 6 octet GEN Lpacket header
                                // plus data pad
    pad = 0;

    txLpacket &lp(new txLpacket(tmp)); // create a new Lpacket buffer

    // build the PDU (Lpacket)
    lp[0] = tmp/2; // size, in words

```

```

lp[1] = 0x12 | ((size&1)<<2);    // control octet,
                                // plus maybe a data pad bit,

lp[2] = 0;                      // tag pad octet (affects memory image of Lpacket)
lp[3] = tag[0];                 // generic tag
lp[4] = tag[1];                 // generic tag
lp[5] = tag[2];                 // generic tag
memcpy(&lp[6], Lpacket, size);  // copy the rest of the data
                                // there may be pad octet after the DLSdu

lp.id = id;                     // store the user's identifier (for the confirm)
lp.fixed = FALSE;               // store type of Lpacket (generic)

fifo[priority].put(lp);         // queue the new Lpacket
}

static txLpacket *picked = 0;    // remember which Lpacket was picked

//
// called by ACM to pick the next Lpacket to be sent out
//
txLpacket *pickLpacket(BOOL scheduled, int octetsLeft)
{
    int wordsleft = (octetsLeft+1)/2; // wordsleft is rounded up, accepting that in
    some                             // cases this results in an Lpacket not being sent

    if(scheduled)                 // if scheduled, only look at scheduled FIFO
    {
        // if there is anything in the FIFO and it fits in the time left
        if(fifo[SCHEDULED].has_data()
            && fifo[SCHEDULED].peek().size <= wordsleft)
        {
            picked = &fifo[SCHEDULED].get(); // then pick it to be sent
        }
        else picked = 0;           // no Lpacket available
    }
    else                          // if unscheduled -- look at all FIFOs in order
    {
        // if there is anything in the FIFO and it fits in the time left
        if(fifo[SCHEDULED].has_data()
            && fifo[SCHEDULED].peek().size <= wordsleft)
        {
            picked = &fifo[SCHEDULED].get(); // then pick it to be sent
        }
        else if(fifo[HIGH].has_data() // ditto for HIGH
            && fifo[HIGH].peek().size <= wordsleft)
        {
            picked = &fifo[HIGH].get();
        }
        else if(fifo[LOW].has_data() // ditto for LOW
            && fifo[LOW].peek().size <= wordsleft)
        {
            picked = &fifo[LOW].get();
        }
        else picked = 0;           // no Lpacket available
    }

    return picked;
}

//
// confirmation from the ACM that the picked Lpacket was sent (or possibly aborted)
//
void LpacketSent(TXSTATUS status)
{
    if(picked->fixed)             // if the Lpacket was fixed
    {
        DLL_xmit_fixed_confirm (picked->id, status); // generate a fixed confirm
    }
    else                          // else
    {
        DLL_xmit_generic_confirm (picked->id, status); // generate a generic confirm
    }

    delete picked;               // delete temp data
    picked = 0;
}

```

19.4 RxLLC

The RxLLC (receive LLC) shall buffer Lpackets from the Receive Machine (RxM) as they are assembled. When the RxM indicates that a good DLPDU has concluded, all buffered Lpackets shall be presented to the upper layers in the order received. If the RxM indicates a bad DLPDU has concluded, all buffered Lpackets shall be discarded.

```
// DLL RX LLC State Machine Description

// This state machine gets Lpackets from the RxM,
// and DLPDU status from the deserializer.
// It handles quarantining, and passes quarantined
// Lpackets up to the DLS-user

/////////////////////////////////////////////////////////////////
//
// generic type and constant definitions
//
typedef enum {FALSE=0, TRUE=1} BOOL;

//
// Lpacket class
//
class Lpacket
{
public:

    // the size octet of the current Lpacket
    USINT size;

    // the ctl octet of the current Lpacket
    USINT ctl;

// Lpacket constants: masks for ctl octet

#define FIXEDSCREEN 1
#define TAGPAD 2
#define DATAPAD 4

    // return the value of the tag pad bit
    int tag_pad(void)
    {
        return (ctl&TAGPAD) >>1;
    }

    // return the value of the data pad bit
    int data_pad(void)
    {
        return (ctl&DATAPAD) >>2;
    }

    // return the number of octets in the Lpacket
    int wire_size(void)
    {
        return size*2 - tag_pad() - data_pad();
    }

    // store next octet to the Lpacket
    void put_octet(USINT data);

    USINT &operator[](int index);

    // init for a new Lpacket
    void init(void);

    Lpacket(void *p) {} // constructor
};

class rxLpacket : public Lpacket
{
public:
    USINT    source;        // the MAC ID of the node that sent this Lpacket
    int      memory_size;   // size of the Lpacket in memory
    BOOL     fixed;         // true if fixed , false if generic

    rxLpacket(void *p) : Lpacket(p) {}
};
```

```

/////////////////////////////////////////////////////////////////
//
// a class to represent message FIFOs
//
class FIFO
{
public:

    void put(rxLpacket lp);          // add an Lpacket to the fifo
    rxLpacket &get(void);            // remove an Lpacket from the fifo
    void flush(void);               // delete all Lpackets in the FIFO
    BOOL has_data();                // true if the fifo is not empty
};

/////////////////////////////////////////////////////////////////
//
// a class to represent generic tags
//
class gen_tag
{
    USINT data[3];
public:
    // constructor
    gen_tag(USINT first, USINT second, USINT third)
    {
        data[0] = first;
        data[1] = second;
        data[2] = third;
    }

    USINT &operator[](int index)
    {
        return data[index];
    }
};

/////////////////////////////////////////////////////////////////
//
// internal variables
//

FIFO fifo;

/////////////////////////////////////////////////////////////////
//
// interface to PhL
//
BOOL ph_lock_indication; // optimistic DLPDU detect (clock recovery is tracking)
BOOL ph_frame_indication; // pessimistic DLPDU detect (Manchester valid since the SD)

/////////////////////////////////////////////////////////////////
//
// interface to Deserializer
//
extern BOOL RX_endMAC;      // the end delimiter has been detected
extern BOOL RX_FCSOK;      // FCS on last DLPDU was OK
extern BOOL RX_abort;      // abort received on last DLPDU

/////////////////////////////////////////////////////////////////
//
// interface to RxM
//
BOOL RX_receivedLpacket; // indication: an Lpacket has been received
rxLpacket RX_Lpacket(0); // data: the Lpacket most recently received

/////////////////////////////////////////////////////////////////
//
// Interface to DLS-user
//
DLL_recv_fixed_indication (
    USINT Lpacket[],
    UINT size,
    USINT service,
    USINT sourceID);

DLL_recv_generic_indication (
    USINT Lpacket[],
    UINT size,
    gen_tag tag);

```

```

/////////////////////////////////////////////////////////////////
//
// interface to station management
//
BOOL SM_powerup;    // input indication: powerup has occurred

/////////////////////////////////////////////////////////////////
//
// RxLLC states
//

// wait for powerup
state: powerup
    event:    SM_powerup
    destination: idle
    action:
        fifo.flush();

// wait for a DLPDU to start
state: idle
    event:    ph_frame_indication == TRUE
    destination: getLpacket

state: getLpacket
    // stuff a new Lpacket into FIFO
    event:    RX_receivedLpacket    // wait for a new Lpacket to be assembled
    destination: getLpacket
    action:
        fifo.put(RX_Lpacket);    // stuff the Lpacket into the FIFO

    // if FCS is good, give all Lpackets to next layer up
    event:    RX_endMAC
    condition: RX_FCSOK
    destination: idle
    action:
        rxLpacket &lp(0);

        while(fifo.has_data())
        {
            lp = fifo.get();

            if(lp.fixed)
            {
                DLL_recv_fixed_indication(
                    &lp[4],    // rip off 4 octets of header to leave DLSdu
                    lp.memory_size-4,
                    lp[2],    // send service octet
                    lp.source); // send source MAC ID
            }
            else
            {
                DLL_recv_generic_indication(
                    &lp[6],    // rip off 6 octets of header to leave DLSdu
                    lp.memory_size-6,
                    gen_tag(lp[3],lp[4],lp[5])); // send a three octet tag
            }
        }

    // if DLPDU is damaged, flush all the Lpackets in the FIFO
    event:    RX_endMAC
    condition: !RX_FCSOK
    destination: idle
    action:
        fifo.flush();    // discard all Lpackets

```

19.5 Transmit Machine (TxM)

The transmit machine (TxM) shall receive commands and data from the Access Control Machine to transmit the components of a DLPDU. It shall transmit a DLPDU by passing data octets and commands to the Serializer.

```
// DLL TxM State Machine Description

// This state machine gets commands and Lpackets from the ACM to
// build and transmit a DLPDU. It uses the services of the
// Serializer to do this.

////////////////////////////////////
//
// generic type and constant definitions
//

typedef enum {FALSE=0, TRUE=1} BOOL;

typedef void *IDENTIFIER;

////////////////////////////////////
//
// Lpacket definition
//
// Lpacket constants: masks for ctl octet
//
#define FIXEDSCREEN 1
#define TAGPAD 2
#define DATAPAD 4

//
// Lpacket class
//
class Lpacket
{
public:

    USINT    size;
    USINT    ctl;

    // return the value of the tag pad bit
    int tag_pad(void)
    {
        return (ctl&TAGPAD) >>1;
    }

    // return the value of the data pad bit
    int data_pad(void)
    {
        return (ctl&DATAPAD) >>2;
    }

    // return the number of octets in the Lpacket
    int wire_size(void)
    {
        return size*2 - tag_pad() - data_pad();
    }

    // get the next octet to be transmitted
    USINT get_next_octet(void);

    // is this Lpacket aborted?
    BOOL abort(void);
};

////////////////////////////////////
//
// internal variables
//
int counter;

////////////////////////////////////
//
// interface from ACM and TxLLC
//
BOOL TX_sendHeader;          // command: send preamble, SD, source MAC ID
USINT myaddr;                // input: the source MAC ID
BOOL TX_headerComplete;      // reply: header has been sent

BOOL TX_sendLpacket;         // command: send an Lpacket
class Lpacket lp;            // input: the Lpacket to be sent
BOOL TX_LpacketComplete;     // reply: Lpacket has been sent

BOOL TX_sendTrailer;         // command: send FCS, ED
```

```

BOOL TX_done;           // reply: DLPDU is complete
BOOL TX_abort;          // status: DLPDU was aborted

////////////////////////////////////
//
// interface to Serializer
//
void SER_txRequest(BOOL state);    // command: turn transmitter on and off
void SER_sendData(USINT data);    // command: send a Manchester coded data octet
void SER_sendSD(void);            // command: send a start delimiter
void SER_sendED(void);            // command: send an end delimiter
void SER_clearFCS(void);          // command: clear the FCS accumulator
void SER_sendFCS(void);           // command: send the FCS

BOOL SER_operationComplete;        // reply: current operation is completed

////////////////////////////////////
//
// interface to station management
//
BOOL SM_powerup;          // event indication

////////////////////////////////////
//
// TxM states
//
// wait for powerup
//
state: powerup

    event:    SM_powerup
    destination: idle

//
// wait for a DLPDU to send
//
state: idle

    // when commanded by ACM to start a DLPDU, send the first preamble
    event:    TX_sendHeader
    destination: pre0
    action:
        TX_done = FALSE;
        TX_headerComplete = FALSE;
        TX_abort = FALSE;
        SER_txRequest(TRUE);    // turn on transmitter
        SER_sendData(0xff);    // send first preamble octet

//
// send second preamble octet
//
state: pre0

    event:    SER_operationComplete
    destination: pre1
    action:
        SER_sendData(0xff);    // send second preamble octet

//
// send start delimiter
//
state: pre1

    event:    SER_operationComplete
    destination: sd
    action:
        SER_sendSD();    // send source MAC ID

//
// send MAC ID
//
state: sd

    event:    SER_operationComplete
    destination: finishHeader
    action:
        SER_clearFCS();
        SER_sendData(myaddr);

//
// wait for header complete
//
state: finishHeader

```

```

event:    SER_operationComplete
destination: nextLP
action:
    TX_headerComplete = TRUE;

//
// wait for a command to either send an Lpacket or to terminate the DLPDU
//
state: nextLP

    // start a new Lpacket, grab the size octet
event:    TX_sendLpacket
destination: sendCtl
action:
    counter = lp.wire_size() - 1; // note size zero is not a permitted value
    SER_sendData(lp.get_next_octet());
    TX_LpacketComplete = FALSE;

    // terminate DLPDU: send the FCS
event:    TX_sendTrailer
destination: sendFCS
action:
    SER_sendFCS();

state: sendCtl

    // sent the CTL octet, ignoring tag pad if present
event:    SER_operationComplete
destination: sendData
action:
    counter = counter - 1;
    SER_sendData(lp.get_next_octet());
    if(lp.tag_pad()) // if tag pad
    {
        lp.get_next_octet(); // discard next octet
    }

//
// send the rest of the data in an Lpacket
//
state: sendData

    // if aborted, send an abort sequence: SD followed immediately by ED
event:    SER_operationComplete
condition: lp.abort()
destination: abort1
action:
    SER_sendData(0xff); // required before the Start Delimiter to avoid
                        // violating run length requirement of phy layer

    // if more data, send the next octet
event:    SER_operationComplete
condition: !lp.abort() && counter > 0
destination: sendData
action:
    counter = counter - 1;
    SER_sendData(lp.get_next_octet());

    // when done, wait for the next command
event:    SER_operationComplete
condition: !lp.abort() && counter <= 0
destination: nextLP
action:
    if(lp.data_pad()) // if data pad
    {
        lp.get_next_octet(); // discard next octet
    }
    TX_LpacketComplete = TRUE; // signal that Lpacket is done

//
// send the ED
//
state: sendFCS

    event:    SER_operationComplete
destination: sendED
action:
    SER_sendED();

//
// wait for ED complete, then shut down and wait for next DLPDU
//
state: sendED

    event:    SER_operationComplete
destination: idle
action:

```



```

        SER_txRequest(FALSE);    // turn off transmitter
        TX_done = TRUE;          // signal that DLPDU is done

//
// sent an FF before the abort, when it's done send an Start Delimiter
//
state: abort1

    event:    SER_operationComplete
    destination: abort2
    action:
        SER_sendSD();

//
// send second part of the abort (End Delimiter)
//
state: abort2

    event:    SER_operationComplete
    destination: abort3
    action:
        SER_sendED();

//
// wait for abort complete, then shut down and wait for next DLPDU
//
state: abort3

    event:    SER_operationComplete
    destination: idle
    action:
        SER_txRequest(FALSE);    // turn off transmitter
        TX_abort = TRUE;          // assert error flag
        TX_LpacketComplete = TRUE; // signal that Lpacket is done
        TX_done = TRUE;          // signal that DLPDU is done

```

19.6 Receive Machine (RxM)

The receive machine (RxM) shall receive framing information and data octets from the Deserializer. It shall decode Lpackets from the received octet stream for presentation to the ACM and the RxLLC. It shall accept generic and fixed Lpackets for which the corresponding tag filter is enabled, and discard Lpackets that are not enabled.

```

// DLL RxM State Machine Description

// This state machine gets octet symbols from the Deserializer and uses
// them to build Lpackets, which are then sent to the RxLLC.

////////////////////////////////////
//
// generic type and constant definitions
//

typedef enum {FALSE=0, TRUE=1} BOOL;

typedef void *IDENTIFIER;

////////////////////////////////////
//
// Lpacket definition
//
// Lpacket constants: masks for ctl octet
//
#define FIXEDSCREEN 1
#define TAGPAD 2
#define DATAPAD 4

//
// Lpacket class
//
class Lpacket
{
public:

    // the size octet of the current Lpacket
    USINT size;

    // the ctl octet of the current Lpacket
    USINT ctl;

    // return the value of the tag pad bit
    int tag_pad(void)

```

```

    {
        return (ctl&TAGPAD) >>1;
    }

// return the value of the data pad bit
int data_pad(void)
{
    return (ctl&DATAPAD) >>2;
}

// return the number of octets in the Lpacket
int wire_size(void)
{
    return size*2 - tag_pad() - data_pad();
}

// store next octet to the Lpacket
void put_octet(USINT data);

// get an octet from the Lpacket
USINT operator[](int index);

// init for a new Lpacket
void init(void);
};

class rxLpacket : public Lpacket
{
public:
    USINT source; // the MAC ID of the node that sent this Lpacket
    int memory_size; // size of the Lpacket in memory
    BOOL fixed; // true if fixed , false if generic
};

////////////////////////////////////
//
// a class to represent generic tags
//
class gen_tag
{
    USINT data[3];

public:
    // constructor
    gen_tag(USINT first, USINT second, USINT third)
    {
        data[0] = first;
        data[1] = second;
        data[2] = third;
    }

    USINT operator[](int index)
    {
        return data[index];
    }
};

////////////////////////////////////
//
// the interface to the tag filter lookup tables -- the implementation is unspecified
//
class gen_screener
{
public:
    BOOL enable(gen_tag tag); // set tag, requires tag, TRUE=success/FALSE=failure
    BOOL disable(gen_tag tag); // clear tag, requires tag, TRUE=success/FALSE=failure
    BOOL lookup(gen_tag tag); // lookup tag, requires tag, TRUE=enabled/FALSE=disabled
    void init(void); // powerup init & clear
};

class fixed_screener
{
public:
    BOOL enable(USINT service); // set tag, requires tag, TRUE=success/FALSE=failure
    BOOL disable(USINT service); // clear tag, requires tag, TRUE=success/FALSE=failure
    BOOL lookup(USINT service); // lookup tag, requires tag,
    TRUE=enabled/FALSE=disabled
    void init(void); // powerup init & clear
};

```

```

/////////////////////////////////////////////////////////////////
//
// internal variables
//
// the tag filtering tables
class gen_screener gen;
class fixed_screener fixed;

// MAC ID of source of DLPDU
USINT source;

// an octet counter
int counter;

// temps for generic screener tag octets
USINT gen0;
USINT gen1;

/////////////////////////////////////////////////////////////////
//
// interface to PhL
//
extern BOOL ph_lock_indication; // optimistic DLPDU detect (clock recovery is
tracking)
extern BOOL ph_frame_indication; // pessimistic DLPDU detect (Manchester valid since
the SD)

/////////////////////////////////////////////////////////////////
//
// interface to Deserializer
//
extern BOOL RX_dataReady; // an octet of a DLPDU is available
extern USINT RX_rxData; // octet most recently received from DLPDU (source
MAC ID)
extern BOOL RX_endMAC; // the end delimiter has been detected
extern BOOL RX_FCSOK; // FCS on last DLPDU was OK
extern BOOL RX_abort; // abort received on last DLPDU

/////////////////////////////////////////////////////////////////
//
// interface to station management
class DLL_config_data
{
public:
USINT myaddr;
UINT NUT_length;
USINT smax;
USINT umax;
USINT slotTime;
USINT blanking;
USINT gb_start;
USINT gb_center;
USINT interval_count;
USINT modulus;
USINT gb_prestart;
};

extern DLL_config_data SM_current; // the DLL config variables -- need my MAC ID
extern BOOL SM_powerup; // input indication: powerup has occurred

/////////////////////////////////////////////////////////////////
//
// interface to RxM
//

// received data interface

BOOL RX_receivedLpacket; // an Lpacket has been received
rxLpacket RX_Lpacket; // the Lpacket most recently received

// interface to manage tag filters

extern enable_generic_confirm (IDENTIFIER id, BOOL result);
extern disable_generic_confirm (IDENTIFIER id, BOOL result);
extern enable_fixed_confirm (IDENTIFIER id, BOOL result);
extern disable_fixed_confirm (IDENTIFIER id, BOOL result);

void DLL_enable_generic_request (IDENTIFIER id, gen_tag tag)
{
enable_generic_confirm (id, gen.enable(tag));
}

```

```

void DLL_disable_generic_request (IDENTIFIER id, gen_tag tag)
{
    disable_generic_confirm (id, gen.disable(tag));
}

void DLL_enable_fixed_request (IDENTIFIER id, USINT service)
{
    enable_fixed_confirm (id, fixed.enable(service));
}

void DLL_disable_fixed_request (IDENTIFIER id, USINT service)
{
    disable_fixed_confirm (id, fixed.disable(service));
}

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////
//
//      RxM states
//
//
//
// wait for powerup
//
state: powerup
    event:      SM_powerup
    destination: idle
    action:
        RX_receivedLpacket = FALSE;

//
// wait for a DLPDU to start
//
state: idle
    event:      ph_frame_indication == TRUE
    destination: getID

state: getID
    // if DLPDU is terminated, wait for next DLPDU
    event:      ph_frame_indication == FALSE
    destination: idle

    // get the source MAC ID and save it for all the Lpackets in the DLPDU
    event:      RX_dataReady == TRUE
    destination: getSize
    action:
        source = RX_rxData;    // save the source MAC ID for subsequent Lpackets

state: getSize
    // if DLPDU is terminated, wait for next DLPDU
    event:      ph_frame_indication == FALSE
    destination: idle

    // get the size octet and init a new Lpacket
    event:      RX_dataReady == TRUE
    destination: getCtl
    action:
        RX_receivedLpacket = FALSE;    // re-initialize the interface
        RX_Lpacket.init();
        RX_Lpacket.source = source;    // save the source in the new Lpacket
        RX_Lpacket.put_octet(RX_rxData); // store the size octet

state: getCtl
    // if DLPDU is terminated, wait for next DLPDU
    event:      ph_frame_indication == FALSE
    destination: idle

    // get the CTL octet for a fixed Lpacket
    event:      RX_dataReady == TRUE && (RX_rxData & 0xFB) == 0x01    // fixed tag
    destination: getFixed0
    action:
        RX_Lpacket.put_octet(RX_rxData); // store the ctl octet
        counter = RX_Lpacket.wire_size() - 2;    // init the octet counter

    // get the CTL octet for a generic Lpacket
    event:      RX_dataReady == TRUE && (RX_rxData & 0xFB) == 0x12    // generic tag
    destination: getGen0
    action:
        RX_Lpacket.put_octet(RX_rxData); // store the ctl octet
        RX_Lpacket.put_octet(0);    // store the pad octet

```

```

        counter = RX_Lpacket.wire_size() - 2;  // init the octet counter

// get the CTL octet for a generic Lpacket
event:    RX_dataReady == TRUE
        && (RX_rxData & 0xFB) != 0x01
        && (RX_rxData & 0xFB) != 0x12
destination: ignoreRest
action:
        counter = RX_Lpacket.wire_size() - 2;  // init the octet counter

//
// handle service octet of fixed Lpacket
//
state: getFixed0

        // if DLPDU is terminated, wait for next DLPDU
event:    ph_frame_indication == FALSE
destination: idle

        // handle case if tag is enabled
event:    RX_dataReady == TRUE
        && fixed.lookup(RX_rxData)
destination: getFixed1
action:
        RX_Lpacket.put_octet(RX_rxData); // put the next data octet into the Lpacket
        counter--;                      // count it

        // handle case if tag is disabled
event:    RX_dataReady == TRUE
        && !fixed.lookup(RX_rxData)
destination: ignoreRest
action:
        counter--;                      // count it

//
// handle dest octet of fixed Lpacket
//
state: getFixed1

        // if DLPDU is terminated, wait for next DLPDU
event:    ph_frame_indication == FALSE
destination: idle

        // handle case if addressed to me or broadcast
event:    RX_dataReady == TRUE
        && (RX_rxData == SM_current.myaddr || RX_rxData == 0xFF)
destination: getRest
action:
        RX_Lpacket.put_octet(RX_rxData); // put the next data octet into the Lpacket
        counter--;                      // count it
        RX_Lpacket.fixed = TRUE;

        // handle case if not addressed to me
event:    RX_dataReady == TRUE
        && RX_rxData != SM_current.myaddr
        && RX_rxData != 0xFF
destination: ignoreRest
action:
        counter--;                      // count it

//
// handle first tag octet of generic Lpacket
//
state: getGen0

        // if DLPDU is terminated, wait for next DLPDU
event:    ph_frame_indication == FALSE
destination: idle

        // save the next octet
event:    RX_dataReady == TRUE
destination: getGen1
action:
        RX_Lpacket.put_octet(gen0=RX_rxData); // save and put the next octet into
Lpacket
        counter--;                      // count it

//
// handle second tag octet of generic Lpacket
//
state: getGen1

        // if DLPDU is terminated, wait for next DLPDU
event:    ph_frame_indication == FALSE
destination: idle

```

```

// save the next octet
event:    RX_dataReady == TRUE
destination: getGen2
action:
    RX_Lpacket.put_octet(gen1=RX_rxData); // save and put next data octet into
Lpacket
    counter--; // count it

//
// handle third octet of generic Lpacket
//
state: getGen2

    // if DLPDU is terminated, wait for next DLPDU
    event:    ph_frame_indication == FALSE
    destination: idle

    // handle case if local node is interested in this generic Lpacket
    event:    RX_dataReady == TRUE
    && gen.lookup(gen_tag(gen0, gen1, RX_rxData))
    destination: getRest
    action:
        RX_Lpacket.put_octet(RX_rxData); // put the next data octet into the Lpacket
        counter--; // count it
        RX_Lpacket.fixed = FALSE;

    // handle case if not interested in this generic Lpacket
    event:    RX_dataReady == TRUE
    && !gen.lookup(gen_tag(gen0, gen1, RX_rxData))
    destination: ignoreRest
    action:
        counter--; // count it

state: getRest

    // if DLPDU is terminated, wait for next DLPDU
    event:    ph_frame_indication == FALSE
    destination: idle

    // get data octets except the last
    event:    RX_dataReady == TRUE
    condition: counter > 1
    destination: getRest
    action:
        RX_Lpacket.put_octet(RX_rxData); // put the next data octet into the Lpacket
        counter--; // count it

    // get the last data octet; handle data pad, indicate arrived Lpacket; wait for
next Lpacket
    event:    RX_dataReady == TRUE
    condition: counter == 1
    destination: getSize
    action:
        RX_Lpacket.put_octet(RX_rxData); // get the last octet
        if(RX_Lpacket.data_pad()) // if a data pad is requested,
        {
            RX_Lpacket.put_octet(0); // insert a zero
        }

        // remove pad octets from data count
        RX_Lpacket.memory_size -= RX_Lpacket.size*2;

        RX_receivedLpacket = TRUE; // notify that that an Lpacket has arrived

state: ignoreRest

    // if DLPDU is terminated, wait for next DLPDU
    event:    ph_frame_indication == FALSE
    destination: idle

    // get data octets except the last
    event:    RX_dataReady == TRUE
    condition: counter > 1
    destination: ignoreRest
    action:
        counter--; // count it

    // handle the last data octet; wait for next Lpacket
    event:    RX_dataReady == TRUE
    condition: counter == 1
    destination: getSize

```

19.7 Serializer

For each command from the TxM, the Serializer shall produce the appropriate M_symbol stream, which shall be output to the medium via the PhL.

```
// DLL octet / octet Serializer

// This state machine gets commands to transmit octet symbols from the TxM.
// It decomposes the octet symbols into M_symbols, and
// uses the services of the PHY layer to send these on the medium.
// This specifies the correct order of transmission of the various octet symbols.

// The FCS algorithm is the same as that used by HDLC

/////////////////////////////////////////////////////////////////
//
// data types
//
typedef enum {FALSE, TRUE} BOOL;

/////////////////////////////////////////////////////////////////
//
// interface to TxM
//

// requests (TxM to SER)

void SER_txRequest(BOOL state);      // command: turn transmitter on and off
void SER_sendData(USINT data);      // command: send a Manchester coded data octet
void SER_sendSD(void);              // command: send a start delimiter
void SER_sendED(void);              // command: send an end delimiter
void SER_clearFCS(void);             // command: clear the FCS accumulator
void SER_sendFCS(void);             // command: send the FCS

// indications (SER to TxM)

extern void SER_operationComplete(void); // notify the TxM that an operation is
complete

/////////////////////////////////////////////////////////////////
//
// interface to the PHY layer
//
typedef enum {M_0, M_1, M_ND_plus, M_ND_minus} M_SYMBOL;

BOOL ph_frame_request;      // transmit enable/disable
M_SYMBOL ph_data_request;   // data to be sent

/////////////////////////////////////////////////////////////////
//
// internal variables
//
static unsigned int accum;    // the FCS accumulator, 16 bits, sign is irrelevant
                             // as long as zeros shift in from the right

/////////////////////////////////////////////////////////////////
//
// internal subroutines
//

// add one octet to the FCS accumulator
// this is just one possible implementation of the HDLC FCS

static void FCS_octet(unsigned int d)
{
    accum ^= d & 0xff;

    for(int i=0;i<8;i++)
    {
        accum = (accum>>1) ^ (0x8408 * (accum&1)); // use FCS-CCITT polynomial
    }
}

/////////////////////////////////////////////////////////////////
//
// Serializer implementation
//

// requests (TxM to SER)

// enable transmitter
```

```

void SER_txRequest(BOOL state)
{
    ph_frame_request = state;
}

// add a data octet to the FCS and send it
void SER_sendData(USINT data)
{
    FCS_octet(data);

    // transmit data bits 0 through 7
    ph_data_request = (data & 0x01) ? M_1 : M_0;
    ph_data_request = (data & 0x02) ? M_1 : M_0;
    ph_data_request = (data & 0x04) ? M_1 : M_0;
    ph_data_request = (data & 0x08) ? M_1 : M_0;
    ph_data_request = (data & 0x10) ? M_1 : M_0;
    ph_data_request = (data & 0x20) ? M_1 : M_0;
    ph_data_request = (data & 0x40) ? M_1 : M_0;
    ph_data_request = (data & 0x80) ? M_1 : M_0;

    SER_operationComplete();    // tell TxM that operation is complete
}

// send a start delimiter
void SER_sendSD(void)
{
    ph_data_request = M_ND_plus;
    ph_data_request = M_0;
    ph_data_request = M_ND_minus;
    ph_data_request = M_ND_plus;
    ph_data_request = M_ND_minus;
    ph_data_request = M_1;
    ph_data_request = M_0;
    ph_data_request = M_1;
    SER_operationComplete();    // tell TxM that operation is complete
}

// send an end delimiter
void SER_sendED(void)
{
    ph_data_request = M_1;
    ph_data_request = M_0;
    ph_data_request = M_0;
    ph_data_request = M_1;
    ph_data_request = M_ND_plus;
    ph_data_request = M_ND_minus;
    ph_data_request = M_ND_plus;
    ph_data_request = M_ND_minus;
    SER_operationComplete();    // tell TxM that operation is complete
}

// initialize the FCS accumulator
void SER_clearFCS(void)
{
    accum = 0xffff;    // init the accumulator to all ones
    SER_operationComplete();    // tell TxM that operation is complete
}

// send the FCS
void SER_sendFCS(void)
{
    int tmp;

    tmp = accum ^ 0xffff;    // invert the FCS before sending;
    SER_sendData(tmp);    // do not calculate FCS while sending itself
    SER_sendData(tmp>>8);
    SER_operationComplete();    // tell TxM that operation is complete
}

```

19.8 Deserializer

19.8.1 Octet construction

The Deserializer shall accept M_symbols from ph_data_indication. M_symbols at the start of a frame shall be ignored until the ph_frame_indication line transitions from false to true. This transition shall start on the first data octet – the source MAC ID. The Deserializer shall then group subsequent sets of eight MAC data symbols into octets. As each subsequent data octet is ready, the Deserializer shall transition the RX_dataReady line from false to true.

19.8.2 FCS checking

A modified CCITT Cyclic Redundancy Check shall be performed on the transferred data to verify the frame check sequence (FCS). The FCS generator and checker shall be as specified in 5.1 for the Type 2 protocol.

NOTE The last two data octets in the DLPDU (FCS octets) are calculated by the transmitting node so that, in the absence of errors, the remainder in the receiving node (after right-shifting into a 16-bit shift register) is always 0xF0B8 (see 5.1.2).

The receive FCS process shall begin by pre-setting the FCS checker to 0xFFFF when the `ph_frame_indication` line transitions from false to true. All subsequent data bits on the `ph_data_indication` line, excluding the end delimiter, shall be applied to the FCS checker. At end of receiving a DLPDU, the remainder (FCS) shall be compared to 0xF0B8.

19.8.3 End of DLPDU processing

Data octet processing shall proceed until `ph_frame_indication` transitions from true to false. This transition can happen mid-octet, but no action shall be taken until the start of the next octet. This time shall mark the end of reception for this frame.

The `ph_status_indication` received after `ph_frame_indication` transitions from true to false shall determine which of the following actions to take:

- if Normal – set `RX_FCSOK` to true if (FCS remainder == 0xF0B8); otherwise, set to false;
- if Abort – set `Rx_abort` to true; set `Rx_FCSOK` to false;
- if Invalid – set `Rx_abort` to false; set `Rx_FCSOK` to false.

19.9 DLL management

DLL Management shall buffer the station management variables that are required for the DLL to operate. It shall manage synchronized changes on these variables via interfaces to Station Management and the ACM.

```
// DLL Station Management Interface Description

// This component holds the station management variables that are
// subject to synchronized change.

////////////////////////////////////
//
// type and constant definitions
//
typedef enum {FALSE=0, TRUE=1} BOOL;

typedef void *IDENTIFIER;    // The data type of the identifier is unspecified.

// config data definition
class DLL_config_data
{
public:
    USINT myaddr;
    UINT  NUT_length;
    USINT smax;
    USINT umax;
    USINT slotTime;
    USINT blanking;
    USINT gb_start;
    USINT gb_center;
    USINT interval_count;
    USINT modulus;
    USINT gb_prestart;
};
```

```

////////////////////////////////////
//
// interface to station management
//
void DLL_tMinus-start-countdown-request (IDENTIFIER id, USINT start_count);
extern void DLL_tMinus-start-countdown-confirm (IDENTIFIER id, BOOL result);

void DLL_set_pending_request (IDENTIFIER id, DLL_config_data params);
extern void DLL_set_pending_confirm (IDENTIFIER id, BOOL result);

void DLL_get_pending_request (IDENTIFIER id);
extern void DLL_get_pending_confirm (IDENTIFIER id , DLL_config_data params);

void DLL_set_current_request (IDENTIFIER id, DLL_config_data params);
extern void DLL_set_current_confirm (IDENTIFIER id, BOOL result);

void DLL_get_current_request (IDENTIFIER id);
extern void DLL_get_current_confirm (IDENTIFIER id , DLL_config_data params);

////////////////////////////////////
//
// interface to ACM
//
extern void DLL_tminus_zero_indication (); // ACM calls this when tMinus==0 occurs
DLL_config_data DLL_SM_pending; // the ACM needs to see both current and pending
DLL_config_data DLL_SM_current;
extern ACM_tMinus; // poke the ACM's tMinus counter to get
// a synchronized change started. The ACM
// continually writes over this every time the
// moderator is received, unless it is the moderator

void SM_supernode(void); // if this node saw a moderator sent by a supernode,
// any pending SM update shall be cancelled

////////////////////////////////////
//
// internal variables
//
BOOL pending_changed = FALSE;

////////////////////////////////////
//
// write the pending copy of DLL_config
void DLL_set_pending_request (IDENTIFIER id, DLL_config_data params)
{
    DLL_SM_pending = params; // save the params in the pending buffer
    pending_changed = TRUE;
    DLL_set_pending_confirm (id, TRUE);
}

// write the current copy of DLL_config
// ONLY USED WHEN OFFLINE
void DLL_set_current_request (IDENTIFIER id, DLL_config_data params)
{
    DLL_SM_current = params; // save the params in the pending buffer
    DLL_set_current_confirm (id, TRUE);
}

// read the pending copy of DLL_config
void DLL_get_pending_request (IDENTIFIER id)
{
    if(pending_changed == TRUE) // if pending has been updated
    {
        // return the contents of the pending buffer
        DLL_get_pending_confirm (id , DLL_SM_pending);
    }
    else // if pending buffer is invalid
    {
        // return the contents of the current buffer
        DLL_get_current_confirm (id , DLL_SM_current);
    }
}

//
// read the current copy of DLL_config
//
void DLL_get_current_request (IDENTIFIER id)
{
    // return the contents of the current buffer
    DLL_get_current_confirm (id , DLL_SM_current);
}

//
// called by the ACM when a tMinus countdown has completed

```

```
//
void DLL_tminus_zero_indication(void)
{
    if(pending_changed)
    {
        DLL_SM_current = DLL_SM_pending;
        pending_changed = FALSE;
    }
}

//
// start a synchronized change sequence
//
void DLL_tMinus-start-countdown-request (IDENTIFIER id, USINT start_count)
{
    ACM_tMinus = start_count;    // the ACM polls this variable
    DLL_tMinus-start-countdown-confirm (id, TRUE);
}

//
// cancel pending change if a supernode has been seen
//
void SM_supernode(void)
{
    pending_changed = FALSE;
}
```

20 Type 3: Overview of the DL-protocol

NOTE Annex E specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. The specification of Annex E is complementary to the textual specification in this and related clauses in the body of this standard. In case of conflict the requirements of Annex E take precedence.

20.1 General

From the requirements of the various application fields, for example, process control, factory automation, power distribution, building automation, primary process industry etc., the following characteristic features of the fieldbus Data Link protocol are resulting (see Table 195).

Table 195 – Characteristic features of the fieldbus Data Link protocol

Feature	Description
Station types	Masters (active stations, with bus access control); Slaves (passive stations, without bus access control); preferably at most 32 masters, optionally up to 127, if the applications are not time critical
Station addressing	0 to 127 (127 = global addresses for broad-cast and multicast messages), address extension for region/DL-segment address and service access address (DLSAP), 6 bit each
Bus access	Hybrid: decentralized and central; "Token Passing" between Master stations and "Master-Slave" between Master and Slave stations
Data transfer services	Send Data with/without Acknowledge Send and Request Data with Reply Send and Request Data with Multicast Reply Clock Synchronization
DLPDU length	max. 255 octets per DLPDU, 0 to 246 octets for each data unit without address extension
Transmission speed	Depending on network topology and cable lengths, for example, step-wise from 9,6 to 12000 kBit/s
Transmission characteristic	Half duplex, asynchronous and synchronous transmission
Data integrity asynchronous transmission	Messages with Hamming distance (Hd) = 4, sync slip detection, special sequence to avoid loss or duplication of data
Data integrity synchronous transmission	Hamming distance (Hd) = 4 for DLPDU lengths shorter than 255 octets and Hd = 5 for lengths shorter than 15 octets, special sequence to avoid loss or duplication of data
NOTE 1 The DLPDU format for asynchronous transmission is the format FT 1.2 (asynchronous transmission with start-stop synchronization) for Telecontrol Equipment and Systems, which is specified in IEC 60870–5–1.	
NOTE 2 The DLPDU format for synchronous transmission is based on octet characters.	

20.2 Overview of the medium access control and transmission protocol

The fieldbus Data Link Layer uses controlled medium access accomplished by a hybrid medium access method: a decentralized method according to the principle of **Token Passing** is underlain by a central method according to the **Master-Slave** principle. Medium access control may be exercised by each Master station (active station). The Token Passing is characterized by the following features:

a) Token Passing allows fair media access for all token holders.

EXAMPLE: When 4 token holders produce the same amount of similar priority data they will share the media so that on average each of them can use 25% of the available message transfer time. With the Token Passing procedure, rules exist to share the message transfer time between the token holders without discrimination of any of them. In case of usage of the whole of the available token holding time by one token holder in one token rotation this token holder is limited to one high priority message in the next token rotation after which it has to transfer the token immediately.

b) Token Passing guarantees short reaction time.

EXAMPLE: For urgent messages, a maximum delay is specified for delivery of the information. Thus, a configurable time (T_{TR}) specifies the target rotation time. Independent of the number of token holders and the amount of messages that have to be sent, at least one urgent message can be sent by each token holder. Thus, a short reaction time is guaranteed for one urgent message from each token holding station in each token holding period.

c) Token Passing allows flexible (re-)configuration.

EXAMPLE: When a token holder is switched on or off, it will be automatically included or excluded in the logical ring. In the next token rotation after detection and inclusion in the logical token ring, the new token holder has the same rights to send messages as the other token holders. The sharing of the bus message transfer time is organized automatically without changing of parameters of the other token holders.

Medium access control may be exercised by each Master station (active station) if the station has the token. The token is passed from Master station to Master station in a logical ring and thus determines the instant, when a Master station may access the medium. Controlled token passing is managed by each station knowing its predecessor (Previous station, PS), the station from which it receives the token. Furthermore, each station knows its successor (Next station, NS), that is, the station to which the token is transmitted, and its own address (This station, TS). Each Master station determines the PS and NS addresses after the initialization of the operating parameters for the first time and then later dynamically according to the algorithm described in 20.3.2.4.

The Master-Slave principle is characterized by the following features:

Communication is always initiated by a Master station which has the permission for medium access, the token. Slave stations (passive stations) act neutrally in respect to medium access, that is, they do not transmit independently but only on request. If the logical ring consists of only one Master and several Slave stations then it is a pure Master-Slave system.

The following error conditions, exceptions and operational states in the system are dealt with:

- multiple tokens
- lost token
- error in token passing
- duplicate station addresses
- stations with faulty transmitter or receiver
- adding and removing stations during operation
- any combinations of Master and Slave stations

20.3 Transmission modes and DL-entity

20.3.1 Overview

The exchange of messages takes place confirmed or unconfirmed. A confirmed message exchange consists of a Master station's send/request DLPDU and the associated acknowledgement or response DLPDU of a Master or a Slave station. User data may be transmitted in the send/request DLPDU as well as in the response DLPDU. The acknowledgement DLPDU does not contain any user data (for DLPDU formats see clause 22).

An unconfirmed message exchange takes place only in case of token transmission and in case of the transmission of data without acknowledgement (for example, necessary for broadcast messages). In both modes of operation, there is no acknowledgement. In broadcast messages a Master station (initiator) addresses all other stations at the same time by means of a global address (highest station address, all address bits are binary "1").

All stations except the respective token holder (initiator) shall in general monitor all requests. The stations acknowledge or respond only when they are addressed. The acknowledgement or response shall arrive within a predefined time, the slot time, otherwise the initiator repeats, depending on the predefined retry limit, the request if it is not a "first request" (see 21.4, FCB). A new request or a token shall not be issued by the initiator before the expiration of a waiting period after receiving an acknowledge or response, the idle time (see 20.5).

If the responder does not acknowledge or respond after a predefined number of retries (see Table 198), it is marked as "non-operational". If a responder is "non-operational", a later unsuccessful request will not be repeated.

The modes of transmission operation define the sequence of the message transfer periods. Three modes are distinguished:

- 1) Token handling
- 2) Send operation
- 3) Send and request operation

20.3.2 Token procedures

20.3.2.1 Token circulation

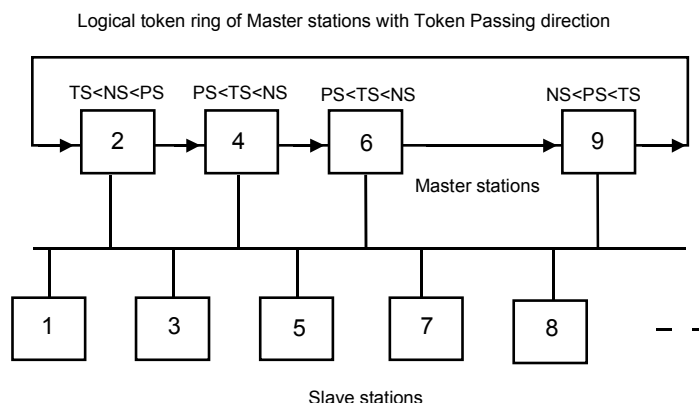
The token is passed from Master station to Master station in ascending numerical order of station addresses by means of the token DLPDU (see 22.4). To close the logical token ring, the station with the highest address passes the token to the station with the lowest address, see Figure 46.

20.3.2.2 Token reception

If a Master station (TS) receives a token DLPDU addressed to itself from a station which is registered as Previous station (PS) in its list of Master stations (LMS), it **owns the token** and may execute message transfer periods. The LMS is generated in a Master station in the "Listen_Token" state (see 23.2.4) after power on and is updated and corrected, if necessary, later on upon each receipt of a token DLPDU.

If the token transmitter is not the registered PS, the addressee assumes an error and shall not accept the token. Only a subsequent retry of the same PS is accepted and results in the token receipt, because the token receiver shall assume now that the logical ring has changed. It replaces the originally recorded PS in its LMS by the new one.

Figure 46 shows the Logical Token Passing ring.



where

TS is This station (address)
 PS is Previous station (address)
 NS is Next station (address)
 Master stations are 2, 4, 6 and 9 (addresses as an example)
 Slave stations are 1, 3, 5, 7 and 8 (addresses as an example)

Figure 46 – Logical Token Passing ring

20.3.2.3 Token transmission

After the Master station has finished its message transfer periods - contingent maintenance of the GAP station list (GAPL, see 20.3.2.4) included - it passes the token to its successor (NS) by transmitting the token DLPDU. The functionality of its transceiver is checked by simultaneous monitoring (see 23.2.11, "Pass-Token" state).

If, after transmitting the token DLPDU and after expiration of the synchronization time within the slot time (see 20.5), the token transmitter receives a valid DLPDU, that is, a DLPDU header without any errors, it assumes that its NS owns the token and executes message transfer periods. If the token transmitter receives an invalid DLPDU, it assumes that another Master station is transmitting. In both cases it ceases monitoring the token passing and retires, that is, it enters the "Active_Idle" state (see 23.2.5).

If the token transmitter does not recognize any bus activity within the slot time, it repeats the token DLPDU and waits another slot time. It retires thereafter, if it recognizes bus activity within the second slot time. Otherwise, it repeats the token DLPDU to its NS for a last time. If, after this second retry, it recognizes bus activity within the slot time, it retires.

If, after the second retry, there is no bus activity, the token transmitter tries to pass the token to the next but one Master station (NS of NS). It continues repeating this procedure until it has found a successor from its LMS. If it does not succeed, the token transmitter assumes that it is the only one left in the logical token ring and transmits the token to itself. If it finds a NS again in a later station registration, it tries again to pass the token.

20.3.2.4 Addition and removal of stations

Master and Slave stations may be connected to or disconnected from the transmission medium at any moment. Each Master station in the logical token ring is responsible for the addition of new stations in the range from the own station address (TS) to the next station (NS), excluding TS and NS. This address range is called GAP and is represented in the GAP List (GAPL), except the address range between Highest Station Address (HSA: 2 to 126) and 127, which does not belong to the GAP.

Each Master station in the logical token ring examines its address range (all GAP addresses) after expiration of the GAP update timer (see 20.5.3.11) for changes concerning Master and Slave stations. This is accomplished by examining **one** address per token receipt, using the "Request DL Status with Reply" request DLPDU (see Table 196, b7=1, Code-No 9: Format 22.1.1 a) and 22.1.2 a)).

Upon receiving the token, GAP maintenance starts immediately after all queued message transfer periods have been executed, if there is still transmission time available (see 20.3.2.6). Otherwise, GAP maintenance starts upon the next or the consecutive token receipts after the high priority message transfer periods and other low priority services invoked before the previous GAP maintenance have been performed (see 20.3.2.7). In realizations, care is necessary to ensure that GAP maintenance and low priority message transfer periods do not block each other.

GAP addresses are examined in ascending order, except the GAP which surpasses the HSA. For example, if the HSA and address 0 are not used by a Master station, the Master station with the highest address examines address 0 after checking the HSA.

If a station acknowledges positively with the DL status "Master station not ready to enter logical token ring" or "Slave station" (see Table 196, b7=0, Code-No 0, no SC, and Figure 64), it is accordingly marked in the GAPL and the next address is checked. If a station answers with the state "Master station ready to enter logical token ring", the token holder changes its GAPL and LMS and passes the token to the new NS. This station, which has newly been admitted to the logical token ring, has already built up its LMS, when it was in the "Listen_Token" state, so that it is able to determine its GAP range and its NS.

If a station answers with the DL status "Master station in logical token ring", then the token holder does not change its GAP and passes the token to the NS given in the LMS. Thus the skipped Master station may retire from the bus in case of permanent not receiving the token. In this case the Master station shall enter the "Listen_Token" state. In this state it generates a new LMS and remains in this state until it is addressed once more by a "Request DL Status with Reply" transmitted by its predecessor (PS).

Stations which were registered in the GAPL and which do not respond to a "Request DL Status with Reply" are removed from the GAPL and are recorded as unused station addresses.

Due to performance requirements repeated requests of "Request DL Status with Reply" are not desired.

20.3.2.5 (Re)initialization of the logical token ring

Initialization is primarily a special case of updating the LMS and GAPL. If after power on (PON) of a Master station a time-out is encountered in the "Listen_Token" state (no bus activity within Time-out Time T_{TO} (see 20.5)), then the Master station shall claim the token ("Claim_Token" state) and start initialization.

When the entire fieldbus system is started, the Master station with the lowest station address starts initialization. By transmitting two token DLPDUs addressed to itself (DA = SA = TS) it informs any other Master stations (entering a NS into their LMS) that it is now the only station in the logical token ring. Then it transmits a "Request DL Status with Reply" DLPDU to each station in an incrementing address sequence, in order to register other stations. If a station responds with "Master station not ready to enter logical token ring" or with "Slave station" it is entered in the GAPL. The first Master station, which answers with "Master station ready to enter logical token ring", is registered as NS in the LMS and thus closes the GAP range of the token holder. Then the token holder passes the token to its NS.

Reinitialization becomes necessary after loss of the token, which causes a time-out (no bus activity). In this case, an entire bus initialization sequence is not required, because LMS and GAPL already exist in the Master stations. The time-out expires first in the Master station with the lowest address. It takes the token and starts executing regular message transfer periods or passes the token to its NS.

20.3.2.6 Token rotation time

After receiving a token, the DL-entity may carry out high priority and low priority message transfer periods according to the token rotation time. The token rotation time is measured by using the token-rotation-timer. At the beginning the token-rotation-timer is loaded with the target rotation time T_{TR} and decremented with each bit time. The token-rotation-timer stops if the value zero is reached.

The token rotation time shall be measured according to the following rules:

- immediately after token reception the Master station shall read the current value of the token-rotation-timer (token holding time T_{TH}) to calculate the real rotation time T_{RR} (difference between the target rotation time and the current value of token-rotation-timer) and shall start the token-rotation-timer with the value target rotation time (T_{TR})
- the T_{RR} represents always the token rotation time of the last token rotation (from the viewpoint of this station)

A system's minimum target rotation time depends on the number of Master stations and thus on the token transfer period (T_{TP}) and the duration of high priority message transfer periods (high T_{MP}). The predefined target rotation time T_{TR} shall also contain sufficient time for low priority message transfer periods and a safety margin for potential retries.

In order to keep within the system reaction time required by the field of application, the target rotation time T_{TR} of the token in the logical ring shall be specified.

The system reaction time is defined as the maximum time interval (worst case) between two consecutive high priority message transfer periods of a Master station, measured at the DLS-user interface at maximum bus load.

In order to achieve a target rotation time as short as possible, it is recommended for the DLS-user (see IEC 61158-5 Type 3), to declare only important data as high priority message transfer periods and to restrict their length (for example, ≤ 32 octets for the DLSDU, see 21.6).

If the transfer periods defined in 20.6 (equations (65) and (72) as well as (66) and (73)) are included and possible retries are taken into consideration, the operating parameter "target rotation time T_{TR} " (see 20.6 and Table 198), which is necessary for initialization (min T_{TR}), is calculated for the DLS-user as follows:

$$\min T_{TR} = n_a \times T_{TP} + (n_a + 1) \times \text{high } T_{MP} + k \times \text{low } T_{MP} + m_t \times T_{RMP} \quad (24)$$

where

n_a is the number of Master stations

k is the estimated number of low priority message transfer periods per token rotation

T_{TP} is the token transfer period (see 20.6.1.1 and 20.6.2.1)

T_{MP} is the message transfer period, depending on DLPDU length (see 20.6.1.2 and 20.6.2.2)

mt is the numbers of message retry transfer periods per token rotation

T_{RMP} is the message retry transfer period

The first term contains one token transfer period per Master station. The second term contains **one** high priority message transfer period for $N+1$ Master stations. The third term contains the estimated number of low priority message transfer periods per token rotation. The fourth term serves as a safety margin for potential retries. The maximum reaction time for high priority message transfer periods is guaranteed for all bus loads.

20.3.2.7 Message priorities

In the parameter `Service_class` of the DL services the DLS-user (application layer) can choose between two priorities: low and high. The priority is passed to the DLE with the service request.

After token reception, each Master station may always execute **one** high priority message transfer period including retries in the case of an error independently of the token holding time T_{TH} .

Further high or low priority message transfer periods may be executed according to the following rules if T_{TH} is still available (see 20.5.5):

- High or low priority messages may be carried out if the calculated real rotation time (T_{RR}) is less than the current value of the token-rotation-timer before the instant of execution of message sending.
- Once a high or low priority message transfer period is started, it is always completed, including any required retry (retries), even if the token-rotation-timer is less than or equal to the value of T_{RR} during the execution.
- If there is no T_{TH} available (the T_{RR} is greater than the current value of the token-rotation-timer before the instant of execution of message sending) then the token shall be passed to NS immediately.

NOTE 1 The prolongation of the token holding time T_{TH} automatically results in a shortening of transmission time for message transfer periods at the next token receipt.

NOTE 2 For further explanation refer to Annex E and Annex F.

20.3.3 Send or send/request mode

In the send or send/request mode, single message transfer periods are conducted sporadically. The Master station's DL-entity initiates this mode due to a local user's request upon receipt of the token. If there are several requests, this mode of operation may be continued until the maximum allowed token holding time expires.

20.4 Service assumed from the PhL

20.4.1 Asynchronous transmission

20.4.1.1 PhS transmission and reception services

The PhL data service includes two service primitives. A request primitive is used to request a service by the DL-entity; an indication primitive is used to indicate a reception to the DL-entity. The names of the respective primitives are as follows:

Ph-ASYN-DATA request

Ph-ASYN-DATA indication

The temporal relationship of the primitives is shown in Figure 47.

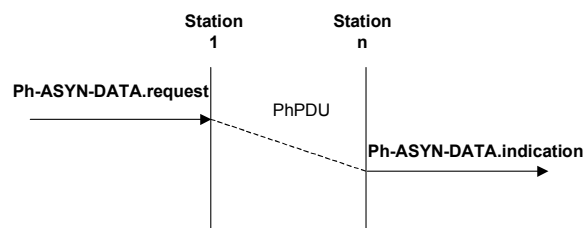


Figure 47 – PhL data service for asynchronous transmission

20.4.1.2 Detailed specification of the service and message exchange

This subclause describes in detail the service primitives and the related parameters in an abstract way. The parameters contain information needed by the PhL-entity.

Parameters of the primitives:

Ph-ASYN-DATA request (DL_symbol)

The parameter DL_symbol shall have one of the following values:

- a) ZERO corresponds to a binary "0"
- b) ONE corresponds to a binary "1"
- c) SILENCE disables the transmitter when no valid DL symbol is to be transmitted

The Ph-ASYN-DATA request primitive is passed from the DL-entity to the PhL-entity to request that the given symbol shall be sent to the fieldbus medium.

The reception of this primitive shall cause the PhL-entity to attempt encoding and transmission of the DL-symbol.

The Ph-ASYN-DATA request is a primitive, which shall only be generated once per DL-symbol period (t_{BIT}). The PhL-entity may confirm this primitive with a locally defined confirmation primitive.

Ph-ASYN-DATA indication (DL_symbol)

The parameter DL_symbol shall have one of the following values:

- ZERO corresponds to a binary "0"
- ONE corresponds to a binary "1"

The Ph-ASYN-DATA indication primitive is passed from the PhL-entity to the DL-entity to indicate that a DL-symbol was received from the fieldbus medium.

The Ph-ASYN-DATA indication is a primitive, which shall only be generated once per received DL-symbol period (t_{BIT}).

20.4.2 Synchronous transmission

The same interface between the PhL-entity and DL is used as described in 6.4.

20.5 Operational elements

20.5.1 Overview

The following times T are measured in bits. A time t in seconds (s) shall therefore be divided by the bit time t_{BIT} .

20.5.2 Bit time t_{BIT}

The bit time t_{BIT} is the time, which elapses during the transmission of one bit. It is equivalent to the reciprocal value of the data rate:

$$t_{BIT} = \frac{1}{\text{data rate}} [\text{s bit}^{-1}] \quad (25)$$

20.5.3 Asynchronous transmission

20.5.3.1 Synchronization time (T_{SYN})

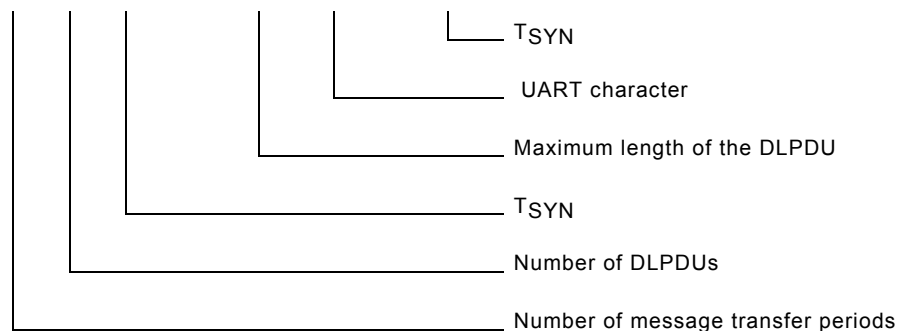
The synchronization time T_{SYN} is the minimum time interval during which each station shall receive idle state (idle = binary "1") from the transmission medium before it may accept the beginning of a send/request DLPDU or token DLPDU. The value of the synchronization time shall be set with:

$$T_{SYN} = 33 \text{ bit} \quad (26)$$

20.5.3.2 Synchronization interval time (T_{SYNI})

The synchronization interval time T_{SYNI} is the maximum allowed time interval between two consecutive synchronization times (T_{SYN}), in order to detect "permanent transmitters". The value of the synchronization interval time shall be set with:

$$T_{SYNI} = 2 \times (2 \times (33 \text{ bit} + 255 \times 11 \text{ bit})) + 33 \text{ bit} = 11\,385 \text{ bit} \quad (27)$$



This value regards two complete message transfer periods, each of which consists of two DLPDUs of maximum length and the related synchronization times. A transmission disturbance is permitted in one of those synchronization times.

20.5.3.3 Station delay time (T_{SDx})

The station delay time T_{SDx} is the period of time which may elapse between the transmission or receipt of a DLPDU's last bit until the transmission or receipt of a following DLPDU's first bit (with respect to the transmission medium, that is, including line receiver and transmitter). The following three station delays are defined:

- Station Delay of Initiator (station transmitting request or token DLPDU):

T_{SDI}

b) Minimum Station Delay of Responders (stations which acknowledge or respond):

min T_{SDR}

c) Maximum Station Delay of Responders:

max T_{SDR}

20.5.3.4 Quiet time (T_{QUI})

When transposing the NRZ signals into a different signal coding, the transmitter fall time after switching off the transmitter (at the initiator) shall be taken into account if it is greater than T_{SDR}.

During this quiet time T_{QUI}, transmission and receipt of DLPDUs shall be disabled. This shall also be taken into account when using self-controlled repeaters, whose switching time shall be taken into consideration. The implementation shall ensure, that the following condition is fulfilled:

$$T_{QUI} < \min T_{SDR} \quad (28)$$

In order to fulfil this condition, it may be necessary to prolong min T_{SDR}.

20.5.3.5 Ready time (T_{RDY})

The ready time T_{RDY} is the time within which a Master station shall be ready to receive an acknowledgement or response after transmitting a request. The implementation shall ensure, that the following condition is fulfilled:

$$T_{RDY} < \min T_{SDR} \quad (29)$$

In order to fulfil this condition it may be necessary to prolong min T_{SDR}.

When transposing NRZ signals into a different signal coding, the quiet time shall also be taken into consideration when switching off the transmitter. The receiver shall not be enabled before this time:

$$T_{QUI} < T_{RDY} \quad (30)$$

In order to fulfil this condition, it may be necessary to prolong T_{RDY} and thus min T_{SDR} accordingly.

20.5.3.6 Safety margin (T_{SM})

The following time interval is specified as safety margin T_{SM}:

$$T_{SM} = 2 \text{ bit} + 2 \times T_{SET} + T_{QUI} \quad (31)$$

T_{SET} is the set-up time, which expires from the occurrence of an event (for example, an interrupt on the last bit of a sent DLPDU or when synchronization time expires) until the necessary reaction is performed (for example, to start synchronization time or to enable the receiver).

20.5.3.7 Idle time (T_{IDx})

The idle time T_{IDx} is the time which expires at the initiator either after receipt of a DLPDU's last bit (measured at the line receiver) as idle = binary "1" **on the transmission medium**, until a new DLPDU's first bit is transmitted on the medium (including line transmitter) or between transmitting the last bit of a DLPDU which is not to be acknowledged and transmitting the first bit of the next DLPDU. The idle time shall be at least the synchronization time plus the safety margin T_{SM} (see Figure 48, Figure 49 and Figure 50 case a)).

$$T_{IDx} \geq T_{SYN} + T_{SM} \quad (32)$$

At high data rates (see Figure 48, case b) and c) and Figure 49 case b)) the synchronization time is very short, hence the station delays become significant and shall be taken into consideration.

Two idle times are distinguished. After an acknowledgement, response or token DLPDU the idle time shall be calculated as follows:

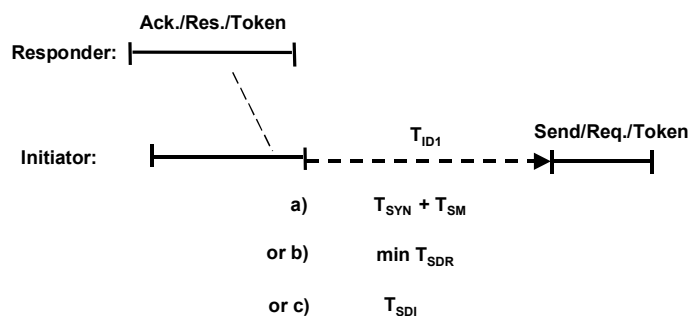


Figure 48 – Idle time T_{ID1}

$$T_{ID1} = \max ((T_{SYN} + T_{SM}), (\min T_{SDR}), T_{SDI}) \quad (33)$$

Care shall be taken that the time to update the LMS is not greater than T_{ID1} . This can be accomplished by prolonging T_{SET} or $\min T_{SDR}$. If the necessary prolongation cannot be reached with the range of value of T_{SET} , then the $\min T_{SDR}$ shall be made longer.

After a send DLPDU which is **not** to be acknowledged (SDN or CS), the idle time shall be calculated as shown in Figure 49 and equation 34.

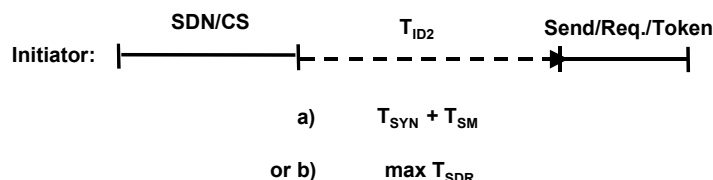
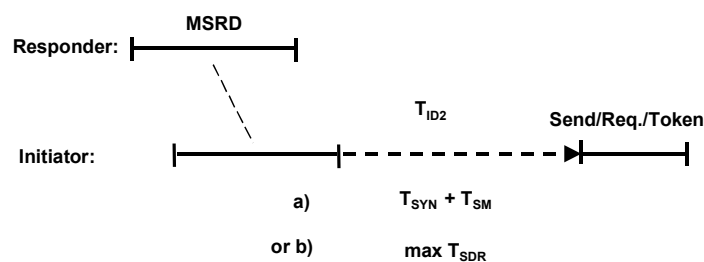


Figure 49 – Idle time T_{ID2} (SDN, CS)

$$T_{ID2} = \max ((T_{SYN} + T_{SM}), (\max T_{SDR})) \quad (34)$$

Also, after a response DLPDU (MSRD) the idle time shall be calculated as shown in Figure 50 and equation 34.

Figure 50 – Idle time T_{ID2} (MSRD)

20.5.3.8 Transmission delay time (T_{TD})

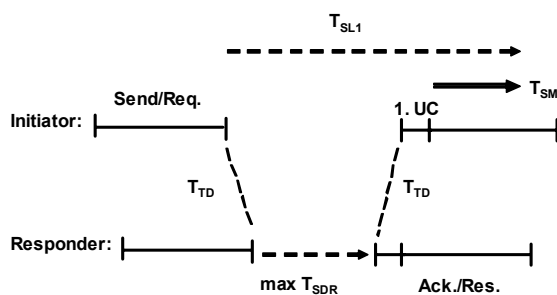
The transmission delay time T_{TD} is the maximum time, which elapses on the transmission medium between transmitter and receiver when a DLPDU is transmitted. When computing it, delay times of repeaters shall be considered, if necessary.

EXAMPLE for computing the transmission delay time: given a line length of 200 m without repeaters, t_{TD} is approximately 1 μ s and thus at 500 kbit/s

$$T_{TD} = (1\mu\text{s} \times 500 \text{ kbit/s}) = 0,5 \text{ bit.}$$

20.5.3.9 Slot time (T_{SL})

The slot time T_{SL} is the maximum time the initiator shall wait for the complete receipt of the first character (see 21.1.1, UART character, 1 UC = 11 bits) of the immediate acknowledgement or response DLPDU, after transmitting the last bit of a send/request DLPDU (including the line transmitter) (see Figure 51). Furthermore, T_{SL} is the maximum time the initiator waits for the token receiver's first UART character (1. UC) after transmitting a token DLPDU (see Figure 52). Theoretically, two slot times are distinguished. After a send/request DLPDU the slot time shall be calculated as follows:

Figure 51 – Slot time T_{SL1}

$$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + 11 \text{ bit} + T_{SM} \quad (35)$$

After a token DLPDU the slot time shall be calculated as follows:

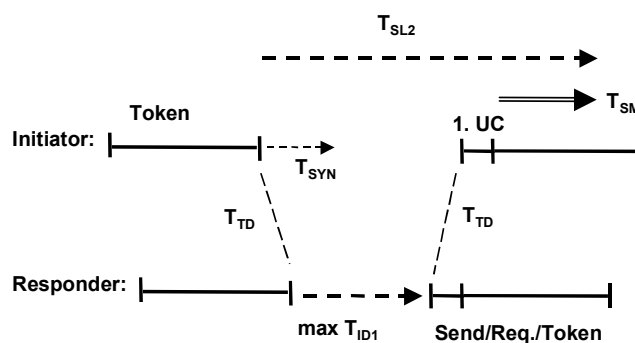


Figure 52 – Slot time T_{SL2}

$$T_{SL2} = 2 \times T_{TD} + \max T_{ID1} + 11 \text{ bit} + T_{SM} \quad (36)$$

In order to simplify the realization, only **one** slot time, the longer one, may be used in the system. This does not influence the system reaction time negatively, as the slot time is a pure monitoring time.

$$T_{SL} = \max (T_{SL1}, T_{SL2}) \quad (37)$$

20.5.3.10 Time-out time (T_{TO})

The time-out time T_{TO} serves to monitor the Master and Slave stations' bus activity and idle time. Monitoring shall start either immediately after PON, in the "Listen_Token" or "Passive_Idle" state, or later after receiving the last bit of a DLPDU. It shall end after receiving the first bit of the following DLPDU. If the phase of no bus activity times out, the bus shall be regarded as inactive. (This is an error case, for example, due to a lost token DLPDU.) The time-out interval shall be set to:

$$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL} \quad (38)$$

For Master stations: n = station address (0 to 126)

For Slave stations: n = 130, independent of the station's address

The first term ensures that there is sufficient difference to the maximum permissible idle time between two DLPDUs. The second term ensures that not all Master stations claim the token at the same moment after an error has occurred.

20.5.3.11 GAP update time (T_{GUD})

The GAP update time T_{GUD} serves for initializing GAP maintenance by the Master station. After the first generation of the GAPL, update of the GAP image is cyclically initialized after every interval T_{GUD} . This initialization takes place at the next possible token receipt, if there is still token holding time available after the regular DLPDU transfer periods, or during later token holding phases. The GAP update time is a multiple of the target rotation time T_{TR} and shall be set to:

$$T_{GUD} = G \times T_{TR} \quad (\text{where } 1 \leq G \leq 100) \quad (39)$$

T_{TR} is the target rotation time (see 20.3.2.6).

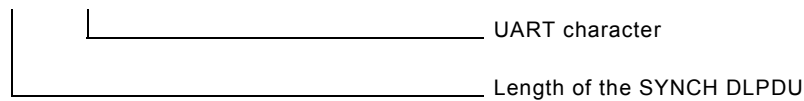
20.5.3.12 Isochronous cycle time (T_{CT})

The isochronous cycle time T_{CT} serves to supervise the isochronous cycle. The first cycle starts by sending the SYNCH DLPDU after reception of the token DLPDU.

20.5.3.13 IsoM synchronization DLPDU time (T_{SYNCH})

The isochronous mode synchronization DLPDU time describes the time, which is needed to send the SYNCH DLPDU at the start of a new IsoM cycle. The value of the IsoM synchronization message time shall be set with:

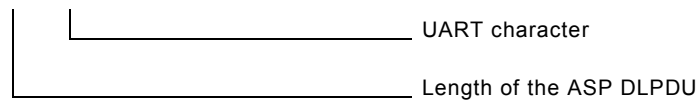
$$T_{SYNCH} = T_{ID2} + (13 \times 11 \text{ bit}) \quad (40)$$



20.5.3.14 Active spare time DLPDU time (T_{ASM})

The active spare time DLPDU time describes the duration that is needed to send an active spare time (ASP) DLPDU.

$$T_{ASM} = T_{ID1} + (6 \times 11 \text{ bit}) \quad (41)$$



20.5.3.15 Real isochronous cycle time (T_{RCT})

The real isochronous cycle time is the read value of the isochronous-cycle-timer.

20.5.3.16 Spare time (T_{RES})

Within the isochronous cycle the token is passed by the IsoM Master after the transmission of all high priority messages and the number of configured low priority messages. After receiving the next token addressed to the IsoM Master, the station read the isochronous cycle timer (T_{RCT}) in order to calculate the remaining time before the next SYNCH DLPDU is to be sent.

$$T_{RES} = T_{CT} - T_{RCT} \quad (42)$$

Within the spare time, at least one active spare time DLPDU shall be sent. For further active spare time, DLPDUs shall be valid.

$$T_{RES} > T_{ASM} + T_{ID1} \quad (43)$$

20.5.3.17 Passive spare time (T_{PSP})

The passive spare time denotes the part of isochronous cycle where the IsoM Master shows no bus activities. It shall be less than the minimum time-out time T_{TO} under consideration of possible delays in the stations and during transmission. The passive spare time is defined as follow:

$$T_{PSP} < 6 \times T_{SL} - T_{SM} - 2 \times T_{TD} \quad (44)$$

$$T_{PSP} > T_{ID1} + T_{ASM} + \max T_{SDR} \quad (45)$$

20.5.3.18 Time shift (T_{SH})

The time shift is the difference between the measured cycle time and the calculated cycle time when the passive-spare-timer expires.

$$T_{SH} = T_{RCT} - T_{CT} \quad (46)$$

20.5.3.19 Send delay time (T_{SD})

The send delay time is the time that elapses between the receiving of a DL-CS-TIME-EVENT.request primitive at the DLE of the time master and the DLPDU's last bit of a resulting TE DLPDU (see 22.7) transmitted by the time master.

20.5.3.20 Receive delay time (T_{RD})

The receive delay time is the time that elapses at a time receiver between the receiving of the last bit of a TE DLPDU (see 22.7) and the receiving of a DLPDU's last bit of a CV DLPDU (see 22.8).

20.5.3.21 Clock synchronization interval time (T_{CSI})

The clock synchronization interval time is used to monitor the synchronization sequences within the DLE.

20.5.4 Synchronous transmission

20.5.4.1 Synchronization time (T_{SYN})

The synchronization time is the minimum time interval during which each station shall receive no activity from the transmission medium before it may accept the beginning of a send/request DLPDU or token DLPDU.

The synchronization time shall correspond to the post-transmission gap time (T_{PTG}) that is defined in IEC 61158-2:2003¹², 9.2.8. Its value shall be set to at least 4 bit and may be increased by the DLMS-user up to 32 bit.

$$T_{SYN} = T_{PTG} = T_{QUI} \quad (47)$$

$$T_{SYN} = 4 \text{ to } 32 \text{ bit} \quad (48)$$

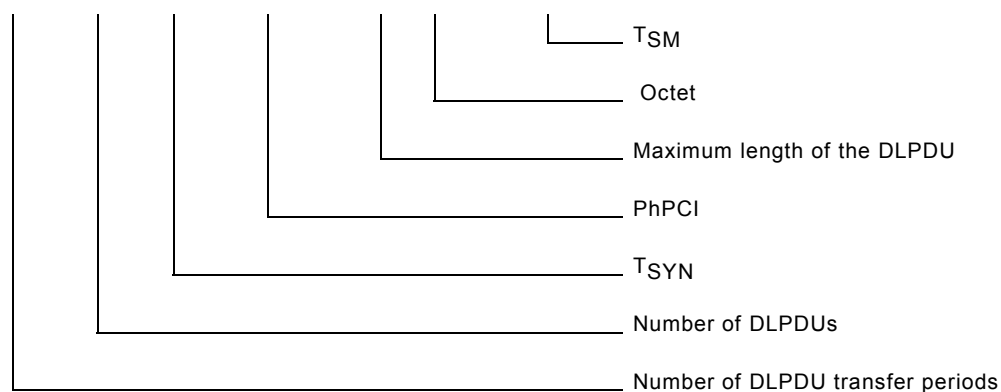
20.5.4.2 Synchronization interval time (T_{SYNI})

The synchronization interval time T_{SYNI} is the maximum allowed time interval between two consecutive synchronization times (T_{SYN}), in order to detect "permanent transmitters".

The value of T_{SYNI} shall be set as follows:

¹² Expected date of concurrent publication.

$$T_{SYNI} = 2 \times (2 \times (32 \text{ bit} + 80 \text{ bit} + 255 \times 8 \text{ bit})) + 64 \text{ bit} = 8672 \text{ bit} \quad (49)$$



This value regards two complete message sequences, each of which consists of two DLPDUs of maximum length and the associated maximum PhL Protocol Control Information (PhPCI: Preamble, Start Delimiter, End Delimiter) and the maximum synchronization time (post-transmission gap). A transmission disturbance is permitted in one of those synchronization times.

20.5.4.3 Station delay time (T_{SDx})

The station delay time T_{SDx} is the period of time, which may elapse between the transmission or receipt of a DLPDU's last octet until the transmission or receipt of a following DLPDU's first octet (with respect to the transmission medium, that is, including line receiver and transmitter). The following three station delays are defined:

- a) Station Delay of Initiator (station transmitting request or token DLPDU)

T_{SDI}

- b) Minimum Station Delay of Responders (station that acknowledges or responds)

$\min T_{SDR}$

- c) Maximum Station Delay of Responders

$\max T_{SDR}$

20.5.4.4 Quiet time (T_{QUI})

The transmitter fall time or repeater switch time corresponds to the post-transmission gap time (T_{SYN}). The following shall apply:

$$T_{QUI} = T_{PTG} = T_{SYN} \quad (50)$$

20.5.4.5 Ready time (T_{RDY})

The ready time T_{RDY} is the time within which a Master station shall be ready to receive an acknowledgement or response after transmitting a request. The implementation shall ensure, that the following condition is fulfilled:

$$T_{RDY} < \min T_{SDR} \quad (51)$$

In order to fulfil this condition it may be necessary to prolong $\min T_{SDR}$.

During the quiet time T_{QUI} , transmission and receipt of DLPDUs shall be disabled. The implementation shall ensure, that the following condition is fulfilled:

$$T_{QUI} = T_{PTG} = T_{SYN} < T_{RDY} \quad (52)$$

In order to fulfil this condition the min T_{SDR} shall be increased according to equation (51) if necessary.

20.5.4.6 Safety margin (T_{SM})

The safety margin T_{SM} is defined as the time interval:

$$T_{SM} = 2 \text{ bit} + 2 \times T_{SET} \quad (53)$$

T_{SET} is the set-up time, which expires from the occurrence of an event (for example, an interrupt on the last octet of a sent DLPDU or on synchronization time expiration) until the execution of the necessary reaction.

20.5.4.7 Idle time (T_{IDx})

The idle time T_{IDx} is the time which expires at the initiator after a Ph-DATA indication primitive until sending of a new DLPDU with Ph-DATA request primitive or after passing a Ph-DATA request primitive with a Ph-DATA confirm primitive to transmit a DLPDU which is not to be acknowledged until passing a new Ph-DATA request primitive for transmitting the next DLPDU. The idle time shall be at least the synchronization time plus the safety margin T_{SM} .

$$T_{IDx} \geq T_{SYN} + T_{SM} \quad (54)$$

Two idle times are distinguished (see description of idle time in 20.5.3.7, Figure 48, Figure 49 and Figure 50). After an acknowledgement, response or token DLPDU the idle time shall be calculated as follows:

$$T_{ID1} = \max((T_{SYN} + T_{SM}), (\min T_{SDR}), T_{SDI}) \quad (55)$$

Care shall be taken that the time to update the LMS is not greater than T_{ID1} . This can be accomplished by prolonging T_{SET} or min T_{SDR} . If the necessary prolongation cannot be reached with the range of value of T_{SET} , than the min T_{SDR} shall be made longer.

After a send DLPDU, which is not to be acknowledged (SDN, CS), the idle time shall be calculated as follows:

$$T_{ID2} = \max((T_{SYN} + T_{SM}), (\max T_{SDR})) \quad (56)$$

20.5.4.8 Transmission delay time (T_{TD})

The transmission delay time T_{TD} is the maximum time, which elapses on the transmission medium between transmitter and receiver when a DLPDU is transmitted. When computing it, delay times of repeaters shall be considered if necessary. It shall also conform to any restrictions placed on it in IEC 61158-2, where it is called "propagation delay".

20.5.4.9 Slot time (T_{SL})

The slot time T_{SL} is the maximum time the initiator shall wait after passing a Ph-DATA request primitive for transmitting a request DLPDU from the Ph-DATA confirm primitive until receiving the first Ph-DATA indication primitive as an indication of receiving the immediate acknowledgement or response (see Figure 53). Furthermore, T_{SL} is the maximum time the initiator shall wait for a Ph-DATA indication primitive after the token DLPDU as reaction to receiving the first DLPDU octet from the token receiver. Theoretically, two slot times are distinguished (see Figure 54). After a send/request DLPDU the following slot time shall be calculated:

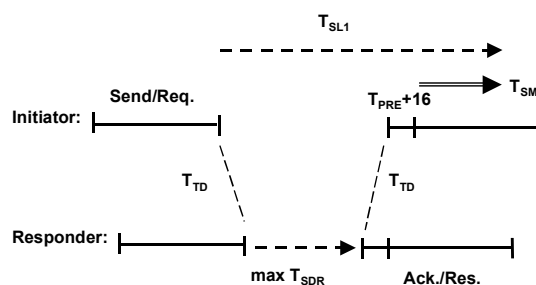


Figure 53 – Slot time T_{SL1}

$$T_{SL1} = 2 \times T_{TD} + \max T_{SDR} + T_{PRE} + 16 \text{ bit} + T_{SM} \quad (57)$$

After a token DLPDU the following slot time shall be calculated:

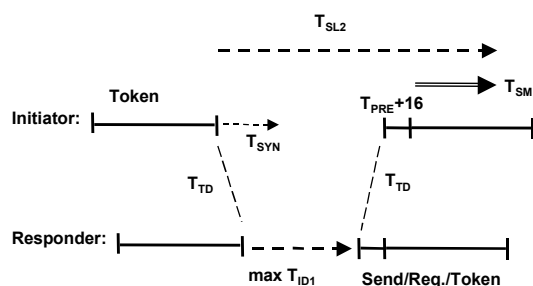


Figure 54 – Slot time T_{SL2}

$$T_{SL2} = 2 \times T_{TD} + \max T_{ID1} + T_{PRE} + 16 \text{ bit} + T_{SM} \quad (58)$$

where T_{PRE} is Preamble period (see IEC 61158-2)

In order to simplify the realization, only the longer slot time may be used in the system. This does not influence the system reaction time negatively, as the slot time is merely a monitoring time.

$$T_{SL} = \max (T_{SL1}, T_{SL2}) \quad (59)$$

20.5.4.10 Time-out time (T_{TO})

The time-out time T_{TO} serves to monitor the Master and Slave stations' bus activity and idle time. Monitoring shall start either immediately after PON, in the "Listen-Token" or "Passive_Idle" state, or later after the reception of a Ph-DATA indication primitive. It shall end upon receipt of a Ph-DATA indication primitive for reception of the first octet of a following DLPDU. If the phase of no bus activity times out, the bus shall be regarded as inactive. (This is an error case, for example, due to a lost token DLPDU.). The time-out time shall be set to the following value:

$$T_{TO} = 6 \times T_{SL} + 2 \times n \times T_{SL} \quad (60)$$

For Master stations: n = station address (0 to 126)

For Slave stations: n = 130, independent of their station address

20.5.4.11 GAP update time (T_{GUD})

20.5.3.11 shall apply.

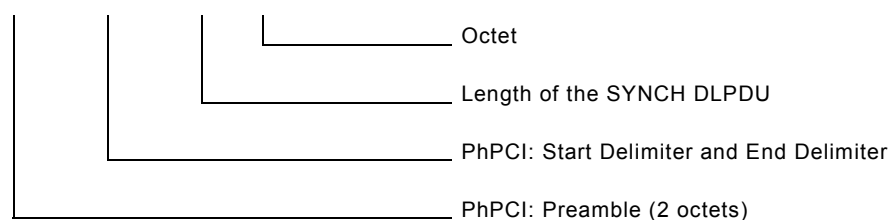
20.5.4.12 Isochronous cycle time (T_{CT})

20.5.3.12 shall apply.

20.5.4.13 IsoM synchronization DLPDU time (T_{SYNCH})

The isochronous mode synchronization DLPDU time describes the time, which is needed in order to send the SYNCH DLPDU at the start of a new IsoM cycle. The value of the IsoM synchronization message time shall be set with:

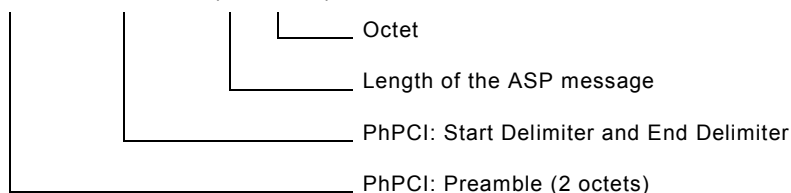
$$T_{SYNCH} = 16 \text{ bit} + 16 \text{ bit} + 13 \times 8 \text{ bit} = 136 \text{ bit} \quad (61)$$



20.5.4.14 Active spare time DLPDU time (T_{ASM})

The active spare time DLPDU time describes the duration that is needed to send an active spare time (ASP) DLPDU.

$$T_{ASM} = T_{PTG} + 16 \text{ bit} + 16 \text{ bit} + (6 \times 8 \text{ bit}) \quad (62)$$



20.5.4.15 Real isochronous cycle time (T_{RCT})

20.5.3.15 shall apply.

20.5.4.16 Spare time (T_{RES})

20.5.3.16 shall apply.

20.5.4.17 Passive spare time (T_{PSP})

The passive spare time denotes the part of isochronous cycle where the IsoM Master shows no bus activities. It shall be less than the minimum time-out time T_{TO} under consideration of possible delays in the stations and during transmission. The passive spare time is defined as follow:

$$T_{PSP} < 6 \times T_{SL} - T_{SM} - 2 \times T_{TD} \quad (63)$$

$$T_{PSP} > T_{PTG} + T_{ASM} + \max T_{SDR} \quad (64)$$

20.5.4.18 Time shift (T_{SH})

20.5.3.18 shall apply.

20.5.4.19 Send delay time (T_{SD})

The send delay time is the time that elapses between a passed DL-CS-TIME-EVENT request primitive to the DLE of the time master and the DLPDU's last octet of a resulting TE DLPDU (see 22.7) transmitted by the time master.

20.5.4.20 Receive delay time (T_{RD})

The receive delay time is the time that elapses by a time receiver between the receiving of a DLPDU's last octet of a TE DLPDU (see 22.7) and the last octet of a CV DLPDU (see 22.8).

20.5.4.21 Clock synchronization interval time (T_{CSI})

20.5.3.19 shall apply.

20.5.5 Timers and counters**20.5.5.1 Asynchronous transmission****20.5.5.1.1 Timers**

In order to measure the token rotation time and to realize the supervisory times the following timers shall be implemented:

token-rotation-timer, idle-timer, slot-timer, time-out-timer, syn-interval-timer, GAP-update-timer, isochronous-cycle-timer, passive-spare-timer, send-delay-timer and receive-delay-timer.

token-rotation-timer: When a Master station receives the token, this timer is loaded with the target rotation time T_{TR} and decremented each bit time. When the station again receives the token, the timer value, the remaining time or token holding time T_{TH}, is read and the timer reloaded with T_{TR}. The real rotation time T_{RR} results from the difference T_{TR} - T_{TH}.

The token-rotation-timer can be read out at every moment, representing always the value of the actual token rotation. Low priority message transfer periods may be processed if at the instant of processing the real token rotation time is less than the value of the actual token rotation.

idle-timer: This timer monitors the idle state (binary "1"), the synchronization time, immediately on the bus line. The synchronization time preceding each request is necessary for unambiguous receiver synchronization. The idle-timer of Slave stations and Master stations "without token" is loaded with T_{SYN} after the transmission or receipt of a DLPDU's last bit and then decremented each bit time. The receiver shall be enabled immediately after the timer has expired. The timer of a Master station "with token" is loaded according to the data transmission service with T_{ID1} or T_{ID2} (see 20.5.3). A new request or token DLPDU may only be transmitted after expiration of the timer. When the signalling level is binary "0", the timer is always reloaded.

slot-timer: This timer in a Master station monitors after a request or token pass whether the receiving station responds or becomes active within the predefined time T_{SL} , the slot time. After transmission of a DLPDU's last bit this timer is loaded with T_{SL} and decremented each bit time as soon as the receiver is enabled. If the timer expires before a DLPDU's first bit is received, an error has occurred. Then a retry or a new message transfer period shall be initiated.

time-out-timer: This timer monitors bus activity in Master and Slave stations. After the transmission or receipt of a DLPDU's last bit the timer is loaded with a multiple of the slot time (see 20.5.3) and decremented each bit time as long as no new DLPDU is received. If the timer expires, a fatal error has occurred, which for the Master station causes a (re)initialization. The DLMS-user of the Slave and Master station receives a time-out notification.

syn-interval-timer: Master and Slave stations use this timer to monitor the transmission medium whether a receiver synchronizing (T_{SYN} , idle state, idle = binary "1") occurs within T_{SYNI} . Each time the receiver is synchronized, the timer is loaded with T_{SYNI} (see 20.5.3). From the beginning of a DLPDU (first start bit) the timer is decremented each bit time as long as no new T_{SYN} is detected. If the timer expires, an error has occurred on the transmission medium, for example, stuck at "0" or permanent "0" / "1" edges. The DLMS-user is notified accordingly.

GAP-update-timer: Only Master stations need this timer. Its expiration indicates the moment for GAP maintenance. After a complete GAP check, which may last several token rotations (segmented; see 20.3.2.4), the timer is loaded with a multiple (G) of the target rotation time T_{TR} (see 20.3.2.6).

NOTE For ease of implementation, this timer can be implemented as a counter, decremented at each token receipt.

isochronous-cycle-timer: The timer supervises the isochronous cycle. It is loaded with value equal to 0 at the starting point of the isochronous mode (after receiving the first token) and increments every bit time. The timer is readable at any time. The timer is started either by loading with the value equal to 0 at the first beginning of the isochronous cycle or after expiration of the passive-spare-timer. In cases the timer is started too late, it is loaded with the value of the time difference between the real isochronous cycle time of the last cycle and the isochronous cycle time.

passive-spare-timer: This timer monitors the idle time before sending a SYNCH message. It shall be set according to the time difference between T_{CT} and the current value of the isochronous-cycle-timer T_{RCT} at the end of the last ASP message if the difference is greater than 0. The transmission of the last bit of the last ASP message in an isochronous cycle starts this timer. In the isochronous mode after expiration of the passive-spare-timer, a SYNCH message shall be sent and a synch notification event shall be sent to the DLMS-user.

send-delay-timer: When the time master receives a DL-CS-TIME-EVENT request primitive from the local DLS-user, this timer is started with the value $= 2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after confirmation of the transmission of the last portion of a TE DLPDU (see 22.7). The value of send delay time is calculated as the difference between the start value ($2 \times T_{CSI}$) and the read send-delay-timer value. The calculated send delay time is passed to the local DLS-user. In case the timer has expired, the DLS-user is notified of a clock synchronization sequence violation.

receive-delay-timer: When the time receiver receives the DLPDU's last bit of a TE DLPDU (see 22.7), this timer is started with the value $= 2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after the receiving the last bit of a CV DLPDU (see 22.8). The value of the receive delay time is calculated as difference between the start value ($2 \times T_{CSI}$) and the read receive-delay-timer value. The calculated receive delay time is passed to the local DLS-user. In case the timer expired a clock synchronization sequence violation is notified to the DLS-user.

When a Master station enters the "Listen_Token" state, the idle-timer is loaded with T_{SYN} , the time-out-timer with T_{TO} , the syn-interval-timer with T_{SYNI} and the other timers are cleared. When a Slave station enters the "Passive_Idle" state, the time-out-timer is loaded with T_{TO} and the syn-interval-timer with T_{SYNI} .

20.5.5.1.2 Counters

For installation and maintenance the following pairs of counters (DL-variables) may be used:

For Master stations:

- counter for transmitted DLPDUs (DLPDU_sent_count), except for the SDN and Request DL Status with Reply services
- counter for transmitted DLPDU retries (Retry_count)

Additional counters may be provided for each individual remote station:

- counter for transmitted DLPDUs (DLPDU_sent_count_sr), except for the SDN and Request DL Status with Reply services and for token DLPDUs
- counter for transmitted DLPDUs with no response or erroneous response (Error_count), except for the SDN and Request DL Status with Reply services and for token DLPDUs

For Slave and Master stations:

- counter for received valid start delimiters (SD_count)
- counter for received invalid start delimiters (SD_error_count)

When a station enters the "Listen_Token" or "Passive_Idle" state, the counters are cleared and enabled. If a counter reaches its maximum, counting of this counter as well as of the related comparative counter is stopped. When clearing a counter the related comparative counter is also cleared and they are enabled again. The DLMS-user may access these counters using the Set/Read Value services.

20.5.5.2 Synchronous transmission

20.5.5.2.1 Timers

As explained in 20.5.5.1.1, the following timers shall be implemented to measure the token rotation time and to realize the monitor timers:

token-rotation-timer, idle-timer, slot-timer, time-out-timer, syn-interval-timer, GAP-update-timer, isochronous-cycle-timer, passive-spare-timer, send-delay-timer and receive-delay-timer.

token-rotation-timer: The functionality of this timer is as defined in 20.5.5.1.1.

idle-timer: This timer monitors the idle state $T_{PTG} = T_{SYN} = T_{QUI}$ on the bus line. The idle-timer in the Master station with the token is loaded with T_{ID1} or T_{ID2} depending on the data transmission service (see 20.5.4). The timer is decremented every bit time, when after a Ph-DATA request primitive (PhICI specifying END-OF-DATA-AND-ACTIVITY) either the Ph-DATA confirm primitive at the transmitter or the Ph-DATA indication primitive (PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) at the receiver is transferred. A new request or a new token DLPDU may be transmitted only after expiration of the timer.

slot-timer: After a request from or a token transfer by a Master station, this timer of the Master station monitors whether the receiving station responds or becomes active within the defined slot time T_{SL} . The timer is initialized with T_{SL} and is decremented every bit time after each transmission of a DLPDU. This DLPDU transmission is indicated by a Ph-DATA confirm primitive after transfer of a Ph-DATA request primitive (PhICI specifying END-OF-DATA-AND-ACTIVITY). If the timer expires before a DLPDU has been received, as indicated by a Ph-DATA indication primitive (PhICI specifying START-OF-ACTIVITY), an error has occurred. As a result of that, a retry or a new message transfer period is initiated.

time-out-timer: This timer monitors bus activity in Master and Slave stations. After transfer of the last Ph-DATA request primitive with PhICI specifying END-OF-DATA-AND-ACTIVITY and return of the corresponding Ph-DATA confirm primitive, or after receiving a Ph-DATA indication primitive with PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY, the timer is loaded with a multiple of the slot time (see 20.5.4) and is decremented every bit time as long as no Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY has been received. If the timer expires, a fatal error has occurred, which for the Master station causes a (re)initialization. The DLMS-user of the Slave or Master station respectively receives a time-out notification.

syn-interval-timer: Master and Slave stations use this timer to monitor the transmission medium for "permanent transmitters". After every Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY, the timer is loaded with the value T_{SYNI} (see 20.5.4) and decremented every bit time, as long as no Ph-DATA indication primitive with PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY has been received. If the timer expires, an error of the transmission medium has occurred. The DLMS-user receives a corresponding notification.

GAP-update-timer: This timer operates in the same way as described in 20.5.5.1.1.

isochronous-cycle-timer: The functionality of this timer is as defined in 20.5.5.1.1.

passive-spare-timer: This timer monitors the idle time before sending a SYNCH message. It shall be set according to the time difference between T_{CT} and the current value of the isochronous-cycle-timer T_{RCT} at the end of the last ASP message if the difference is greater than 0. The transmission of the last bit of the last ASP message in an isochronous cycle starts this timer. In the isochronous mode after expiration of the passive-spare-timer, a SYNCH message shall be sent and a synch notification event shall be sent to the DLMS-user.

send-delay-timer: When the time master receives a DL-CS-TIME-EVENT request primitive from the local DLS-user, this timer is started with the value $= 2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after confirmation of the transmission of the last portion of a TE DLPDU (see 22.7). The value of send delay time is calculated as the difference between the start value ($2 \times T_{CSI}$) and the read send-delay-timer value. The calculated send delay time is passed to the local DLS-user. In case the timer has expired, the DLS-user is notified of a clock synchronization sequence violation.

receive-delay-timer: When the time receiver receives the DLPDU's last bit of a TE DLPDU (see 22.7) indicated by a Ph-DATA indication primitive (PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) has been received, this timer is started with the value $= 2 \times T_{CSI}$ and decremented each bit time. The timer is stopped after the receiving of the DLPDU's last bit of a CV DLPDU (see 22.8) indicated by a Ph-DATA indication primitive (PhICI specifying either END-OF-ACTIVITY or END-OF-DATA-AND-ACTIVITY) has been received. The value of the receive delay time is calculated as difference between the start value ($2 \times T_{CSI}$) and the read receive-delay-timer value. The calculated receive delay time is passed to the local DLS-user. In case the timer expired a clock synchronization sequence violation is notified to the DLS-user.

20.5.5.2.2 Counters

The specifications given in 20.5.5.1 shall apply for the optional counters.

20.6 Cycle and system reaction times

20.6.1 Asynchronous transmission

20.6.1.1 Token transfer period

The base load in a system with several Master stations, that is, the busload caused by medium access control (token DLPDUs) and not by regular message transfer periods, is determined by the token period TP. The total base load per token rotation results from na (number of Master stations) token cycles. The transfer period T_{TP} is composed of the token DLPDU time T_{TF} , the transmission delay time T_{TD} and the idle time T_{ID1} . T_{ID1} results from the station delay time T_{SD1} or the synchronization time T_{SYN} respectively. T_{TP} is measured in bits (see Figure 55).

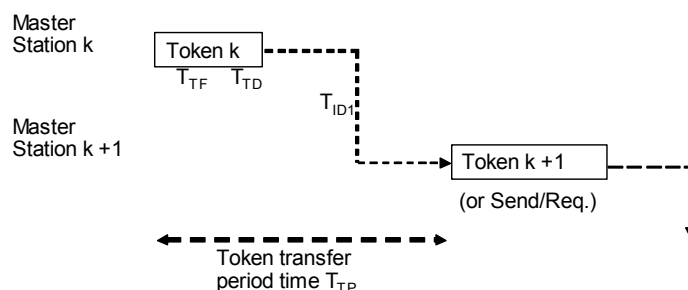


Figure 55 – Token transfer period

$$T_{TP} = T_{TF} + T_{TD} + T_{ID1} \quad (65)$$

The token DLPDU time T_{TF} is determined by the number of UART characters, UC , in the token DLPDU. A UART character always consists of 11 bits (see 21.1.1) and hence the token DLPDU comprises altogether 33 bits. The transmission delay time T_{TD} depends on the line length (about 5 ns/m without repeater) and is mostly substantially less than the other times. The idle time T_{ID1} , which elapses between the token DLPDUs, contains the station delay time T_{SD1} of the token receiver on the one hand, on the other hand the synchronization time $T_{SYN} + T_{SM}$ shall be used, if this sum is larger than T_{SD1} (see 20.5.3). This mainly occurs at low data rates (< 100 kbit/s).

20.6.1.2 Message transfer period

A message transfer period MP consists of the send/request DLPDU and the acknowledgement or response DLPDU. The transfer period is composed of the DLPDU transmission times, the transmission delay times and the station delay times.

The station delay time T_{SDR} elapses between request and acknowledgement or response. This time is needed for decoding the request and assembling the acknowledgement or response DLPDU. It depends on the protocol implementation in the station and is mostly substantially greater than the transmission delay time T_{TD} . The idle time T_{ID} , which elapses between acknowledgement or response and new request, contains also the station delay time (see 20.5.3). However, $T_{SYN} + T_{SM}$ shall be used, if this sum is greater than T_{SDI} .

Figure 56 shows the Message transfer periods.

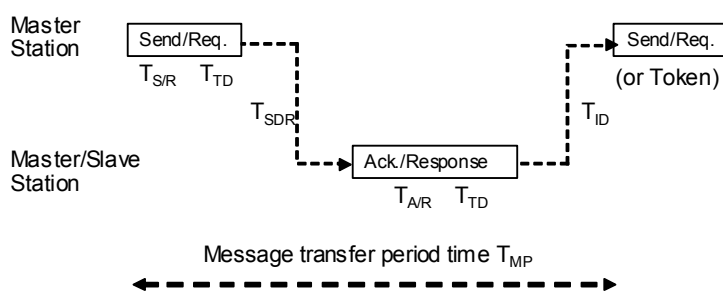


Figure 56 – Message transfer period

$$T_{MP} = T_{S/R} + T_{SDR} + T_{A/R} + T_{ID} + 2 \times T_{TD} \quad (66)$$

If the initiator does not receive a valid response DLPDU, caused by disturbances at a station or at the medium, the initiator will repeat the send/request DLPDU depending on its retry limit and the state of the addressed station. For this retry message transfer period (RMP) another transfer period T_{RMP} applies. T_{RMP} is composed of the send/request DLPDU transmission time and the slot time.

$$T_{RMP} = T_{S/R} + T_{SL} \quad (67)$$

At data rates < 100 kbit/s decoding and evaluation of DLPDUs may partially keep pace with their reception. Station delay times become considerably shorter then.

The DLPDU transmission times ($T_{S/R}$, $T_{A/R}$) are determined by the number of UART characters (UC). Thus, they are calculated as follows:

$$T_{S/R} = a \times 11 \text{ bit, where } a \text{ is the number of UC in a send/request DLPDU}$$

$$T_{A/R} = b \times 11 \text{ bit, where } b \text{ is the number of UC in an ack/response DLPDU}$$

EXAMPLE

$a = 6$ for the request DLPDU; $T_{S/R} = 66$ bit

$b = 59$ for the response DLPDU (50 octet DATA_UNIT); $T_{A/R} = 649$ bit

20.6.1.3 System reaction times

The message rate R_{SYS} in the system equals the possible number of message transfer periods per second:

$$R_{SYS} = 1 / t_{MP}; t_{MP} = T_{MP} \times t_{BIT} \quad (68)$$

The maximum system reaction time T_{SR} in a system with **one** Master station and n Slave stations (Master-Slave system) is calculated from the message transfer period and the number of Slave stations. If message retries are allowed, T_{SR} is calculated as follows:

$$T_{SR} = np \times T_{MP} + mp \times T_{RMP} \quad (69)$$

where

np is the number of Slave stations

mp is the number of message retry transfer periods

T_{RMP} is the message retry transfer period

The maximum system reaction time in a system with **several** Master stations and Slave stations equals the target rotation time:

$$T_{SR} = T_{TR} \quad (\text{see } 20.3.2.6) \quad (70)$$

20.6.1.4 Isochronous cycle time

The isochronous cycle time starts with the transmission of the SYNCH DLPDU by the IsoM Master. The IsoM Master transmits all high priority messages and the configured number of low priority messages. To permit a multi-master environment the token is passed to the next Master station thus leaving other Master stations the necessary time to perform their actions.

$$T_{CT} = T_{SYNCH} + N \times T_{TP} + \sum_{i=1}^N P_i + T_{RES} \quad (71)$$

N = number of master stations within token ring

P_i = DLPDU time by master station i (includes isochronous master's cyclic DLPDUs)

The positive difference between the real cycle time (T_{RCT}) and the calculated cycle time (T_{CT}) represents the spare time T_{RES} . Depending on the amount of time available, a number of active spare time messages (ASM) are sent. At least one ASM message shall be sent. After each sending of the ASM message, the spare time shall be calculated. The transmission of the last ASM is succeeded by the start of the passive-spare-timer if the difference between the real cycle time (T_{RCT}) and the calculated cycle time (T_{CT}) is greater than zero. The sending of a new SYNCH DLPDU starts after expiration of the passive-spare-timer. If the difference between the real cycle time (T_{RCT}) and the calculated cycle time (T_{CT}) is less than or equal to zero then the sending of a new SYNCH DLPDU starts immediately. The transmission of a new SYNCH DLPDU marks the beginning of the next cycle. The start of the next cycle is notified to the DLMS-user with a synch notification.

In parallel, the value of isochronous-cycle-timer is read and compared with the calculated cycle time. If the result is zero or within the allowed time shift ($\max T_{SH}$) no message is sent to the DLMS-user. If the result is not in the range of the allowed time shift a Synch_Delay event with the value of the difference between real and calculated cycle time is sent to the DLMS-user.

20.6.2 Synchronous transmission

20.6.2.1 Token transfer period

Similar to asynchronous transmission, the following shall apply for the token transfer period T_{TP} :

$$T_{TP} = T_{TF} + T_{TD} \quad (72)$$

Due to the DLPDU character (see 21.1.2) and the changed DLPDU format (see 22.4.2), the token DLPDU time T_{TF} is 80 bit with a preamble of 16 bit and a post-transmission gap time of 8 bit.

NOTE Only the transmission speed of 31,25 kbit/s may be selected.

20.6.2.2 Message transfer period

The following shall apply for the message transfer period T_{MP} :

$$T_{MP} = T_{S/R} + T_{SDR} + T_{A/R} + T_{PTG} + 2 \times T_{TD} \quad (73)$$

The specifications for the message transfer period stipulated in 20.6.1.2 shall apply except for the following cases:

The PDU transmission times ($T_{S/R}$, $T_{A/R}$) are determined by the number of PhL octets. Thus, they are calculated as follows:

$T_{S/R} = a$, where a is the number of octets in a request DLPDU (≤ 255)

$T_{A/R} = b$, where b is the number of octets in an ack/response DLPDU (≤ 255)

EXAMPLE

$a = 10$ for the request DLPDU: $T_{S/R} = 80$ bit

(additionally to 20.6.1.2: 2 octet Preamble and 2 octet PhL start/end delimiter)

$b = 63$ for the response DLPDU: $T_{A/R} = 504$ bit

(additionally to 20.6.1.2: 2 octet Preamble and 2 octet PhL start/end delimiter)

20.6.2.3 System reaction times

See 20.6.1.3.

20.6.2.4 Isochronous cycle time

See 20.6.1.4.

21 Type 3: General structure and encoding of DLPDUs, and related elements of procedure

21.1 DLPDU granularity

21.1.1 Asynchronous transmission – UART character

21.1.1.1 General

Each DLPDU shall consist of a number of UART characters (UC). Each UART character is a start-stop character for asynchronous transmission, structured as shown in Figure 57.

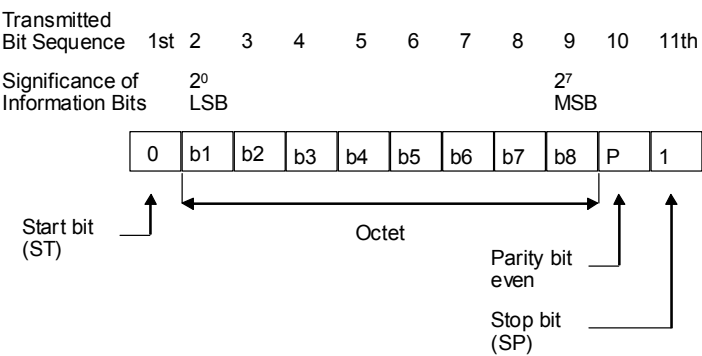


Figure 57 – UART character

The presentation of UART characters is based on ISO/IEC 1177 and ISO/IEC 2022.

21.1.1.2 Transmission Rule

Each UART character shall consist of 11 bits: a start bit (ST) which shall be always binary "0", 8 information bits (I) which may be binary "0" or binary "1", an even parity bit (P) which may be binary "0" or binary "1" and a stop bit (SP) which shall be always binary "1".

21.1.1.3 Bit synchronizing

The receiver's bit synchronizing shall always start with the falling edge of the start bit, that is, at the transition from binary "1" to binary "0". The start bit and all consecutive bits shall be scanned in the middle of the bit time. The start bit shall be binary "0" in the middle of the bit, otherwise the synchronizing shall be regarded as failed and shall be stopped. The synchronizing of the UART character ends with the stop bit being binary "1". If a binary "0" bit is encountered instead of the stop bit, a synchronizing error or UART character error shall be assumed and reported and the next leading edge of a start bit shall be waited for.

A maximum deviation of $\pm 0,3 \%$ of the nominal data rate (bit-period) for transmission and receipt shall not be exceeded for data rates of less than 1500 kbit/s. For data rates of 1500 kbit/s and higher a maximum deviation of $\pm 0,03 \%$ of the nominal data rate (bit-period) shall not be exceeded.

21.1.2 Synchronous transmission

Each octet of the DLPDU shall be structured as shown in Figure 58 for synchronous transmission.

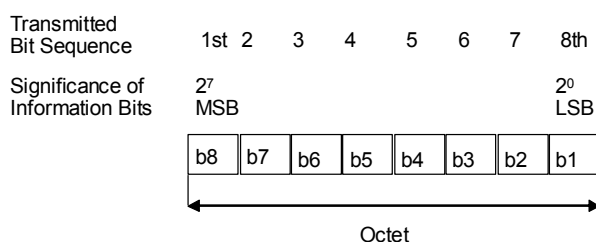


Figure 58 – Octet structure

21.2 Length octet (LE, LEr)

The two length octets of identical value in the DLPDU header of the format for variable data length shall contain the number of information octets in the DLPDU body. These comprise: DA, SA, FC and the DATA_UNIT. The value shall cover the range from 4 to 249, so that a maximum of 246 octets may be transmitted in a DLPDU's DATA_UNIT (see 21.6). A value < 4 is not permitted, as a DLPDU contains at least DA, SA, FC and **one** DATA octet. The longest DLPDU may contain a total of 255 octets. Figure 59 illustrates the Length octet coding.

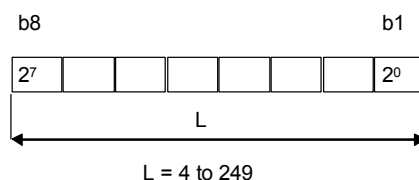


Figure 59 – Length octet coding

21.3 Address octet

21.3.1 Destination and source station address (DA and SA)

The two address octets in the DLPDU header (request, acknowledgement and response DLPDUs) shall contain the destination (DA) and source (SA) station address. The short acknowledgement DLPDU does not contain these two address octets. Figure 60 illustrates the Address octet coding.



Figure 60 – Address octet coding

Address 127 (b1 to b7 = 1) is reserved as global address for broadcast and multicast messages (DLPDU to all stations or a group of stations selected by means of a DL-service-access-point; it may only occur in SDN and CS, and the response DLPDU of MSRD).

Thus, 127 station addresses (0 to 126) are available for Slave and Master stations, of which preferably no more than 32 may be occupied by Master stations. For non time-critical applications, there may be up to 127 Master stations. As at least one Master station is required, a maximum of 126 addresses for Slave stations is allowed.

Except for MSRD, the send/request DLPDU's address octets shall be sent back mirrored in the acknowledgement or response DLPDU. That is, SA of the acknowledgement or response DLPDU shall contain the destination station address and DA shall contain the source station address of the send/request DLPDU. For MSRD the DA of the response DLPDU shall contain the address 127 and SA of the response DLPDU shall contain the destination station address of the send/request DLPDU.

21.3.2 Address Extension (EXT)

Address extensions apply only to DLPDUs with DATA_UNIT. The EXT bit (extension) shall indicate a destination and/or source address extension (DAE, SAE) (see Figure 61), which shall immediately follow the FC octet in the DATA_UNIT. It may be distinguished between access address (DL-service-access-point, DLSAP, see 21.3.4) and region/DL-segment address. Both address types may also occur simultaneously, as each address extension contains an EXT bit again (see bit b8 in Figure 62).

The address extensions of the send/request DLPDU shall be sent back mirrored in the response DLPDU.

EXT = 0: No address extension in the DATA_UNIT

EXT = 1: Address extension follows in the DATA_UNIT

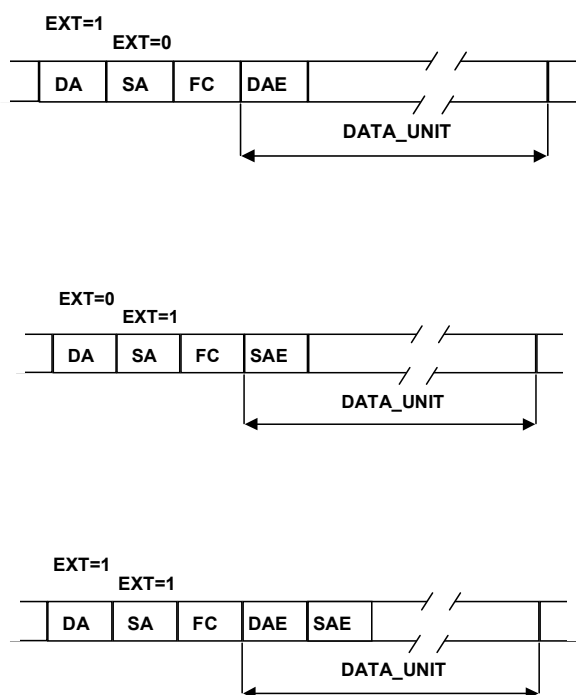
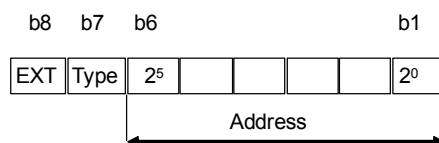


Figure 61 – DAE/SAE octet in the DLPDU



where

b8 (EXT) denotes an additional address extension:

- 0: No additional address extension octet
- 1: One additional address extension octet follows immediately. The following order shall be followed:
 First octet: region/DL-segment address with b7=1, b8=1
 Second octet: DLSAP with b7=0, b8=0

b7 denotes the type:

- 0: 6 bit DL-service-access-point (DLSAP): DAE = 0 to 63; SAE = 0 to 62
- 1: 6 bit region/DL-segment address to realize hierarchical systems, values are not specified in this standard.

Figure 62 – Address extension octet

21.3.3 Address check

A receiver shall check the destination address information in a DLPDU addressed to itself against TS according to the following rules:

- a) If there is no region/DL-segment address in the DLPDU existent (DA[b8=0] or DAE[b7=0]), it shall only check the DA on equality with TS.
- b) If there is a region/DL-segment address in the DLPDU existent (DAE[b7=1]), it shall check the DAE and the DA on equality with region/DL-segment address and TS of the addressed station.
- c) If the receiver of the station, which is addressed with a region/DL-segment address, does not contain a region/DL-segment address then the DLPDU is not addressed to this station.

21.3.4 DL-service-access-point (DLSAP)

At the DLS interface, (see IEC 61158-3) a data transmission service (or several thereof) is processed via a DL-service-access-point (DLSAP). Several DLSAPs may exist at the same time in Master and Slave stations. The related DLSAP-address shall be transmitted together with the message, except for the DLSAP-addresses NIL and CS.

The address extensions DAE and SAE shall be used for the transmission of DLSAPs. The source service access point index (S_SAP_index), which represents the access address of the local user to the DL, shall be transmitted in the SAE octet. The destination service access point index (D_SAP_index), which represents one or all access addresses of the remote user to the DL, shall be transmitted in the DAE octet. S_SAP_index values from 0 to 62 and D_SAP_index values from 0 to 63 may be chosen. The D_SAP_index value 63 denotes the global access address. This D_SAP_index may only be used for the Send Data with No Acknowledge service.

NOTE For DLPDUs with a Function = SDA_H/L or SRD_H/L (see Table 196) a D_SAP_index of 63 in the send/request DLPDU results in a negative acknowledge (RS).

If the transmission of DLSAP addresses is omitted for reasons of DLPDU efficiency, the data transmission services shall be processed via the **default DLSAP** at the initiator or the responder or both. In these cases, all DLPDUs shall be transmitted without the related address extension (SAE or DAE or both). At the DLS interface the default DLSAP is mandatory and is addressed with the value NIL.

The DLSAP-address CS used for the time synchronization services Clock Value and Time Event is omitted in the related DLPDUs for the same reason.

21.4 Control octet (FC)

21.4.1 General

The control octet in the DLPDU header shall indicate the DLPDU type, such as send/request DLPDU and acknowledgement or response DLPDU. In addition, the control octet shall contain the function Code No and the control information, which prevents loss and multiplication of messages, or the station type with the DL status. Figure 63 and Figure 64 illustrate the FC octet coding.

b8	b7	b6	b5	b4	b3	b2	b1
0/1	1	FCB	FCV	2 ³	Function code No		2 ⁰

where

b8 = 0/1, b7 = 1: send/request DLPDU

FCB: is Frame count bit: 0 or 1, alternating

FCV: is Frame count bit valid:

0: alternating function of FCB is invalid

1: alternating function of FCB is valid

Function code No: shall be as described in Table 196

Figure 63 – FC octet coding for send/request DLPDUs

b8	b7	b6	b5	b4	b3	b2	b1
Res	0	Station type and DL status		2 ³	Function code No		2 ⁰

where

DLPDU type b7 = 0: is acknowledgement or response DLPDU

b6 and b5 are station type and DL status combined

b6 b5

0 0 Slave station

0 1 Master station not ready to enter logical token ring

1 0 Master station ready to enter logical token ring

1 1 Master station in logical token ring

Res: means reserved (the sender shall set to binary "0", the receiver does not have to interpret this bit)

Function code No: shall be as described in Table 196

Figure 64 – FC octet coding for acknowledgement or response DLPDUs

Table 196 shows the function code No of the control octet for both send/request DLPDUs and acknowledgement or response DLPDUs.

Table 196 – Transmission function code No

Code	Function	Format in subclause described	Possible for the following stations	
	DLPDU type b8 = 1 & b7 = 1		Initiator	Receiver/Responder
0	Clock Value	22.3.1 , 22.3.2	M	M and S
1...15	Reserved			
	DLPDU type b8 = 0 & b7 = 1			
0	Time Event	22.1.1 , 22.1.2	M	M and S
1,2	Reserved			
3	Send Data with Acknowledge low	22.2.1 , 22.3.1 22.2.2 , 22.3.2	M	M and S
4	Send Data with No Acknowledge low		M	M and S
5	Send Data with Acknowledge high		M	M and S
6	Send Data with No Acknowledge high		M	M and S
7	Send and Request Data Multicast	22.2.1 , 22.3.1 , 22.2.2 , 22.3.2	M	M and S
8	Reserved			
9	Request DL Status with Reply	22.1.1 , 22.1.2	M	M and S
10, 11	Reserved			
12	Send and Request Data low	22.1.1 , 22.2.1 , 22.3.1	M	M and S
13	Send and Request Data high	22.1.2 , 22.2.2 , 22.3.2	M	M and S
14	Request Ident with Reply	22.1.1 , 22.1.2	M	M and S
15	Reserved			
	DLPDU type b7 = 0			
0	Acknowledgement positive (OK)	22.1.1 , 22.1.2 ^a	M and S	M
1	ACK negative DL/DLM User Error (UE)	22.1.1 , 22.1.2	M and S	M
2	ACK negative no resource for send data (& no response DL data) (RR)		M and S	M
3	ACK negative no service activated (RS)		M and S	M
4 to 7	Reserved		M and S	M
8	Response DL/DLM data low (& send data ok) (DL)	22.2.1 , 22.3.1 22.2.2 , 22.3.2	M and S	M
9	ACK negative no response DL/DLM data, (& send data ok) (NR)	22.1.1 , 22.1.2 ^a	M and S	M
10	Response DL data high, (& send data ok) (DH)	22.2.1 , 22.3.1 22.2.2 , 22.3.2	M and S	M
11	Reserved		M and S	M
12	Response DL data low, no resource for send data (RDL)	22.2.1 , 22.3.1	M and S	M
13	Response DL data high, no resource for send data (RDH)	22.2.2 , 22.3.2	M and S	M
14, 15	Reserved			

where M: Master, S: Slave

Function code No 0: b4 b1
 0: 0 0 0 0

Function code No 15: 1 1 1 1

() Value of the L/M_status Parameter of the Service Primitives (see IEC 61158-3)

^a Also possible Short Acknowledgement SC = E5H;
exception: if Request DL Status with Reply, no SC is permitted

21.4.2 Frame count bit

The frame count bit FCB (b6) prevents the duplication of messages at the responder and the loss at the initiator. However, "Send Data with No Acknowledge" (SDN), "Request DL Status with Reply", "Request Ident with Reply", "Time Event" and "Clock Value" are excluded from this.

In order to manage the security sequence, the initiator shall carry a FCB for each responder. When an send/request DLPDU is transmitted to a responder for the first time or to a responder currently marked as "non operational", the associated FCB shall be set unambiguously. The initiator shall achieve this by an send/request DLPDU with FCV=0 and FCB=1. The responder shall classify such a DLPDU as first message transfer period and store FCB=1 together with the initiator's address (SA and optional SAE[b7=1]) (see Table 197). This message transfer period is not repeated by the initiator.

If a responder supports the region/DL-segment addressing and the send/request DLPDU contains a source region/DL-segment address (SAE[b7=1]), which is unequal to the own region/DL-segment address (DAE[b7=1]), then the responder shall store the SAE[b7=1] together with the SA.

In the following send/request DLPDUs to the same responder the initiator shall set FCV=1 and toggle FCB with each new send/request DLPDU. The responder shall evaluate FCB when receiving an send/request DLPDU addressed to itself with FCV=1. A FCB changed in comparison with the same initiator's (same SA and optional same SAE[b7=1]) preceding send/request DLPDU shall be considered as confirmation of the preceding message transfer period's correct completion. If the send/request DLPDU originates from a different initiator (different SA or optional different SAE[b7=1]), there shall be no evaluation of the FCB. In both cases, the responder shall store the FCB with the source address (SA and optional SAE[b7=1]) until it receives a new DLPDU addressed to itself.

If an acknowledgement or response DLPDU is missing or corrupted, the FCB shall **not** be changed by the initiator in the retry; this indicates the faulty preceding message transfer period. If a responder receives an send/request DLPDU with FCV=1 and the same FCB as in the same initiator's (same SA and optional same SAE[b7=1]) immediately preceding send/request DLPDU, a retry shall be carried out. As a result, the responder shall again transmit the acknowledgement or response DLPDU kept in readiness.

The responder shall keep ready the preceding acknowledgement or response DLPDU for a potential retry, until it detects a confirmation of the preceding message transfer period's correct completion as described above, or until it receives a token DLPDU or a DLPDU with a changed address (SA or DA or optional SAE[b7=1] or DAE[b7=1]).

For "Send Data with No Acknowledge", "Request DL Status with Reply", "Request Ident with Reply", "Time Event" and "Clock Value", FCV and FCB are both zero for the following DLPDU types; the responder does not need to analyse FCB:

Table 197 – FCB, FCV in responder

b6 FCB	b5 FCV	← bit position		
		Condition	Meaning	Action
0	0	DA = TS/127	Request with No Ack Request DL Status with Reply Request Ident with Reply Time Event Clock Event	last Ack or Reply may be deleted
0/1	0/1	DA # TS	Request to other Responder	last Ack or Reply may be deleted
1	0	DA = TS	First Request	FCBM := 1 SAM := SA last Ack or Reply may be deleted
0/1	1	DA = TS SA = SAM FCB # FCBM	New Request	last Ack or delete Reply FCBM := FCB have Ack or Reply ready for Retry
0/1	1	DA = TS SA = SAM FCB = FCBM	Request Retry	FCBM := FCB repeat Ack or Reply and keep it ready
0/1	1	DA = TS SA # SAM	New Initiator	FCBM := FCB SAM := SA have Ack or Reply ready for Retry
—	—	Token-DLPDU	—	last Ack or Reply may be deleted
where FCBM is the stored FCB SAM is the stored SA				

21.5 DLPDU content error detection

21.5.1 Asynchronous transmission – frame checksum (FCS)

The 8-bit (1 octet) frame checksum that is required for Hamming distance 4 in a DLPDU shall always immediately precede the end delimiter. The checksum shall be structured as shown in Figure 65.



Figure 65 – FCS octet coding

In DLPDUs of fixed length with no data field (see 22.1.1) the checksum shall be calculated from the two's-complement arithmetic sum of DA, SA and FC (without start and end delimiters).

In DLPDUs of fixed length with data field (see 22.2.1) and in DLPDUs with variable data field length (see 22.3.1) the checksum shall additionally include the DATA_UNIT.

21.5.2 Synchronous transmission –frame check sequence (FCS)

A 16-bit (2 octet) frame check sequence is required. It shall be calculated and appended to the DLPDU as specified in 5.1, where its Hamming distance properties are also described.

21.6 DATA_UNIT

21.6.1 General

The DATA_UNIT shall consist of the User Data (DLSDU) of the DLS/DLMS-user and optionally of one or more address extension octets (see Figure 66). Up to 4 address extension octets are permissible (see 21.3.1). The User Data comprises a maximum of 242 up to 246 octets depending on the number of used address extension octets.

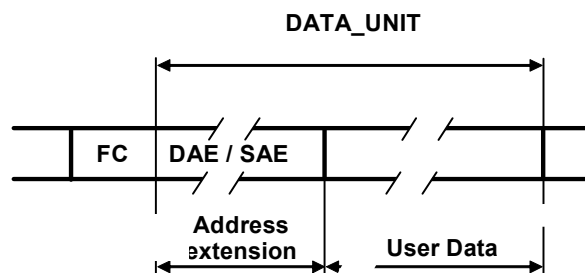
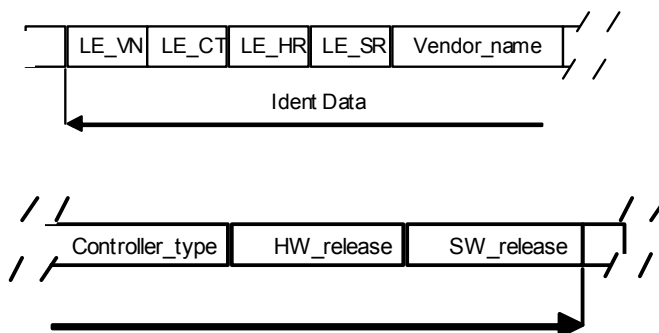


Figure 66 – Data field

The following user data are defined for the Ident DLPDU:

21.6.2 Ident user data

As shown in Figure 67 the Ident user data shall contain the station's Ident_List with vendor name (Vendor_name), fieldbus controller type (controller_type) and hardware and software release (HW/SW_release). It comprises a maximum of 200 octets:



where

LE_VN, LE_CT, LE_HR, LE_SR is

1 octet each. Length of corresponding data field in octets;

dual coding, significance as in Figure 65.

Vendor_name, VN is

Name of manufacturer as ASCII string (ISO 7 Bit Code, b8=0).

Controller_type, CT is

Hardware controller type as ASCII string (ISO 7 Bit Code, b8=0).

HW_release, HR is

Hardware release of controller as ASCII string (ISO 7 Bit Code, b8=0).

SW_release, SR is

Software release of controller as ASCII string (ISO 7 Bit Code, b8=0).

Figure 67 – Ident user data

21.7 Error control procedures

21.7.1 Asynchronous transmission

Line protocol errors, for example, character framing errors, overrun errors and parity errors, and transmission protocol errors, for example faulty start delimiters, frame check octets and end delimiters, invalid DLPDU length, response times, etc. shall result in the following station reactions:

A send/request DLPDU or token DLPDU that has been received incorrectly by a station shall not be processed, acknowledged or answered. The initiator shall retry the request after expiration of the slot time. The request shall also be retried if the acknowledgement or response was corrupted. The initiator shall complete a request only after having received a valid response or if the retry (retries) was not (were not) successful (see Table 198). This means that a "Send/Request" shall be kept until the responder confirms its correct receipt by an acknowledgement or response or the retry (retries) was not (were not) successful. In the same way, a responder shall terminate a "Request" or "Send/Request" only if a new request with altered frame count bit is received or another station is addressed (see 21.4, FCB).

If a station does not acknowledge or respond after retry (retries), it shall be marked as "non operational". When processing the following requests, the initiator shall transmit the request to this station without retry, until the station acknowledges or responds correctly again. After positive acknowledgement, the initiator shall mark again the addressed station as "operational". When processing the next request, the initiator shall continue the original mode of operation with this station.

21.7.2 Synchronous transmission

Errors in the line protocol (see IEC 61158-2:2003¹³, Clause 9) and in the medium access protocol, such as erroneous start octets and FCS octets, DLPDU length, response times etc. shall result in the station reactions specified in 21.7.1.

¹³ Expected date of concurrent publication.

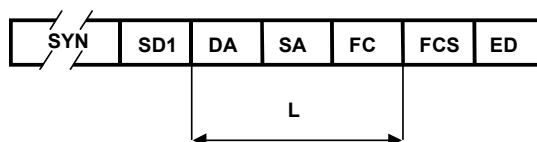
22 Type 3: DLPDU-specific structure, encoding and elements of procedure

22.1 DLPDUs of fixed length with no data field

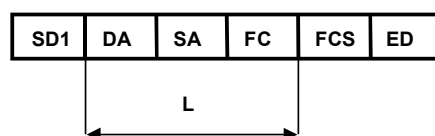
22.1.1 Asynchronous transmission

The formats of DLPDUs of fixed length with no data field shall be as shown in Figure 68.

- a) Format of the request DLPDU:



- b) Format of the acknowledgement DLPDU:



- c) Format of the short acknowledgement DLPDU:



where

- SYN is the synchronization period, a minimum of 33 line idle bits
- SD1 is the start delimiter, value: 10H
- DA is the destination address
- SA is the source address
- FC is the frame control
- FCS is the frame checksum
- ED is the end delimiter, value: 16H
- L is the information field length, fixed number of octets: L = 3
- SC is the single character, value: E5H

Figure 68 – DLPDUs of fixed length with no data field

Transmission Rules

- 1) Line idle state shall correspond to signalling level binary "1".
- 2) Each request DLPDU shall be preceded by at least 33 line idle bits (synchronization time).
- 3) No idle states are allowed between a DLPDU's UART characters.
- 4) The receiver shall check:
 - for each UART character: start bit, stop bit and parity bit (even),
 - for each DLPDU: start delimiter, DA, SA, FCS and end delimiter
 - and the synchronization time in case of a request DLPDU.

If the check fails, the entire DLPDU shall be discarded.

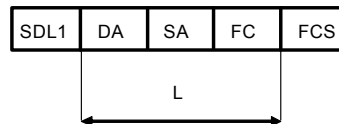
SC and SD1 (as well as SD2 and SD3, see 22.2.1 and 22.3.1) have Hamming distance $H_d=4$ and are safe against being shifted (see IEC 60870-5-1), that is, the single character SC appears to be a DLPDU with $H_d=4$.

For requests to be acknowledged (Send Data with Acknowledge), only SC is a permissible positive acknowledgement. For requests to be answered (Send and Request Data with Reply), SC is permissible if no data is available (see Table 196, b7=0, Code-No 9).

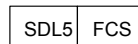
22.1.2 Synchronous transmission

The formats of DLPDUs of fixed length with no data field shall be as shown in Figure 69.

- a) Format of the request DLPDU and format of the acknowledge DLPDU:



- b) Format of the short acknowledgement DLPDU:



where

SDL1 is the start octet 1 (start delimiter 1 data link), code: 10H

SDL5 is the start octet 5 (start delimiter 5 data link), code: E5H

DA is the destination address

SA is the source address

FC is the frame control

FCS is the frame check sequence, 2 octets

L is the information field length, fixed number of octets: L = 3

Figure 69 – DLPDUs of fixed length with no data field

Transmission Rules

In addition to the transmission rules of the Ph-layer in IEC 61158-2:2003¹⁴, Clause 9, the receiver shall check the SDL, DA/SA and FCS octets for each DLPDU. If the check fails, the whole DLPDU shall be discarded.

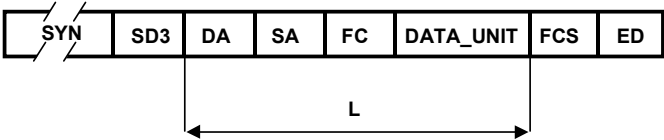
¹⁴ Expected date of concurrent publication.

22.2 DLPDUs of fixed length with data field

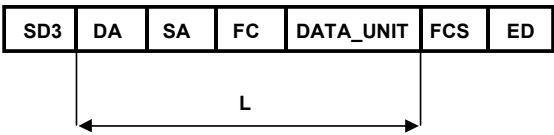
22.2.1 Asynchronous transmission

The format of DLPDUs of fixed length with data field shall be as shown in Figure 70.

a) Format of the send/request DLPDU:



b) Format of the response DLPDU:



- where
- SYN is the synchronization period, a minimum of 33 line idle bits
 - SD3 is the start delimiter, value: A2H
 - DA is the destination address
 - SA is the source address
 - FC is the frame control
 - DATA_UNIT is the data field, fixed length (L-3) = 8 octets
 - FCS is the frame checksum
 - ED is the end delimiter, value: 16H
 - L is the information field length, fixed number of octets: L = 11

Figure 70 – DLPDUs of fixed length with data field

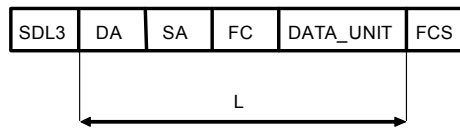
Transmission Rules

The same transmission rules shall apply as for DLPDUs of fixed length with no data field (see 22.1.1).

22.2.2 Synchronous transmission

The formats of DLPDUs of fixed length with data field shall be as shown in Figure 71.

a) Format of the send/request and response DLPDUs:



where

- SDL3 is the start delimiter 3 data link, code: A2H
- DA is the destination address
- SA is the source address
- FC is the frame control
- DATA_UNIT is the data field, fixed length (L-3) = 8 octets
- FCS is the frame check sequence, 2 octets
- L is the information field length, fixed number of octets: L = 11

Figure 71 – DLPDUs of fixed length with data field

Transmission Rules

The same transmission rules shall apply as for DLPDUs of fixed length with no data field (see 22.1.2).

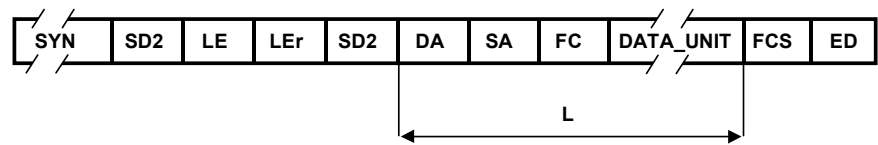
22.3 DLPDUs with variable data field length

22.3.1 Asynchronous transmission

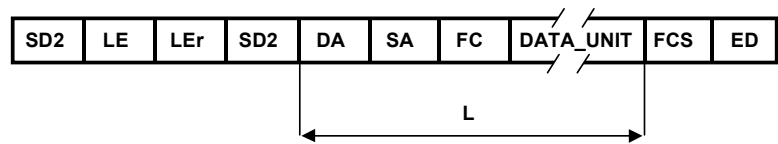
For a variable number of data octets the length information shall also be transmitted in the DLPDU. This length information shall be contained twice in a fixed DLPDU header at the beginning of the DLPDU. Thus it is protected with Hd = 4 and safe against slip.

The format of DLPDUs with variable data field length shall be as shown in Figure 72.

a) Format of the send/request DLPDU:



b) Format of the response DLPDU:



- where
- SYN is the synchronization period, a minimum of 33 line idle bits
 - SD2 is the start delimiter, value: 68H
 - LE is the octet length, allowed values: 4 to 249
 - LEr is the octet length repeated
 - DA is the destination address
 - SA is the source address
 - FC is the frame control
 - DATA_UNIT is the data field, variable length (L-3), max. 246 octets
 - FCS is the frame checksum
 - ED is the end delimiter, value: 16H
 - L is the information field length, variable number of octets: L = 4 to 249

Figure 72 – DLPDUs with variable data field length

Transmission Rules

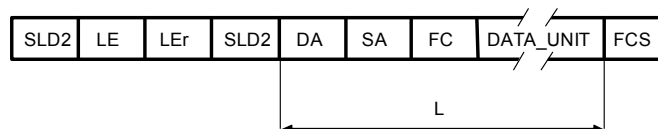
The same transmission rules as for DLPDUs of fixed length with no data field shall be applied (see 22.1.1).

In addition to transmission rule 4, LE shall be identical to LEr, and the information octets shall be counted from the destination address (DA) up to the frame checksum (FCS) and the result shall be compared with LE.

22.3.2 Synchronous transmission

The formats of DLPDUs with variable data field length shall be as shown in Figure 73.

a) Format of the send/request and response DLPDUs:



where

- SDL2 is the start delimiter 2 data link, code: 68H
- LE is the length, value: 4 to 249
- LEr is the length (repeated)
- DA is the destination address
- SA is the source address
- FC is the frame control
- DATA_UNIT is the data field, variable length (L-3), max. 246 octets
- FCS is the frame check sequence, 2 octets
- L is the information field length, variable number of octets: L = 4 to 249

Figure 73 – DLPDUs with variable data field length

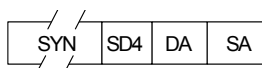
Transmission Rules

The same transmission rules as for DLPDUs of fixed length with no data field shall apply (see 22.1.2). In addition, the receiver shall check if LE and LEr coincide. The information octets shall be counted from the destination address (DA) up to the frame check sequence (FCS) and shall be compared with LE.

22.4 Token DLPDU

22.4.1 Asynchronous transmission

The format of the token DLPDU shall be as shown in Figure 74.



where

- SYN is the synchronization period, a minimum of 33 line idle bits
- SD4 is the start delimiter, value: DCH
- DA is the destination address
- SA is the source address

Figure 74 – Token DLPDU

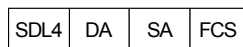
Transmission Rules

- 1) Line idle state shall correspond to signalling level binary "1".
- 2) Each token DLPDU shall be preceded by at least 33 line idle bits (synchronization time).
- 3) No idle states are permitted between a DLPDU's UART characters.
- 4) The receiver shall check:
 - per UART character: start bit, stop bit and parity bit (even),
 - per DLPDU: synchronization time, start delimiter and DA/SA.

If the result of the check is negative, the whole DLPDU shall be discarded.

22.4.2 Synchronous transmission

The format of the token DLPDU shall be as shown in Figure 75.



where

SDL4 is the start delimiter 4 data link, code: DCH

DA is the destination address

SA is the source address

FCS is the frame check sequence, 2 octets

Figure 75 – Token DLPDU

Transmission Rules

The same transmission rules as for DLPDUs of fixed length with no data shall apply (see 22.1.2).

22.5 ASP DLPDU

The ASP DLPDU is generated by the DLE in the isochronous mode to produce bus activity in the spare time of an isochronous cycle to avoid a time-out (T_{TO}). The format of the ASP DLPDU is the same as described in 22.1, with the following restrictions:

- Control Octet (FC) is equal to Request DL Status with Reply (Code 49H).
- DA is equal to SA.
- There is no response DLPDU by any DLE for this ASP DLPDU.

22.6 SYNCH DLPDU

The SYNCH DLPDU is generated by the DLE in the isochronous mode to mark the beginning of a new isochronous cycle. The format of the SYNCH DLPDU is the same as described in 22.3, with the following restrictions:

- FC, the Control Octet, is equal to Send Data with No Acknowledge high (Code 46H)
- DA is equal to 127
- SAE is equal to 62
- DAE is equal to 58
- the DATA_UNIT shall contain the value of the operating parameter SYNCHT
- there is no response DLPDU by any DLE for this SYNCH DLPDU

22.7 Time Event (TE) DLPDU

The TE DLPDU is used to synchronize the clock at the time receiver. The format of the TE DLPDU is the same as described in 22.1, with the following restrictions:

- FC, the Control Octet, is equal to Time Event (Code 40H)
- DA is equal to 127
- there is no response DLPDU by any DLE for this TE DLPDU

22.8 Clock Value (CV) DLPDU

The CV DLPDU is used to transmit the clock value to the time receivers. The format of the CV DLPDU is the same as described in 22.3, with the following restrictions:

- FC, the Control Octet, is equal to Clock Value (Code C0H)

- DA is equal to 127
- SAE and DAE are not used
- there is no response DLPDU by any DLE for this CV DLPDU.

22.9 Transmission procedures

22.9.1 Asynchronous transmission

Permissible DLPDU sequences (message transfer periods) for asynchronous transmission are described in Figure 76 through Figure 78. Error sequences and broadcast/multicast messages are excluded.

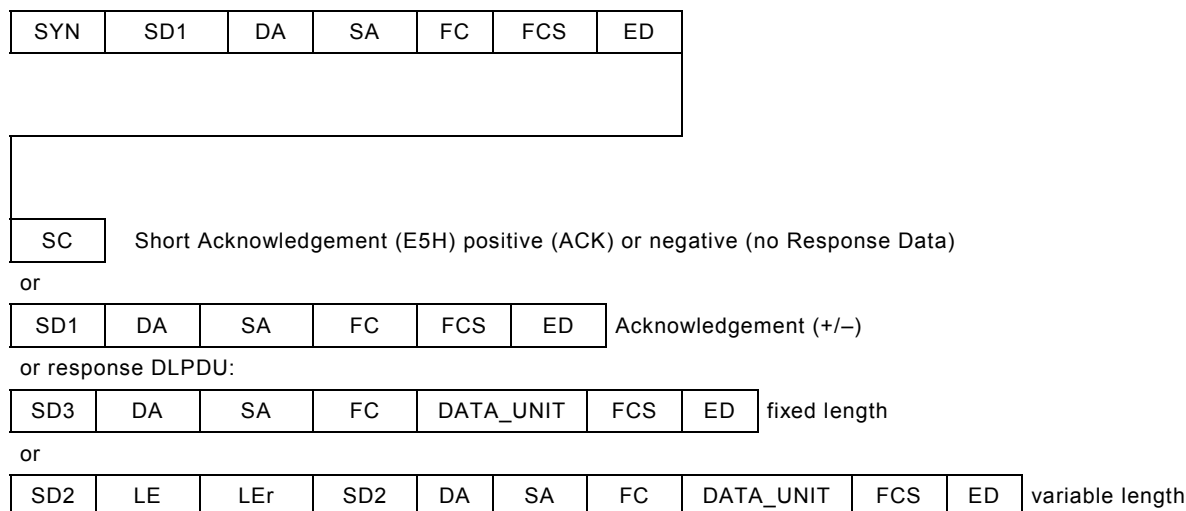


Figure 76 – Send/request DLPDU of fixed length with no data

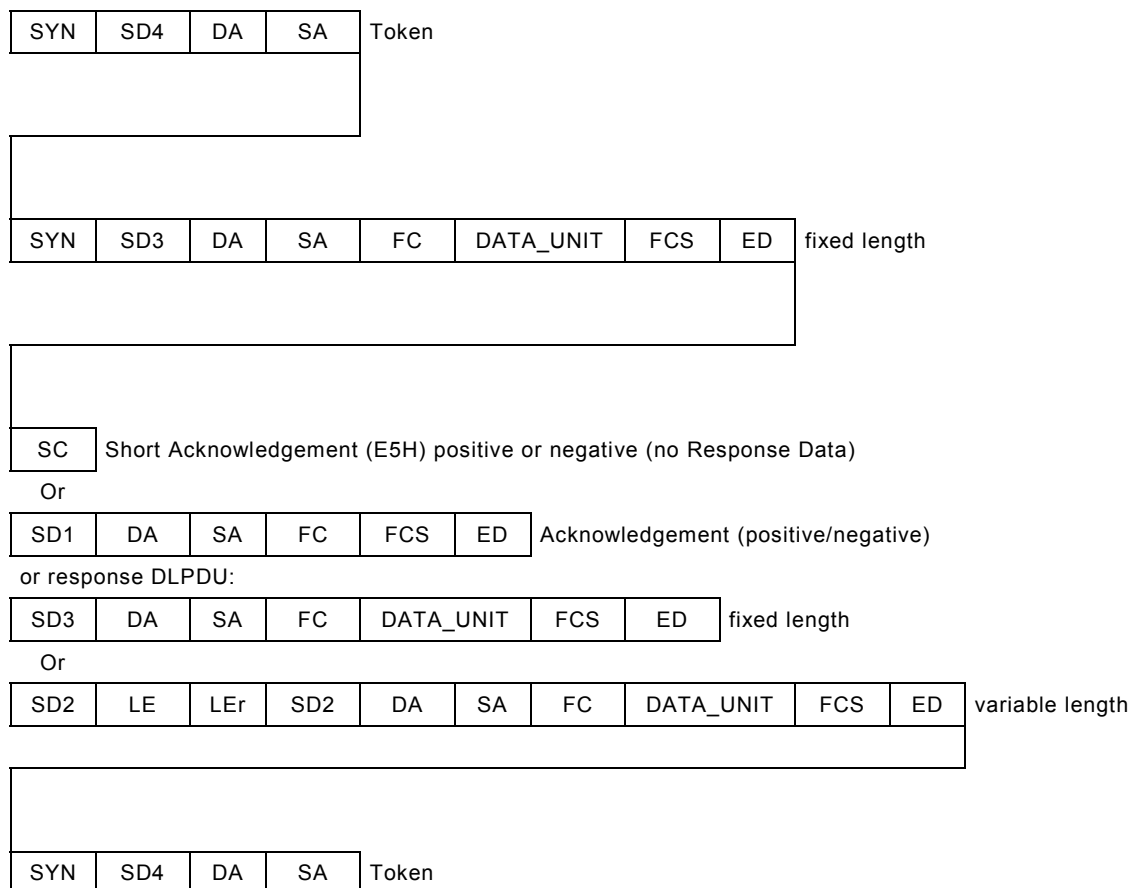


Figure 77 – Token DLPDU and send/request DLPDU of fixed length with data

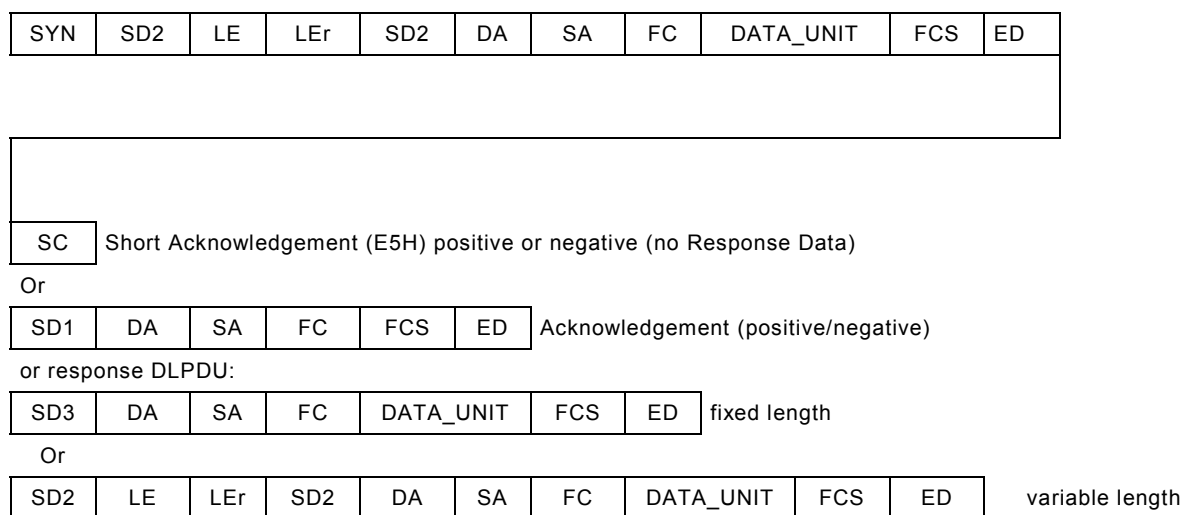


Figure 78 – Send/request DLPDU with variable data field length

22.9.2 Synchronous transmission

Permissible DLPDU sequences (message transfer periods) for synchronous transmission are described in Figure 79 through Figure 81. Error sequences and broadcast/multicast messages are excluded.

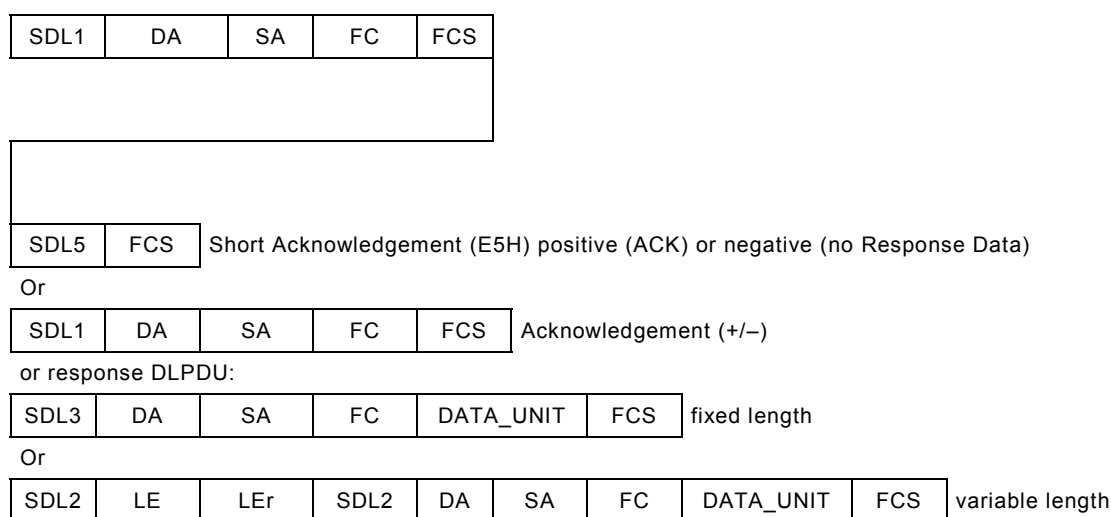


Figure 79 – Send/request DLPDU of fixed length with no data

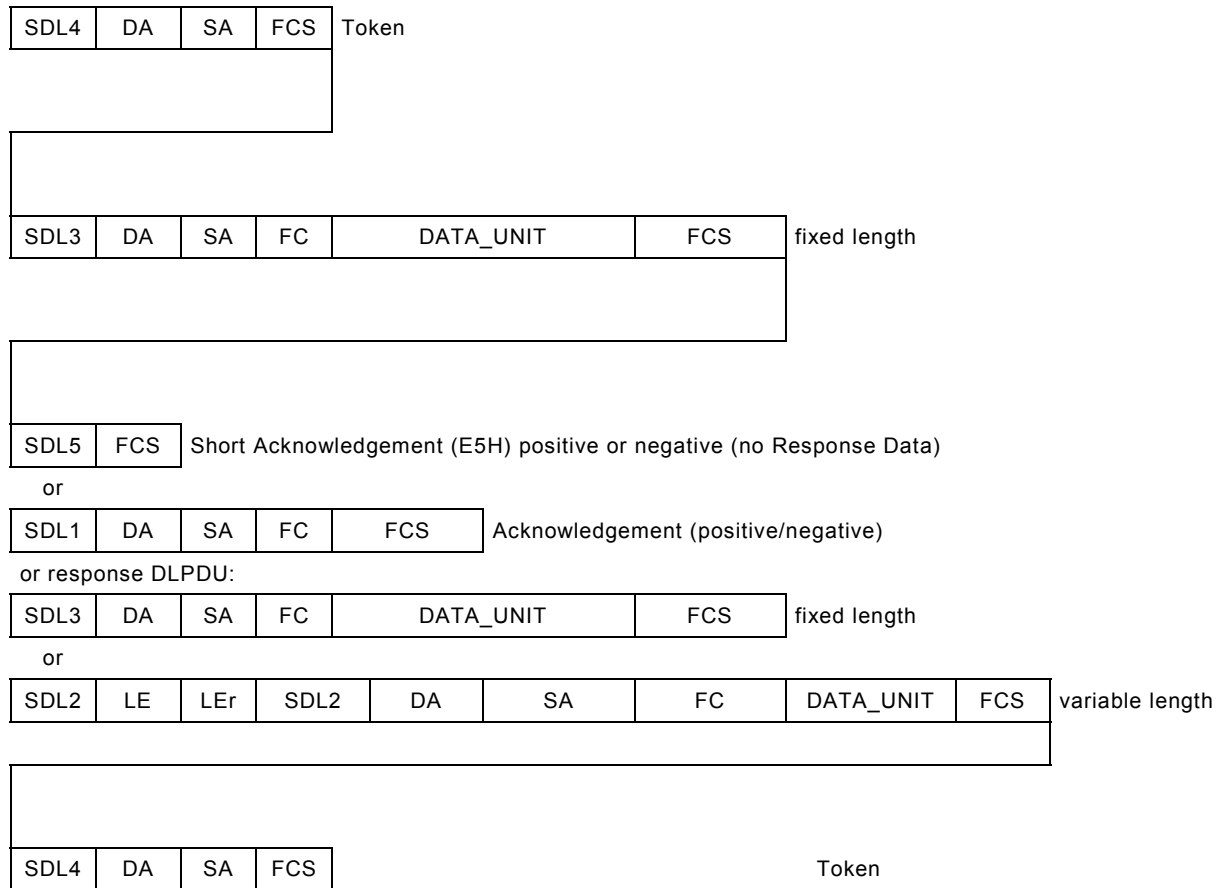


Figure 80 – Token DLPDU and send/request DLPDU of fixed length with data

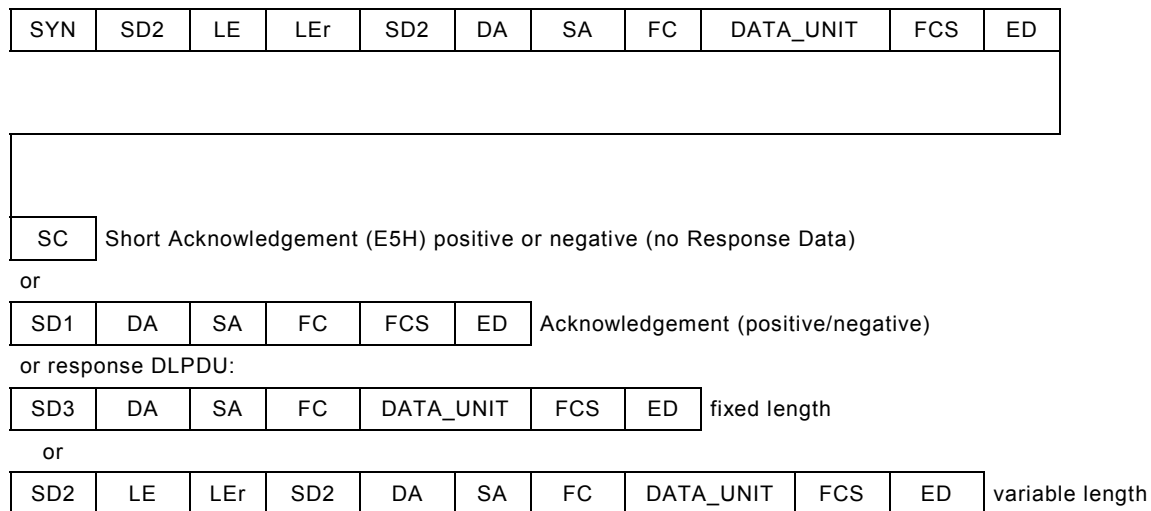


Figure 81 – Send/request DLPDU with variable data field length

23 Type 3: Other DLE elements of procedure

NOTE Annex E specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. The specification of Annex E is complementary to the textual specification in this and related clauses in the body of this standard. In case of conflict the requirements of Annex E take precedence.

23.1 DL-entity initialization

After power on (PON) the DL-entity of Master and Slave stations shall enter the "Offline" state. Within this state, the DL-entity does not receive or transmit any signals (DLPDUs) from or to the bus.

The DL-entity may enter the "Passive_Idle" or "Listen_Token" state from the "Offline" state only, if the operating parameters (DL-variables) have been set for correct protocol handling (see IEC 61158-3). The operating parameters, shown in Table 198, shall be provided by the DLMS-user.

Table 198 – Operating parameters

Parameter number	Name
1	Station DL-address TS
2	Data rate (kbit/s)
3	Single/redundant media available
4	Hardware release
5	Software release
6	Slot time T_{SL}
7	Station delay time min T_{SDR}
8 (Note 1)	Station delay time max T_{SDR}
9 (Note 1)	Transmitter fall/Repeater switch time T_{QUI}
10 (Note 1)	Setup time T_{SET}
11 (Note 1)	Target rotation time T_{TR}
12 (Note 1)	GAP update factor G
13 (Note 1)	Master station enter/leave the logical ring (in_ring_desired)
14 (Note 1)	Highest station address (HSA)
15 (Note 1)	Maximum number of retries (max_retry_limit)
16 (Note 2)	Clock synchronization interval T_{CSI}
17 (Note 3)	Contents of the SYNCH User Data (SYNCHT)
18 (Note 3)	Isochronous cycle time T_{CT}
19 (Note 3)	Allowed maximal time shift T_{TS}
NOTE 1 This applies only to Master stations	
NOTE 2 This applies only to stations able to support clock synchronization	
NOTE 3 This applies only to stations able to work in isochronous mode	

23.2 States of the media access control of the DL-entity

23.2.1 General

A Master station's media access control state machine (MAC) is described by means of 11 states and the transitions between them. A Slave station's MAC has 2 states.

Figure 82 shows as an overview the combined state diagram of the Master (state 0 to 9 and 11) and the Slave station (state 0 and 10).

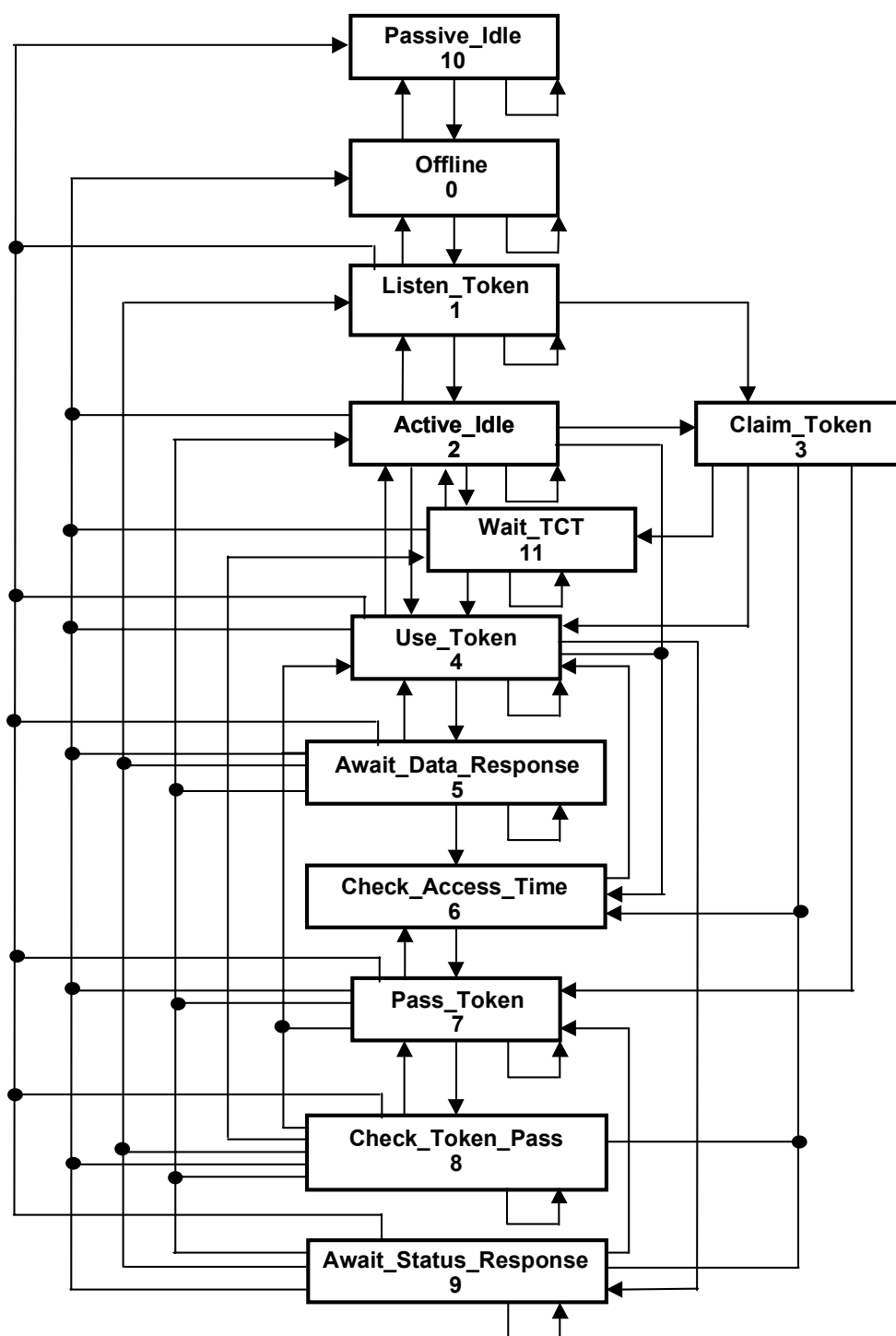


Figure 82 – DL-state-diagram

23.2.2 Offline

The "Offline" state shall be entered immediately after power on, after the DLM-RESET service, or after certain error conditions have been detected. After power on each station should perform a self-test. This internal self-test depends on the implementation and does not influence the other stations. For this reason, the self-test procedure is not specified in this specification.

After terminating the power on sequence, the MAC remains in the "Offline" state until all required operating parameters (see 23.1) have been initialized. The DL-entity may only afterwards connect to the transmission medium, but without becoming active.

23.2.3 Passive_Idle

After its parameters are initialized, the Slave station's MAC or the Master station's MAC depending on the operating parameter "in_ring_desired" (in_ring_desired equal false) (see Table 198) shall enter the "Passive_Idle" state and listen to the line. If a plausible send/request DLPDU addressed to that station is received, the DL-entity shall acknowledge or respond as required, except for DLPDUs with global address (broadcast message, see 21.3.1) and token DLPDUs addressed to itself. The token DLPDU is discarded.

At the occurrence of the DLM-RESET service the MAC re-enters the "Offline" state.

23.2.4 Listen_Token

After its operating parameters have been initialized and the value of the parameter "in_ring_desired" is true, the Master station's MAC shall enter the "Listen_Token" state. In this state the Master station's DL-entity shall monitor the line in order to identify those Master stations which are already in the logical token ring. For that purpose, token DLPDUs are analysed and the station addresses contained in them are used to generate the list of Master stations (LMS). After listening to two complete token rotations, the MAC shall enter the state "Active_Idle".

During LMS generation a "Request DL Status with Reply" responds with "Master station not ready" after one token rotation is completed. All other DLPDUs are not processed in the "Listen_Token" state, that is, they are neither acknowledged/responded nor indicated to the local DLS-user.

If the MAC detects its own address as source address (SA) in two token DLPDUs when registering the Master stations, it shall assume that another Master station with the same address exists already in the ring. The MAC shall then re-enter the "Offline" state and report this event to the local DLMS-user.

If the MAC observes no bus activity for the time-out period, it shall assume that an initialization or a restoration of the logical token ring is necessary. The MAC shall enter the "Claim_Token" state.

23.2.5 Active_Idle

On leaving the "Listen_Token" state, the Master station's MAC shall enter the "Active_Idle" state and shall wait for received messages. If it receives an send/request DLPDU addressed to itself or as broadcast, it shall proceed, acknowledge or respond as required and shall unfreeze the token-rotation-timer if the token-rotation-timer is frozen.

The "Listen_Token" state shall be entered in the case of an error, if two token DLPDUs with SA = TS are received in immediate succession. This event shall be reported to the local DLMS-user.

If the MAC find out that it was taken from the logical token ring not on its own initiative, it also shall enter in the "Listen_Token" state and report this (Out_of_ring) to the local DLMS-user. The "Listen_Token" state shall also be entered in case of the MAC's LMS is not ready, that is, the SA and DA in the token DLPDU are not compliant with the LMS entry for a number of Tokens receipts. Token DLPDUs, which indicate the addition and removal of a Master, shall not be counted as not compliant.

If the MAC observes no bus activity for the time-out period, it shall assume that a restoration of the logical token ring is necessary. The MAC shall try to claim the token and to (re)initialize the logical ring ("Claim_Token" state).

If the MAC is addressed by a "Request DL Status with Reply" transmitted by its predecessor (PS) then:

- a) if the DL-address (TS) is contained in the LMS of the MAC, the MAC shall respond with "Master station in logical ring" or
- b) if the DL-address (TS) is not contained in the LMS of the MAC, the MAC shall respond with "Master station ready to enter logical token ring".

After receiving a token DLPDU addressed to the MAC itself and the isochronous mode is not set, it shall:

- 1) unfreeze the token-rotation-timer if this timer is frozen,
- 2) drop the token, if the token is not sent by the predecessor or if the DL-address (TS) is not contained in the LMS of the MAC. In these both cases, the MAC shall update the LMS.
- 3) otherwise, by processing the transition the T_{RR} (real rotation time) shall be calculated as the difference between target rotation time and the read value of the token-rotation-timer and the token-rotation-timer shall be restarted with target rotation time. The MAC shall enter:
 - i) the "Use_Token" state, if a high priority message is available or
 - ii) the "Check_Access_Time" state if no high priority message is available;

After receiving a token DLPDU addressed to the MAC itself and the isochronous mode is set, it shall:

- 4) send an ASP message and
- 5) change into the "Wait_TCT" state.

If the MAC receives an ASP message and the isochronous mode is set, it shall freeze the token-rotation-timer.

23.2.6 Claim_Token

The MAC shall enter the "Claim_Token" state after the "Active_Idle" state or the "Listen_Token" state, when its time-out has expired. In this state it shall try to initialize the logical ring or to start an initialization.

When re-initializing and the MAC is not in the isochronous mode, the DL-entity address is contained in the LMS of the MAC, and thus the "Use_Token" state is entered immediately, if high priority message are pending. Otherwise the MAC shall enter the "Check_Access_Time" state.

When re-initializing and the MAC is in the isochronous mode, the DL-entity address is contained in the LMS of the MAC, and thus the MAC shall send an ASP message and shall enter the "Wait_TCT" state.

When initializing (the DL-address (TS) is not contained in the LMS of the MAC), at first the token shall be addressed twice to the own DL-entity, that is, $NS = TS$, namely in the "Pass_Token" state. This is necessary in order to cause an entry in the other Master stations' LMS.

23.2.7 Wait_TCT

This state shall be entered after the "Claim-Token" state, "Check-Token_Pass" state or "Active_Idle" state when a ASP message was passed if the Master is in the isochronous mode. In this state the MAC shall pass as much ASP messages as spare time is available ($TRCT < (TCT - TPSP)$). If no time available to pass an ASP messages then the MAC shall

- a) load the passive-spare-time-timer with $TCT - TRCT$,
- b) start the passive-spare-time-timer,
- c) wait until the passive-spare-time-timer expires and
- d) pass a SYNCH message for synchronization purposes.

The MAC shall indicate the sending of the SYNCH message to the local DLMS-user by a Synch event. If the SYNCH message could not be sent within a defined maximum time shift ($maxTSH$) then the MAC shall pass a Synch_Delay event to the local DLMS-user in addition.

23.2.8 Use-Token

The MAC enters the "Use-Token" state in order to process high priority or low priority message transfer periods.

The MAC shall enter the:

- a) "Await_Data_Response" state after each transmitted send/request DLPDU with acknowledgement or response and start the slot-timer (see 20.5.5) or
- b) "Check_Access_Time" state in the case of a SDN or CS service is transmitted or
- c) "Await_Status_Response" state in the case of a "Request DL Status with Reply" is transmitted;

23.2.9 Await_Data_Response

This state shall be entered in the case of the transmission of requests with acknowledgement or response. The MAC waits **one** slot time for the receipt of the acknowledgement or response DLPDU.

The MAC shall enter the

- a) "Check_Access_Time" state in order to check the available token holding time for processing potential further requests, if:
 - 1) a valid acknowledgement DLPDU or valid response DLPDU addressed to the request's initiator or
 - 2) after the retry (retries) no valid acknowledgement or response is received and no more retry is possible. In this case the MAC shall notify the DLS-user accordingly. Further requests to this station are not repeated in case of errors, until a correct message transfer period (send/request data) has been performed;
- b) "Use-Token state, if:
 - 1) an invalid DLPDU is received (start-, end-, length-, FCS error; start-, stop-, parity-bit error or slip error) and a retry is possible or
 - 2) slot time expires (see 20.5) and a retry is possible;

and shall retry the transmission of the send/request DLPDU.

Receipt of another valid DLPDU indicates that an error has occurred. The MAC shall enter the "Active_Idle" state and discard the received DLPDU.

23.2.10 Check_Access_Time

In this state the available token holding time shall be computed (see 20.5.5). Only if there is still token holding time available, the MAC may re-enter the "Use_Token" state. Otherwise the MAC shall enter the "Pass_Token" state.

If token holding time is available then the MAC shall transmit:

- a) if high priority message are pending a high prior send/request DLPDU and shall enter the "Use_Token" state or
- b) if low priority message are pending a low prior send/request DLPDU and shall enter the "Use_Token" state or
- c) when the GAP update time has expired, but there is still token holding time T_{TH} available, then the MAC shall try to record one possible new station in the GAP before passing the token, in order to include it in the logical ring, if necessary. For that purpose, the MAC transmits a "Request DL Status with Reply" and enters the "Use_Token" state;

23.2.11 Pass_Token

In the "Pass_Token" state the MAC shall try to pass the token to the next station (NS) in the logical ring. When transmitting the token DLPDU, the MAC shall check by simultaneous monitoring if the transceiver is working correctly. If it does not receive its own token DLPDU, there is a fatal error in the transmit or receive channel. The MAC shall stop its activity in the logical ring, enter the "Offline" state and notify the DLMS-user.

If the MAC receives its own token DLPDU corrupted, this may be caused by a temporarily defective transmitter, receiver, or by the bus line. This error condition does not result in stopping activities in the first instance, instead the MAC shall enter the "Check_Token_Pass" state as after receiving the token DLPDU correctly.

Only after the token DLPDU has been retransmitted (due to no reaction from NS) and monitored as incorrect, the MAC shall stop its activity in the logical ring, enter the "Offline" state and notify the DLMS-user.

If the MAC has sent the token successfully then it shall enter the "Check_Token_Pass" state.

Only if no successor is known, that is, this DL-entity is at the moment the only active station on the bus, it shall pass the token to itself and then re-enter:

- a) the "Use_Token" state if a high priority message available and has been sent or
- b) the "Check_Access_Time" state if no high priority message available;

23.2.12 Check_Token_Pass

"Check_Token_Pass" is the state in which the MAC waits **one** slot time for a reaction of the station to which it has passed the token. This waiting time allows for the delay between the receipt of the token DLPDU and the ensuing transmission reaction of the addressed station.

If the MAC detects a valid DLPDU header within a slot time, it shall assume that the token passing was successful. The DLPDU shall be processed as if it were received in the "Active_Idle" state.

If an invalid DLPDU is detected within the slot time, the MAC shall assume that another station is active and therefore also enter the "Active_Idle" state.

If the MAC does not receive any DLPDU within one slot time, it shall pass the token to the successor again and re-enter the "Pass_Token" state and react as described in 20.3.2.1.

If the MAC does not receive any DLPDU within one slot time and it was the second pass token to the same successor then the MAC shall remove its NS from the LMS, pass the token to the new successor (NS of NS) and re-enter the "Pass-Token" state and react as described in 20.3.2.1.

If the MAC receives a Token with DA equal TS and the isochronous mode is active, then the MAC shall send an ASP message and shall enter the "Wait_TCT" state.

23.2.13 Await_Status_Response

This state is entered from the "Use-Token" state during GAP maintenance. In this state the MAC shall wait **one** slot time for an acknowledgement DLPDU.

The MAC shall enter the "Check_Access_Time" state:

- a) if nothing is received during the slot time or
- b) a corrupted DLPDU is received;

If an existing station does not answer it shall be deleted from the GAPL, that is, be marked as an unused address. Then the MAC shall enter the "Check_Access_Time" state.

If a status request is answered by a new Slave station or a Master station that does not want to be included in the ring, that station shall be entered in the GAPL.

If the MAC receives a valid acknowledgement with "Master station ready to enter logical token ring" then a new Master station acknowledges that it wants to be included in the logical ring. The MAC shall shorten the own GAP to the new NS and shall pass the token to this station and enter the "Pass-Token" state.

If the MAC receives any other DLPDU instead of an acknowledgement DLPDU (indicates that multiple tokens may exist), it shall enter the "Active_Idle" state.

23.3 Clock synchronization protocol

23.3.1 Overview

Synchronization of clocks between devices on a fieldbus segment and a time master application is provided in parallel with other communication functions. A time master is a fieldbus master device. The scheme used is a "backwards time based correction". This results in very few real-time constraints being imposed on a field device. There is no requirement for generating messages at precise instants. Instead, knowledge of when a special timer event message has been broadcasted is subsequently distributed and used to calculate appropriate clock adjustments.

There is no confirmation of correct reception at the remote DLE, as neither acknowledgements are given nor local retries take place. Once the data is sent it reaches all remote DLEs at the same time (not taking into account signal propagation time, Physical Layer delays and Data Link Layer delays). For the correct and secure time calculation the time transfer will take place in a sequence of two messages: the first transfer (Time Event) and the second transfer (Clock Value). By receiving the Timer Event, every remote DLE starts its receive-delay-timer. By receiving the clock value, the value of the receive-delay-timer is send within the indication. Each addressed remote DLE that has received the clock value error-free passes it to the DLS-user by means of a DL-CS-CLOCK-VALUE indication primitive (see IEC 61158-3, Type 3 fieldbus connectionless-mode Data Link Service, Clock Synchronization service). If errors occur by a violation of the sequence (e. g. a Timer Event is received twice or expiration of the Clock Synchronization Interval Timer), this error is notified to the DLS-user.

23.3.2 State machine time master

The clock synchronization sequence consists of two messages broadcasted by the time master. When the first message, called Time Event, is broadcast, all of the DLEs receiving it start a receive-delay-timer. The time master then sends a second message, the Clock Value, which contains the actual time when the Time Event was sent. Upon reception of this message, the remote DLEs can calculate their receive delay time value. In conjunction with the received clock value each DLS-users is able to back calculate the Time Event message reception time according to its own clock, and compare it with the value distributed in the clock message. Their clocks may then be adjusted to agree with the time master's. In order to support scheduling and to check that message pairs Time Event / Clock Value are matched, Clock Value also contain the time at which the last clock synchronization occurred, which is the time sent at the last Time Event.

Special DL services exist to support clock synchronization. A station's clock is adjusted by the DLS-user, but precise measurement of the time when sending or receiving the Time Event must be done in the DLE. Therefore the DLE uses special timers to measure the send delay respective receive delay.

On the sending side, the DLE in the time master starts its DL timer - the send-delay-timer - by receiving a DL-CS-TIME-EVENT request from the DLS-user. The DLE will send a TE DLPDU and stop the send-delay-timer when the last bit is sent. It will then deliver a confirmation to the DLS-user that the message was sent, including a service parameter which reports the calculated value of the send delay time. The DLS-user then adds this value to the time to determine when the Time Event was actually sent. A Clock Value DLPDU which includes this time is then broadcast to remote DLEs.

On the receiving side, the DLE starts its timer - the receive-delay-timer - when it receives the TE DLPDU. When the clock value is received (CV DLPDU), the receive-delay-timer is stopped and an indication is delivered to the DLS-user together with the calculated receive delay time to determine when the time message was received according to its clock.

The clock synchronization sequence supports optionally redundant time masters. Although only one is normally active at a given time, when switchover occurs more than one time master may be broadcasting the Time Event and Clock Value messages. Therefore, when the Time Event is received, the source station address is saved by the receiving DLEs. CV DLPDUs from other masters will be ignored.

Figure 83 illustrates an overview of the Clock Synchronization.

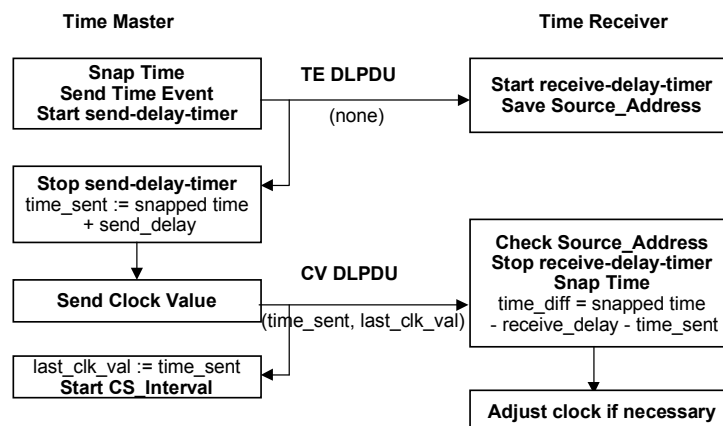


Figure 83 – Overview of clock synchronization

The clock synchronization protocol provides optionally redundant time masters, and a simple election procedure for determining the active time master. One time master on each link is presumed to be the primary time master. Other time masters are standby time masters that can act as sources of clock synchronization when the primary time master is not active. Standby time masters are not ranked in order of importance. The clock synchronization protocol will always select the primary time master as the clock synchronization source if it is present. The figure below is the state machine definition for the operation of a primary time master (TM) or a standby time master.

Standby time masters remain inactive as long as they receive clock synchronization messages from the active time master within a monitoring window. The monitoring window is restarted at each received clock synchronization message. The length of the monitoring window is set to $2 \times T_{CSI}$, a multiple of the Clock Synchronization Interval Time.

If no clock synchronization message is received during the monitoring window it is presumed that the active time master has failed. Standby time master then begin sending clock synchronization messages.

If a standby time master receives a clock synchronization message from another standby time master, then it will fall back to monitoring clock synchronization sequences. Messages received from the primary time master always result in the standby time master falling back to monitoring (see Figure 84).

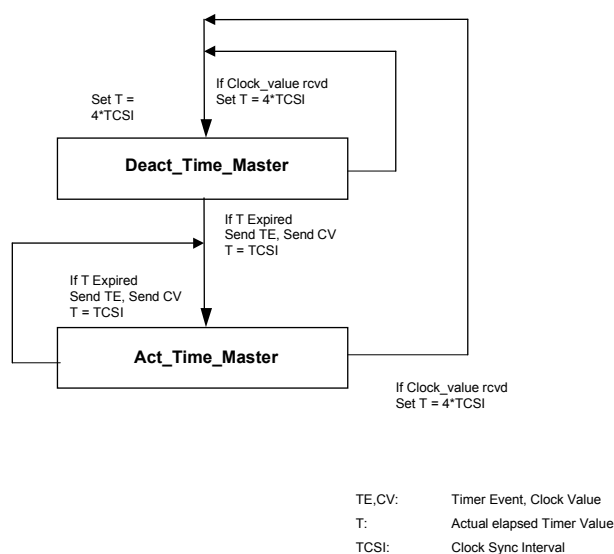


Figure 84 – Time master state machine

23.3.3 State machine time receiver

The receiver state machine is described in Figure 85. The receiver monitors clock messages within a time window of a multiple of the Clock Synchronization Interval Time: $(2 \times T_{CSI})$

It adjusts, if necessary, its clock after reception of a correct clock synchronization sequence. No clock adjustment is made and errors are marked in following cases :

- Different source addresses in Time Event and Clock Value

- b) No clock synchronization message received within time window
- c) Clock values not matched (different last_clk_val)

The handling of the Time Event, checking the source address of TE/CV DLPDU and the receive-delay-timer is part of the DLE.

With the reception of a valid Clock Value (SA is the same as the SA in the previously received Timer Event) , a DL service gives the received synchronization time and the receiver part of the communication delay to the DLS-user. The receiver part of the communication delay includes the time past since the receive-delay-timer was started with the reception of a TE DLPDU till read from the DLE after the reception of a CV DLPDU.

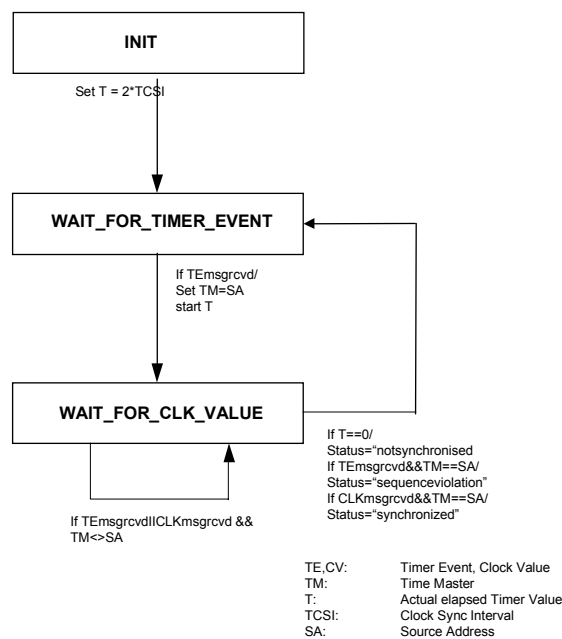


Figure 85 – Time receiver state machine

Figure 86 illustrates the Clock Synchronization sequences.

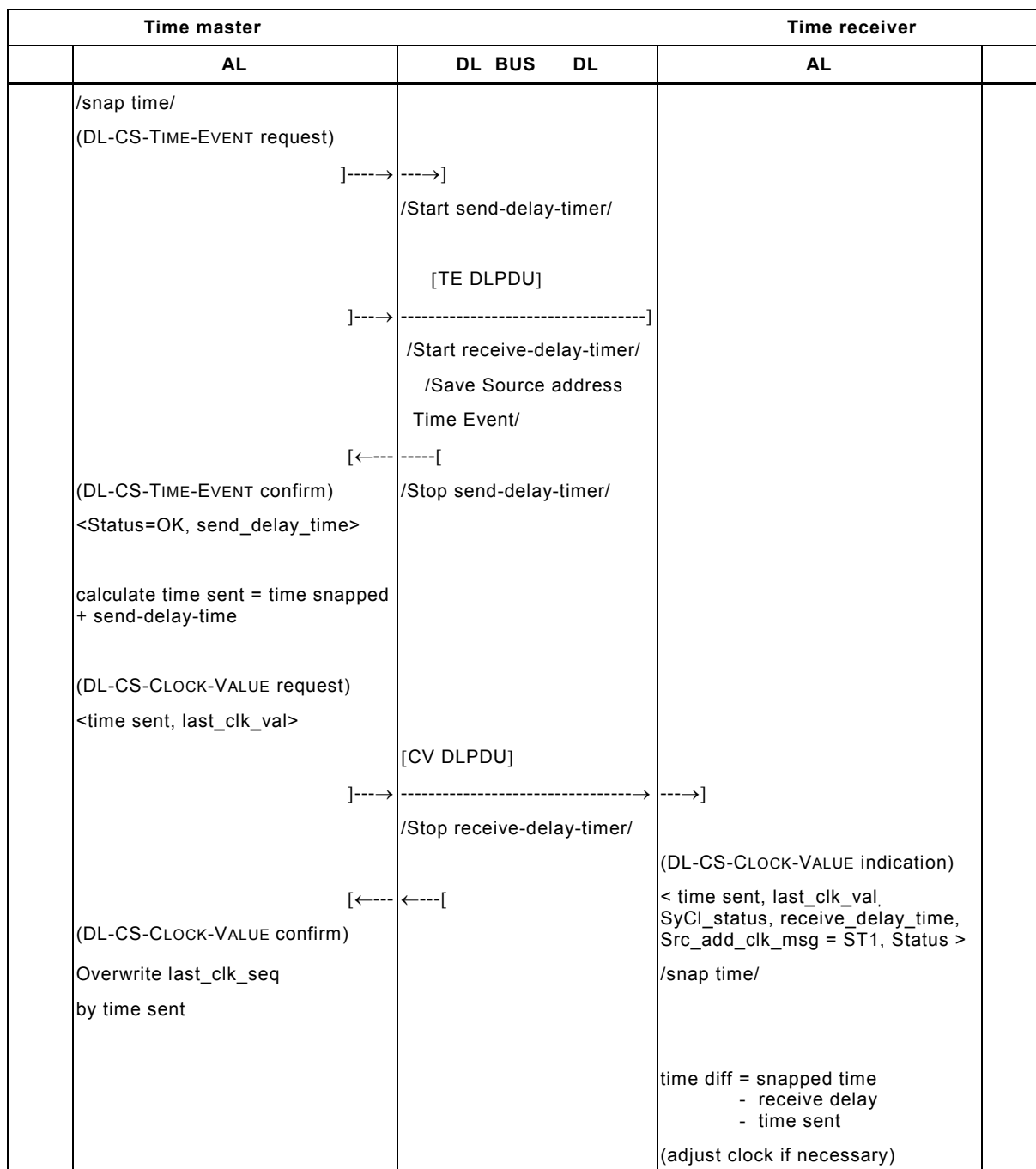


Figure 86 – Clock synchronization

24 Type 4: Data Link Protocol Definition

24.1 Overview of the DL-protocol

The DLL provides connectionless data transfer services for limited-size DLSDUs from one DLS-user to one or more (broadcast) DLS-users.

A DLE is implicitly connected to one PhE and to a single DLSAP. This means, that when a local DLS-user issues a service primitive at a certain DLSAP, the DLE and hence the Link is implicitly selected.

A DLE always delivers received DLSDUs at the same DLSAP, and hence to the same DLS-user.

This concept is illustrated in Figure 87.

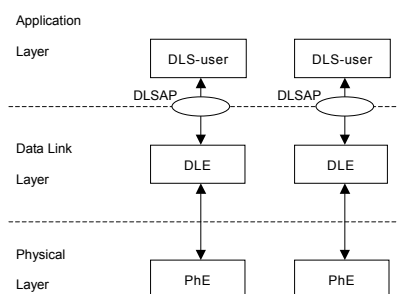


Figure 87 – Relationship of PhE, DLE and DLS-user

Each DLE has a Node-address. Node-addresses uniquely identify DLEs within the same Link.

A DL-route-element is an octet, which can hold a Node-address, or an address used by the DLS-user.

A Destination-DL-route holds a sequence of DL-route-elements, describing the complete route to the destination.

A Source-DL-route holds a sequence of DL-route-elements, describing the complete route back to the source.

A DL-route is defined as a Destination-DL-route and a Source-DL-route.

24.1.1 Functional classes

The functional class of a DLE determines its capabilities, and thus the complexity of conforming implementations. Two functional classes are defined:

Simple class, including only responder functionality (server).

Normal class, including initiator and responder functionality (client and server, also called peer).

24.1.2 Functions of the DLL

The functions of the DLL are those necessary to bridge the gap between the services available from the PhL and those offered to DLS-users. The functions are:

As a responder (in Simple class or Normal class DLEs):

- a) Receive a DLPDU from a remote DLE, perform frame check, parse the received DLPDU into its DL-protocol information and data components, and generate a DLS-user indication primitive. Possibly wait for a DLS-user request or response primitive, convert it to a DLPDU, and send that DLPDU to the remote DLE.
- b) Receive a single PhIDU specifying LINK-IDLE, and use that to time-out when waiting for a DLS-user request primitive.

As an initiator (in Normal class DLEs):

- c) Convert a DLS-user request primitive to a DLPDU, queue it, and send it to a remote DLE (or all DLEs at the Link if broadcast) at the first opportunity. Possibly wait for an Acknowledge or Immediate-reply DLPDU from the remote DLE, and (if an Immediate-reply DLPDU is received) generate a DLS-user indication primitive.
- d) Receive an SPDU, and use the associated data to check or gain Link-access synchronization.
- e) Receive a single PhIDU specifying LINK-IDLE, use that to keep Link-access synchronized, and possibly to initiate sending a DLPDU from the queue if the queue is not empty, or if the queue is empty, to send an SPDU for Link-access synchronization.

These functions are illustrated in Figure 88 to Figure 90.

24.1.2.1 Acknowledged vs. confirmed

The terms acknowledged and unacknowledged are used to describe whether the receiving DLE must acknowledge the receipt of a DLPDU or not. The terms confirmed and unconfirmed are used to describe whether the receiving DLS-user must confirm the receipt of a DLSDU or not.

The variable V(ACPDU) - Acknowledge Confirmed PDU - defines, whether the DLE must acknowledge the receipt of Confirmed DLPDUs. The variable V(AUPDU) - Acknowledge Unconfirmed PDU - defines, whether the DLE must acknowledge the receipt of Unconfirmed DLPDUs.

A special case is when the first Node-address in a received DLPDU is equal to the Broadcast-Node-address (BNA). In this case, the receiving DLE shall never acknowledge the receipt of the DLPDU.

24.1.2.2 Half-duplex and full duplex

Unless otherwise stated, the PhL is assumed to support half-duplex transfer. However, a PhL supporting full duplex is allowed.

Full duplex systems allow only two DLEs on a Link, both of Normal class. Each DLE is allowed to transmit immediately, that is, there is no Link Access system. DLEs supporting full duplex PhEs have separate state machines for receive and transmit, as illustrated in Figure 91 to Figure 92.

In full duplex systems, as well Confirmed as Unconfirmed DLPDUs are unacknowledged.

PhLs supporting full duplex shall not provide Link-Idle indications.

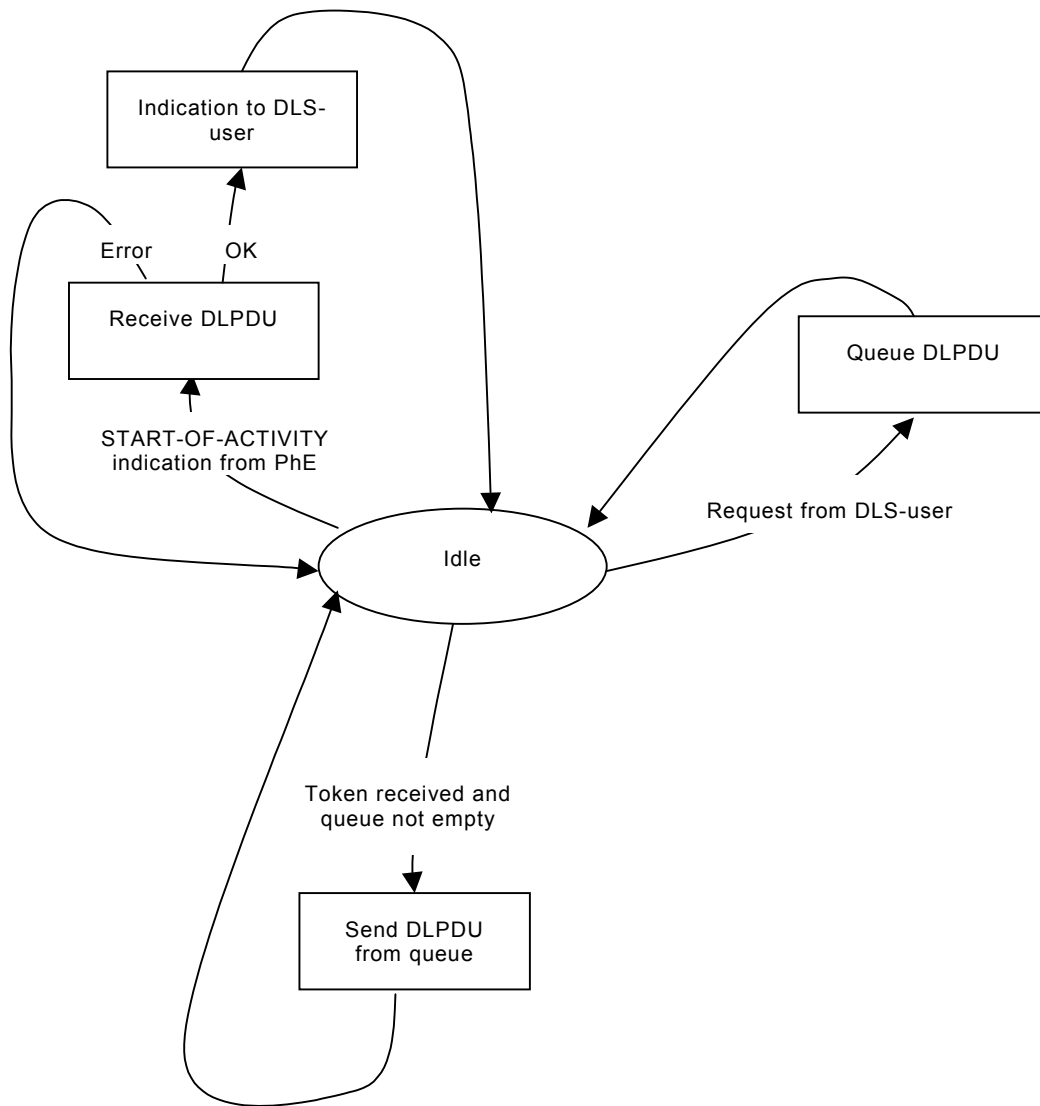


Figure 88 – DLE state diagram for confirmed and unconfirmed, unacknowledged DLPDUs

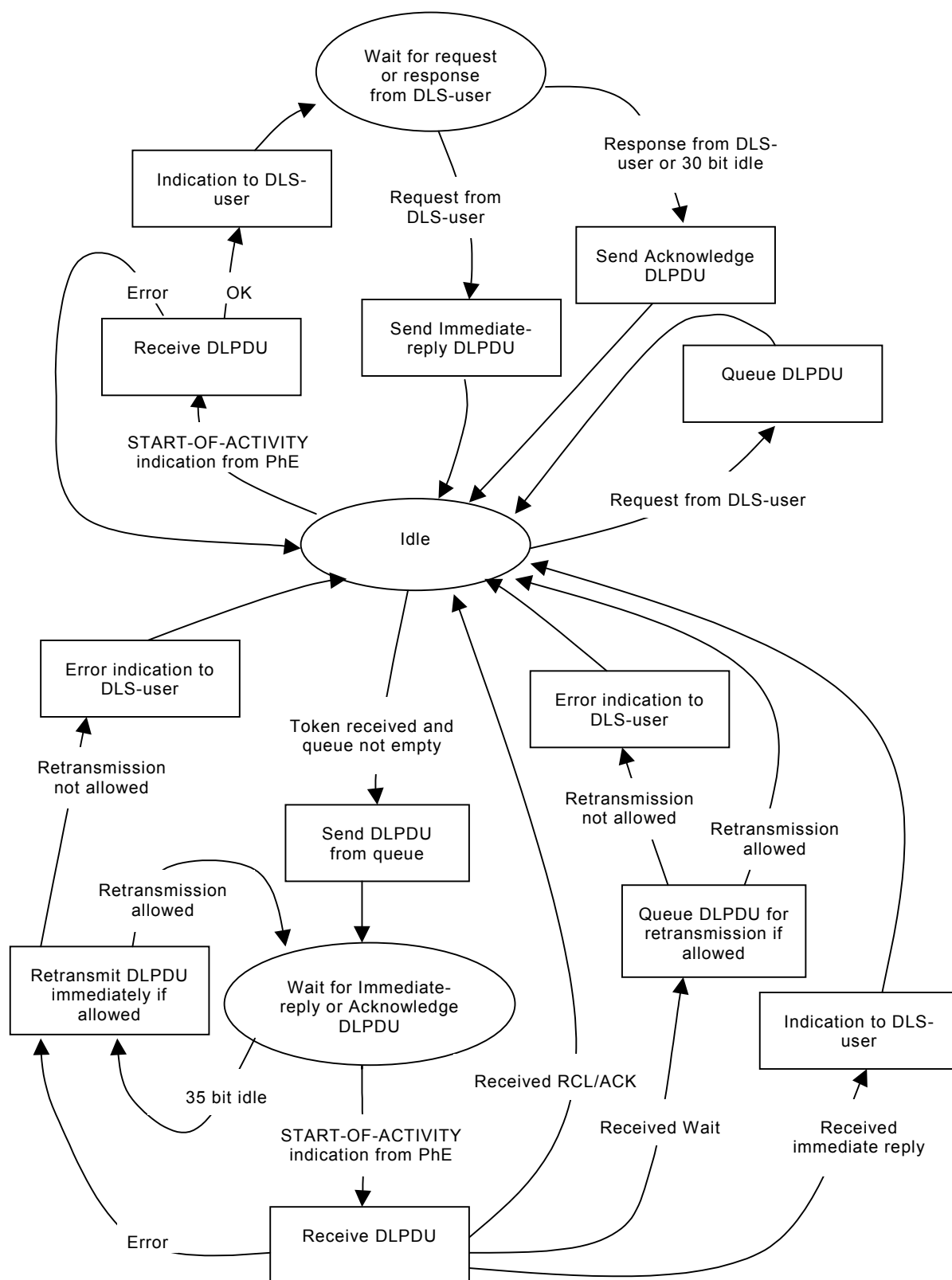


Figure 89 – DLE state diagram for confirmed acknowledged DLPDUs



Figure 90 – DLE state diagram for unconfirmed acknowledged DLPDUs

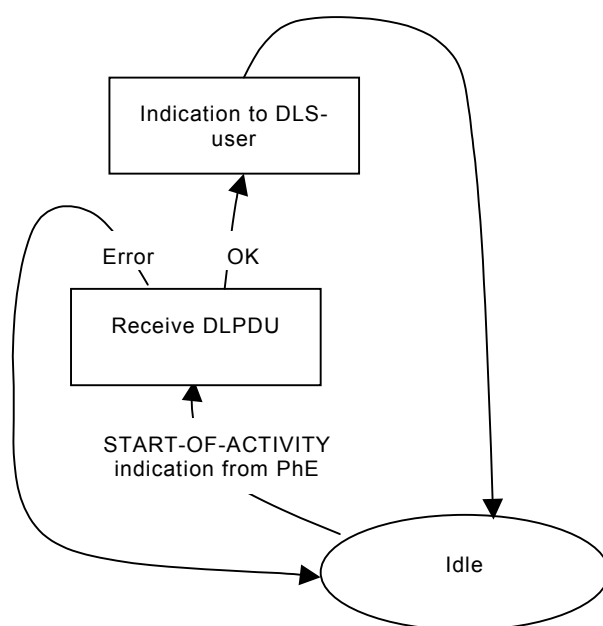


Figure 91 – Full duplex DLE receive state diagram

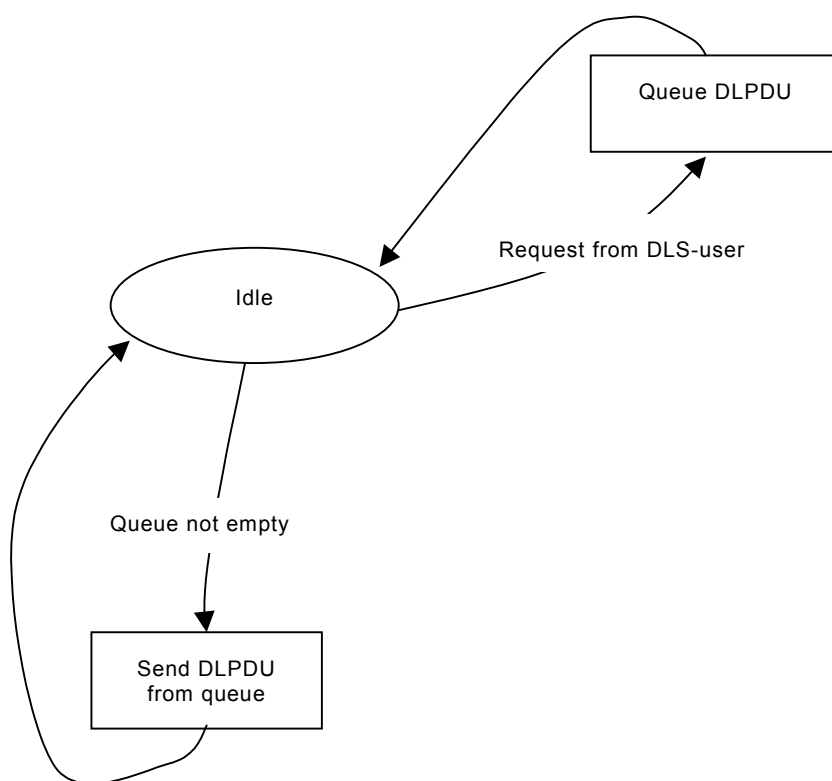


Figure 92 – Full duplex DLE transmit state diagram

24.1.2.3 DLPDU types

Four different types of DLPDUs are defined.

- a) Confirmed - used to send confirmed requests between DLS-users.
- b) Unconfirmed - used to send responses or unconfirmed requests between DLS-users.
- c) Acknowledge - used by DLEs to acknowledge receipt of Confirmed or Unconfirmed DLPDUs. The receipt of Acknowledge DLPDUs must never be acknowledged.
- d) Immediate-reply - used to send responses between DLS-users. The receipt of Immediate-reply DLPDUs must never be acknowledged.

24.1.2.4 SPDU types

Only one type of SPDU (Support Protocol Data Unit) is defined.

- a) Sync - used to send Link access synchronization information between DLEs. An SPDU holds the Node-address of the DLE holding the Virtual Link-access token. An SPDU can be "stand-alone" or part of an Acknowledge or Immediate-reply DLPDU.

24.1.2.5 Responder role, receiving a DLPDU from the PhE

This action includes a sequence of steps, as described in the following.

- a) Receive a single PhIDU specifying START-OF-ACTIVITY. This PhIDU holds a Node address. This address is examined to determine, whether its value is equal to the Node-address of this DLE, or equal to the Broadcast-Node-address (BNA) or the Service-Node-Address (SNA). If not, ignore this sequence and wait for the next PhIDU specifying START-OF-ACTIVITY.
- b) Receive a sequence of PhIDUs from the PhE, specifying DATA, concatenate them to a received DLPDU, compute a frame check sequence over the entire sequence of received data, and check for the proper value. If the value is not correct, ignore the DLPDU and wait for the next PhIDU specifying START-OF-ACTIVITY.
- c) Convert the received DLPDU into its DL-protocol control information and data components.
- d) Generate a DLS-user indication primitive.
- e) If the DLPDU received from the remote DLE is of type Confirmed, and the receipt of the DLPDU must be acknowledged, according to the rules described in 24.1.2.1, wait for a request or response primitive from the local DLS-user.

If no request or response primitive is issued from the local DLS-user in time (before a PhIDU specifying "LINK-IDLE for 30 bit periods" is received from the PhE), generate and immediately send an Acknowledge DLPDU. This DLPDU must specify "Wait" if this DLE is of Simple class, and "Response Comes Later / Acknowledge" ("RCL/ACK") if this DLE is of Normal class.

If a response primitive is issued from the local DLS-user in time, generate and immediately send an Acknowledge DLPDU, specifying "Wait" if this DLE is of Simple class, and "RCL/ACK" if this DLE is of Normal class.

If a request primitive is issued from the local DLS-user in time, convert it into an Immediate-reply DLPDU and send it immediately. After sending, wait for the next PhIDU specifying START-OF-ACTIVITY.

- f) If the DLPDU received from the remote DLE is of type Confirmed, and the receipt of the DLPDU shall not be acknowledged, wait for the next PhIDU specifying START-OF-ACTIVITY.
- g) If the DLPDU received from the remote DLE is of type Unconfirmed, and the receipt of the DLPDU must be acknowledged, according to the rules described in 24.1.2.1, generate and immediately send an Acknowledge DLPDU, specifying RCL/ACK. After sending, wait for the next PhIDU specifying START-OF-ACTIVITY.

- h) If the DLPDU received from the remote DLE is of type Unconfirmed, and the receipt of the DLPDU shall not be acknowledged, wait for the next PhIDU specifying START-OF-ACTIVITY.

24.1.2.6 Responder role, receiving a PhIDU specifying LINK-IDLE

As a responder, when waiting for a request or response primitive from the local DLS-user, the receipt of a PhIDU from the PhE specifying "LINK-IDLE for 30 bit periods" is used to timeout waiting for the DLS-user. The possible actions resulting from the timeout are defined in 24.1.2.5.

24.1.2.7 Initiator role, managing request primitives from the local DLS-user

This action includes a sequence of steps, as described in the following.

- a) Convert a request primitive from the local DLS-user into a DLPDU, queue it, and send it to a remote DLE (or all DLEs on the Link if broadcast) at the first opportunity.
- b) If the DLPDU sent is of type Unconfirmed, and the receiving DLE should acknowledge the receipt, according to the rules defined in 24.1.2.1, wait for an Acknowledge DLPDU from the remote DLE specifying RCL/ACK. If no acknowledge is received in time (before a PhIDU specifying "LINK-IDLE for 35 bit periods" is received from the PhE), immediately re-transmit the DLPDU if the permitted number of transmission retries have not been sent. If the permitted number of transmission retries have failed, do nothing, and this action is completed.
- c) If the DLPDU sent is of type Unconfirmed, and the receiving DLE should not acknowledge the receipt, this action is completed.
- d) If the DLPDU sent is of type Confirmed, and the receiving DLE should acknowledge the receipt, wait for an Immediate-reply DLPDU holding the response, or an Acknowledge DLPDU, from the remote DLE.

If an Acknowledge DLPDU is received from the remote DLE in time (before a PhIDU specifying "LINK-IDLE for 35 bit periods" is received from the PhE), and the acknowledge specifies "RCL/ACK", this action is completed. If the acknowledge specifies "Wait", queue the DLPDU for retransmission if the associated retry timer has not expired. If the retry timer has expired, generate a DLS-user indication primitive with the appropriate error information.

If an Immediate-reply DLPDU holding the response is received in time from the remote DLE, convert the received DLPDU into its DL-protocol control information and data components, and generate a DLS-user indication primitive.

If neither acknowledge nor response is received from the remote DLE in time, re-transmit the DLPDU immediately (while this DLE still holds the Virtual Link-access token) if the permitted number of transmission retries have not been sent. If the permitted number of transmission retries have failed, generate a DLS-user indication primitive with the appropriate error information.

- e) If the DLPDU sent is of type Confirmed, and the receiving DLE should not acknowledge the receipt, this action is completed.

24.1.2.8 Initiator role, Link-access

The Link-access system is based on a so-called Virtual Link-access token. Virtual because the token is not explicitly sent from one Normal class DLE to another, but implicitly passed as the Link is idle.

The following DLE variables and counters are used by the Link-access system.

V(NA) - Node-address. Each DLE on a Link is uniquely identified by its Node-address, the value of which is stored in V(NA). The value of V(NA) must be different in all DLEs on the Link.

V(NDLE) - Number of DLEs - holds the maximum number of Normal class DLEs on the Link. The value of V(NA) must be lower than or equal to the value of V(NDLE). The value of V(NDLE) must not exceed 32. The value of V(NDLE) must be the same in all DLEs on the Link.

C(LAC) - Link Access Counter - holds the Node-address of the DLE holding the Virtual Link-access token. The value of C(LAC) will be the same in all DLEs on the Link.

C(LIC) - Link Idle Counter - holds information on, for how long the Link has been idle. The value of C(LIC) will be the same in all DLEs on the Link.

Figure 93 illustrates the functionality of the Link-access system. The "Action" line describes the use of the Link. The first action is that the DLE having Node-address 2 sends a Confirmed DLPDU, and receives the corresponding Immediate-reply DLPDU. The second action is that the DLE having Node-address 3 sends an Unconfirmed DLPDU. Then, after a long idle period, the DLE with Node-address 2 sends a Sync SPDU.

The DLE having Node-address 4 is not present. Had it been present, DLE4 should have sent the Sync SPDU, as the Link had been idle for 360 bit periods when it "received" the Virtual Link-access token. The next DLE holding the token is DLE1, which is present and therefore sends the Sync SPDU.

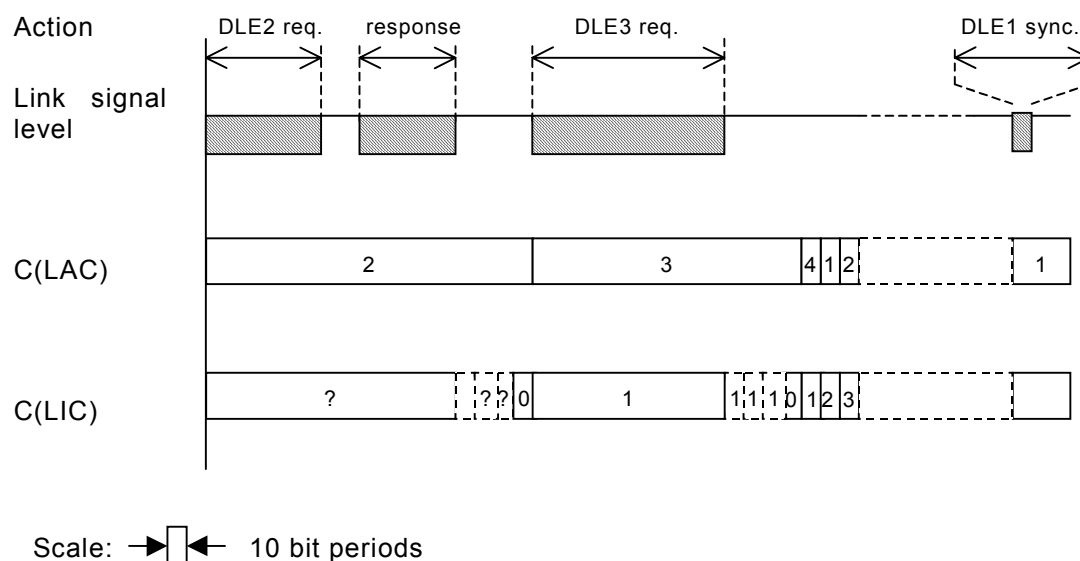


Figure 93 – Link access example

Each single PhIDU specifying LINK-IDLE holds information on, whether the Link has been idle for 30 bit periods, for 35 bit periods, or for 40 or more bit periods in the associated status parameter.

Each time a LINK-IDLE specifying that the Link has been idle for 40 or more bit periods is received, the value of C(LAC) - Link Access Counter - and the value of C(LIC) - Link Idle Counter - is incremented by 1. When the value of C(LAC) becomes higher than the value of V(NDLE) the value of C(LAC) is set to 1.

Each time a LINK-IDLE specifying that the Link has been idle for 30 bit periods is received, the value of C(LIC) is set to 0.

If, immediately after incrementing C(LAC), the value of C(LAC) is equal to the Node-address of this DLE, it means this DLE holds the Virtual token, and therefore is allowed to send (and possibly re-transmit) a DLPDU from the queue. This must be initiated immediately, by sending a START-OF-ACTIVITY-2 to the PhE. It is a task of the implementation to ensure that the transmission is initiated within 7 bit-periods after receipt of the LINK-IDLE service primitive. If the queue is empty, the DLE must check the value of C(LIC), to see for how long time the Link has been idle. If the value of C(LIC) is equal to or higher than 33, it means the Link has been idle for 360 bit periods or more. If this applies, the DLE must send a Sync SPDU for Link-access synchronization. This must be done immediately, by sending a START-OF-ACTIVITY-2 to the PhE. The associated data field must specify Source, and hold the Node-address of this DLE. This system is used to keep the idle counters in all PhEs on the Link, and thus the values of C(LAC) and C(LIC) in all DLEs on the Link, synchronized.

Each single PhIDU specifying START-OF-ACTIVITY holds a Node-address and a Source/Destination designator in the associated data field. If the Node-address is a source Node-address, it identifies the DLE holding the Virtual Link-access token at this moment. Such a PhIDU forms a complete Sync SPDU.

When the DLE receives a Sync SPDU, it must compare the received Node-address with the value of C(LAC). If the 2 values are equal, it means the DLE is synchronized to the other DLEs on the Link. If they are not equal, it means the DLE is out of synchronization. As long as the DLE is out of synchronization, it is only allowed to act as a responder. 24.6 describes how to gain Link-access synchronization again.

24.1.3 Service assumed from the PhL

This subclause defines the assumed Physical Service (PhS) primitives and their constraints on use by the DLE.

24.1.3.1 Assumed primitives of the PhS

The granularity of transmission in the fieldbus protocol is one octet. This is the granularity of PhS-user data exchanged at the PhL - DLL interface.

24.1.3.1.1 PhS management services

The PhS is assumed to provide the following service primitives to get and set PhE parameter values:

- a) Ph-SETVALUE request (parameter name, new value)
- b) Ph-SETVALUE confirm (status)
- c) Ph-GETVALUE request (parameter name)
- d) Ph-GETVALUE confirm (current value)

These services are used by the DLE to

- 1) set the bit rate, as a result of bit rate changing through DL-management;
- 2) get the bit rate, as a result of bit rate or Max Indication Delay reading through DL-management. The value of Max Indication Delay is calculated from the current value of bit rate, and shall indicate a value corresponding to 30 bit periods.

24.1.3.1.2 PhS transmission and reception services

The PhS is assumed to provide the following service primitives for transmission and reception:

- a) Ph-DATA request (class, data)
- b) Ph-DATA indication (class, data, status)
- c) Ph-DATA confirm (status)

where

class — specifies the Ph-interface-control-information (PhICI) component of the Ph-interface-data-unit (PhIDU).

For a Ph-DATA request, its possible values are

START-OF-ACTIVITY-1 — the PhE shall enable its driver, and initiate transmission by transmitting the associated data parameter as an "Address character". The PhE shall do this immediately, though not until the value of the PhE's idle counter has reached 11.

START-OF-ACTIVITY-2 — the PhE shall enable its driver, and initiate transmission by transmitting the associated data parameter as an "Address character". The PhE shall do this immediately, though not until the value of the PhE's idle counter modulus 10 has reached 2.

DATA — the PhE shall transmit the associated data parameter as a "Data character" immediately.

END-OF-ACTIVITY — the PhE shall wait till transmission of all formerly received data from the DLE has finished, and then disable its driver. The associated data parameter shall not be transmitted.

For a Ph-DATA indication, its possible values are

START-OF-ACTIVITY — the PhE has received an "Address character", the value of which is reported in the associated data parameter. The associated status parameter specifies success or the locally detected reason for failure.

DATA — the PhE has received a "Data character", the value of which is reported in the associated data parameter. The associated status parameter specifies success or the locally detected reason for failure.

LINK-IDLE — the PhE has detected, that the signal level on the Link has been "Idle" for 30, 35, 40, 50, 60... bit periods. The associated status parameter specifies if the Link has been idle for 30 bit periods, for 35 bit periods, or for 40 or more bit periods.

NOTE The PhE holds an idle counter. This counter is incremented by one each time the signal level on the Link has been idle for one bit period. Each time the signal level is not idle, the idle counter is cleared. When the idle counter reaches 30, the PhE reports this with a Ph-DATA indication of class LINK-IDLE, and associated status indicating 30 bit periods. Five bit periods later, if the Link is still idle, the PhE reports this with another Ph-DATA indication of class LINK-IDLE, and associated status indicating 35 bit periods. Five bit periods later, if the Link is still idle, the PhE reports this with another Ph-DATA indication of class LINK-IDLE, and associated status indicating 40 or more bit periods. This goes on for each 10 bit period with indications specifying 40 or more bit periods, till the signal level on the Link is no longer idle.

data — specifies the Ph-interface-data (PhID) component of the PhIDU. It consists of one octet of Ph-user data to be transmitted (Ph-DATA request), or one octet of Ph-user data which was received (Ph-DATA indication).

status — specifies either success or the locally detected reason for failure, or specifies if

the associated LINK-IDLE indication indicates "30", "35" or "40 or more" bit periods of idle after Link activity.

The Ph-DATA confirm primitive provides the feedback necessary to enable the DLE to report failures such as Link short-circuit or noise resulting in framing error to the DLS-user, and provides the critical physical timing necessary to prevent the DLE from starting a second transmission before the first is complete.

24.1.3.2 Transmission of Ph-user data

When a DLE has a DLPDU to transmit, and the Link-access system gives that DLE the right to transmit, then the DLE shall send the DLPDU, including a concatenated FCS. Making a sequence of Ph-DATA requests as follows does this:

- a) The first request shall specify START-OF-ACTIVITY-11 if the DLPDU to transmit is an Acknowledge or Immediate-reply DLPDU, or if the transmission is an immediate re-transmission of a Confirmed or Unconfirmed DLPDU. The first request shall specify START-OF-ACTIVITY-2 if transmission of a Confirmed or Unconfirmed DLPDU from the queue is commenced.
- b) This first request shall be followed by consecutive requests specifying DATA, and concluded by a single request specifying END-OF-ACTIVITY.

The PhE signals its completion of each Ph-DATA request, and its readiness to accept a new Ph-DATA request, with a Ph-DATA confirm primitive. The status parameter of the Ph-DATA confirm primitive conveys the success or failure of the associated Ph-DATA request.

24.1.3.3 Reception of Ph-user data

The PhE reports a received transmission with Ph-DATA indications, which shall consist of either a single indication specifying START-OF-ACTIVITY, or a single indication specifying START-OF-ACTIVITY followed by consecutive indications specifying data. Each indication has an associated status parameter, specifying successful reception of the associated data, or the locally detected reason for failure.

24.2 General structure and encoding of PhIDUs and DLPDUs, and related elements of procedure

24.2.1 PhIDU structure and encoding

Each PhIDU consists of Ph-interface-control-information and in some cases one octet of Ph-interface-data (see 24.1.3). When the DLE transmits a DLPDU, it computes a frame check sequence for the DLPDU as specified in 24.2.2, concatenates the DLPDU and the frame check sequence, and transmits the concatenated pair as a sequence of PhIDUs as follows:

- a) The DLE issues a single Ph-DATA request primitive with PhICI specifying START-OF-ACTIVITY-2 if sending from the queue, and specifying START-OF-ACTIVITY-11 if sending an Acknowledge or Immediate-reply DLPDU, or if re-transmitting because of missing acknowledge. The request primitive is accompanied by one octet holding the first octet from the DLPDU as Ph-interface-data. After that, the DLE awaits the consequent Ph-DATA confirm primitive.
- b) The DLE issues a sequence of Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the DLPDU as Ph-interface-data, from second to last octet of the DLPDU, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.
- c) If the value of V(FCM) - FrameCheckMethod - specifies reduced frame check, the DLE issues a single Ph-DATA request primitive with PhICI specifying DATA, accompanied by one octet holding the computed FCS as Ph-interface-data, and after the Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive. If the value of V(FCM) - FrameCheckMethod - specifies normal frame check, the DLE issues a sequence of Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the

FCS as Ph-interface-data, from first to last octet of the FCS, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.

- d) The DLE issues a single Ph-DATA request primitive with PhICI specifying END-OF-ACTIVITY, and awaits the consequent Ph-DATA confirm primitive.

It is a task of the implementation to ensure that there are no idle periods between the octets of a transmitted DLPDU.

The DLE forms a received DLPDU by concatenating the sequence of octets received as Ph-interface-data of consecutive Ph-DATA indications, computing a frame check sequence for those received octets as specified in 24.2.2, and compares the received FCS value with the computed, as follows:

- 1) The DLE received a single Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY, accompanied by one octet of the received DLPDU as Ph-interface-data, and initializes its computation of an FCS for the received DLPDU.
- 2) The DLE receives a sequence of Ph-DATA indication primitives with PhICI specifying DATA, each accompanied by one octet of the received DLPDU as Ph-interface-data, incrementally computes an FCS on the received octet, and concatenates all but the last one or two (depending on V(FCM) of those received octets to form the received DLPDU. During reception, the DLE encodes the DLPDU being received to compute the number of octets forming the DLPDU.
- 3) When the DLE has received the last Ph-DATA indication, it compares the value(s) of the computed FCS to zero:
 - i) If the value(s) is (are) zero, then the DLE reports the reconstructed DLPDU as a correctly received DLPDU suitable for further analysis.
 - ii) If the value(s) is (are) not zero, the DLE ignores the received DLPDU, and performs no further actions related to the received DLPDU.

24.2.2 Frame check sequence

The value of the DLE local variable V(FCM) determines which frame check method to use.

Two different frame check methods are defined, "Normal" and "Reduced".

24.2.2.1 Normal frame check method

The "Normal" frame check method uses two frame-check codes, FCA and FCB. The method gives a "Hamming Distance" of 4 for a codeword size of 64 bits. This means, that up to three (Hamming Distance minus one) randomly-located error bits within a 64-bit wide window will be detected. Any burst of errors up to 15 bits in length and any error with an odd number of bits in error will be detected.

At the transmitting DLE, the following sequence is followed:

- a) Before transmitting the first octet of the DLPDU, clear the two variables FCA and FCB.
- b) For each octet of the DLPDU to be sent, exclusive OR the value of the octet to be sent to the value of FCA, and store the result in FCA. Exclusive OR the value of the octet to be sent to the value of FCB, rotate the result one bit left, and store the final result in FCB. This is done in the order in which the octets are sent.
- c) When the last octet of the DLPDU has been sent, send the value of FCA, exclusive OR the value of FCA to the value of FCB, rotate the result one bit left, and store the final result in FCB.
- d) When FCA has been sent, send the value of FCB.

At the receiving DLE, the following sequence is followed:

- e) Before receiving the first octet of the DLPDU, clear the two variables FCA and FCB.

- f) For each octet of the DLPDU received, exclusive OR the value of the received octet to the value of FCA, and store the result in FCA. Exclusive OR the value of the received octet to the value of FCB, rotate the result one bit left, and store the final result in FCB. This must be done in the order in which the octets are received.
- g) When the first octet of the FCS has been received, and the normal frame check computation performed, check that the value of FCA is equal to zero.
- h) When the last octet of the FCS has been received, and the normal frame check computation performed, check that the value of FCB is equal to zero.

24.2.2.2 Reduced frame check method

The "Reduced" frame check method uses one frame-check code, FC. The method gives a "Hamming Distance" of 2 for the whole DLPDU, which means any single error bit will be detected. A single error burst up to 8 bits in length will also be detected.

At the transmitting DLE, the following sequence is followed:

- a) Before transmitting the first octet of the DLPDU, clear the variable FC.
- b) For each octet of the DLPDU to be sent, add the value of the octet to be sent to the value of FC, without carry, and store the result in FC.
- c) When the last octet of the DLPDU has been sent, send the 2's complement of the value of FC.

At the receiving DLE, the following sequence is followed:

- d) Before receiving the first octet of the DLPDU, clear the variable FC.
- e) For each octet of the DLPDU received, add the value of the received octet to the value of FC without carry, and store the result in FC.
- f) When the FCS has been received, and the normal frame check computation performed, check that the value of FC is equal to zero.

24.2.3 Common DLPDU structure, encoding and elements of procedure

Each DLPDU consists of a P-NET-route field, a Control-status field, a Data-field-format field, and for most DLPDUs a Data field. An FCS field (see 24.2.2) which is used to check the integrity of the received DLPDU is appended before transmission, and removed after reception.

24.2.3.1 P-NET-route field

The first field in each DLPDU is a P-NET-route field. The P-NET-route field holds a P-NET-route and consists of 2-30 octets, called P-NET-route-elements. Each P-NET-route-element is an octet, holding a 7-bit DL-route-element or Remaining-route-length, and a 1-bit Source/Destination designator. Four different P-NET-route field formats are defined: "Simple", "Extended", "Complex" and "Immediate". The P-NET-route field format is indicated by the sequence of Source/Destination designators.

The Source/Destination designator is physically located as bit 8 in the octet. A value of "0" designates "Destination", and a value of "1" designates "Source".

24.2.3.1.1 Simple P-NET-route format

P-NET-route fields of Simple format consist of one destination P-NET-route-element followed by one source P-NET-route-element, as illustrated in Figure 94.

0	Destination address
1	Source address

Figure 94 – Simple P-NET-route format

The Destination address identifies the DLE to receive the DLPDU. The Source address identifies the transmitting DLE.

Simple routes are used when sending Confirmed or Unconfirmed DLPDUs holding requests to DLEs of simple class. The DLPDU is of type Unconfirmed if the Destination address is equal to the Broadcast-Node-Address (BNA).

24.2.3.1.2 Extended P-NET-route format

P-NET-route fields of Extended format consist of two destination P-NET-route-elements followed by two source P-NET-route-elements, as illustrated in Figure 95.

0	Destination address
0	Destination address
1	Source address
1	Source address

Figure 95 – Extended P-NET-route format

The first Destination address identifies the DLE to receive the DLPDU. The second Destination address is used by the DLS-user. The first Source address identifies the transmitting DLE. The second Source address is used by the DLS-user.

Extended routes are used when sending Confirmed or Unconfirmed DLPDUs holding requests to DLEs of normal class. The DLPDU is Unconfirmed if the value of the first Destination address equals BNA.

24.2.3.1.3 Complex P-NET-route format

P-NET-route fields of Complex format consist of more than 2 destination P-NET-route-elements followed by 2 or more source P-NET-route-elements, as illustrated in Figure 96.

0	Destination address
0	Destination address
0	Remaining-route-length
.....	
1	Source address
1	Source address

Figure 96 – Complex P-NET-route format

The first Destination address identifies the DLE to receive the DLPDU. The remaining Destination addresses are used by the DLS-user. The third P-NET-route-element holds the number of P-NET-route-elements following the third P-NET-route-element. The first Source address identifies the transmitting DLE. The remaining source addresses (maybe except the last) are used by the DLS-user.

Complex routes are used when sending Confirmed or Unconfirmed DLPDUs holding requests or Unconfirmed DLPDUs holding responses to DLEs of normal class. The DLPDU is Unconfirmed if the value of the last Source address equals 0, or the value of one of the Destination addresses equals BNA.

24.2.3.1.4 Immediate P-NET-route format

P-NET-route fields of Immediate format consist of one source P-NET-route-element followed by one destination P-NET-route-element, as illustrated in Figure 97.

1	Source address
0	Destination address

Figure 97 – Immediate P-NET-route format

The Source address identifies the DLE to receive the DLPDU. The Destination address identifies the transmitting DLE. Note that the Source address does NOT identify the transmitting DLE, but the DLE that transmitted the request resulting in this DLPDU. As always, the first octet in the P-NET-route identifies the DLE to receive the DLPDU.

Immediate routes are used when sending Acknowledge or Immediate-reply DLPDUs to DLEs of normal class.

24.2.3.2 Control-status field

The second field in each DLPDU is a Control-status field. This field consists of 1 octet, used by the DLE in conjunction with the P-NET-route field format and the Data-field-format field to determine the DLPDU type. If the DLPDU type is Acknowledge, the Control-status field holds status information to the DLE. The coding of the Control-status field is illustrated in Figure 98.

If the format of the P-NET-route-field is Immediate, the DLPDU is of type Immediate-reply or Acknowledge. The value of the Control-status field indicates (in conjunction with the Data-field-format field) whether the DLPDU is of type Immediate-reply or Acknowledge, according to the following:

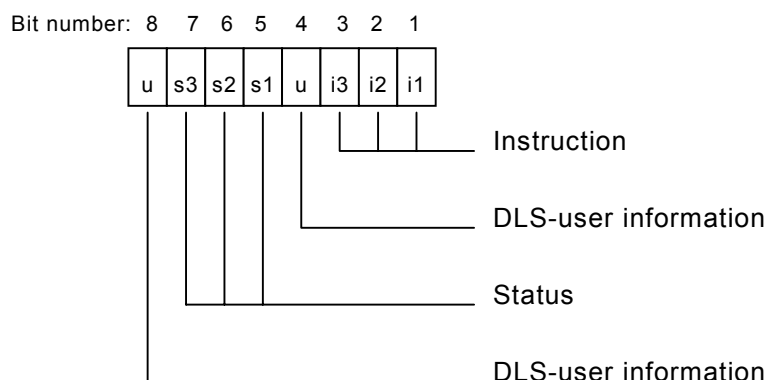


Figure 98 – Control-status format

The range for the Instruction subfield is 0 to 7. The range for the Status subfield is 0 to 7.

The DLPDU is an Acknowledge DLPDU specifying Wait if all of these conditions are fulfilled:

- a) the P-NET-route-field format is immediate
- b) the value of the Instruction subfield is greater than 0
- c) the value of the Data-size subfield in the Data-field-format field equals 0
- d) the value of the Status subfield is 4

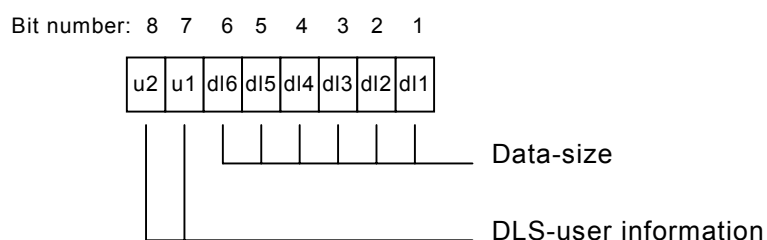
The DLPDU is an Acknowledge DLPDU specifying RCL/ACK if all of these conditions are fulfilled:

- e) the P-NET-route-field format is immediate
- f) the value of the Instruction subfield is greater than 0
- g) the value of the Data-size subfield in the Data-field-format field equals 0
- h) the value of the Status subfield is 5

If neither of these two apply, the DLPDU is not an Acknowledge DLPDU. Acknowledge DLPDUs hold information for the DLE, and the receipt of an Acknowledge DLPDU does not result in a DLS-user indication.

24.2.3.3 Data-field-format field

The third field in each DLPDU is a Data-field-format field. This field consists of 1 octet, holding a 6-bit subfield indicating Data-size, and a 2-bit subfield for DLS-user information, as illustrated in Figure 99.

**Figure 99 – Data-field-format**

The range for the Data-size subfield is 0 to 63. The value indicates the number of octets in the Data-field of the DLPDU, and does not include the FCS octet(s).

24.2.3.4 Data field

The next field in the DLPDU is the Data field. The size and DLS-user interpretation of this field is indicated in the Data-field-format and Control-status fields. The size of the Data field is 0 to 63 octets.

24.3 DLPDU-specific structure, encoding and elements of procedure

Table 199 – Summary structure of DLPDUs

DLPDU type	P-NET-route format	Destination node-addresses	Last source node-address	Control-status	Data size	Data
Confirmed	Simple	≠ BNA	≠ 0	Any	> 2	user data
Confirmed	Extended	≠ BNA	≠ 0	Any	> 2	user data
Confirmed	Complex	≠ BNA	≠ 0	Any	> 2	user data
Unconfirmed	Simple	= BNA	≠ 0	Any	> 2	user data
Unconfirmed	Extended	= BNA	≠ 0	Any	> 2	user data
Unconfirmed	Complex	= BNA	≠ 0	Any	> 2	user data
Unconfirmed	Complex	≠ BNA	= 0	Any	≥ 0	user data
Immediate-reply	Immediate	Any	Any	Any	≥ 0	user data
Acknowledge	Immediate	Any	Any	= Wait / RCL/ACK	= 0	-

The DLPDU type is indicated by the P-NET-route format, the contents of the Destination Node-addresses in the P-NET-route, the contents of the last Source address in the P-NET-route, the contents of Control-status, and the contents of the data size subfield of Data-field-format, as shown in Table 199.

When the value in the column Destination Node-addresses is "≠ BNA" it means, that none of the Node-addresses in the P-NET-route field are = BNA. When the value in the column Destination Node-addresses is "= BNA" it means, that at least one of the Node-addresses in the P-NET-route field is = BNA.

24.3.1 Confirmed DLPDU

A Confirmed DLPDU is used

- a) to request the transfer of a limited amount of transparent user data from another DLS-user to the requesting DLS-user;

- b) to transfer a limited amount of transparent user data from the requesting DLS-user to another DLS-user;
- c) to transfer a limited amount of transparent user data from the requesting DLS-user to another DLS-user, and at the same time request the transfer of the same amount of transparent user data from that other DLS-user to the requesting.

24.3.1.1 Structure of Confirmed DLPDUs

The Structure of Confirmed DLPDUs is shown in Table 200.

Table 200 – Structure of Confirmed DLPDUs

P-NET-route format	Destination Node-addresses	Last source node-address	Control-status	Data size	Data
Simple	≠ BNA	≠ 0	DLS-user info	> 2	user data
Extended	≠ BNA	≠ 0	DLS-user info	> 2	user data
Complex	≠ BNA	≠ 0	DLS-user info	> 2	user data

24.3.1.2 Sending the Confirmed DLPDU

A Confirmed DLPDU is selected for transmission on the Link when the DLPDU is the first in the queue, and the DLE receives the Virtual Link-access token. Once selected, the DLPDU is removed from the queue, and transmission of the DLPDU commences. If the receipt of the DLPDU must be acknowledged, according to the rules described in 24.1.2.1, the DLPDU shall be transmitted until either

- a) an Immediate-reply DLPDU is received, or
- b) an Acknowledge DLPDU is received, or
- c) the original transmission and the permitted maximum number or transmission retries, V(MRC), have all failed to elicit one of the permissible reply DLPDUs.

In addition to the above, the transmitting DLE shall act according to the rules described in 24.1.2.7.

24.3.1.3 Receiving the Confirmed DLPDU

A received Confirmed DLPDU shall be treated as follows by the receiving DLE:

NOTE The next alternative is used to detect the reception of a duplicated Confirmed DLPDU resulting from an immediate retry by the current token-holding DLE, which itself was probably caused by an error detected during receipt of the earlier Acknowledge or Immediate-reply DLPDU.

- a)
 - 1) the receipt of the DLPDU shall be acknowledged, according to the rules described in 24.1.2.1, and
 - 2) no PhE LINK-IDLE indication primitive, specifying the Link has been idle for 40 or more bit periods, has been received since the last DLPDU was received, and
 - 3) the contents of the P-NET-route field in the just received DLPDU is exactly the same as the contents of the P-NET-route field in the last DLPDU received

then the receiving DLE shall

- i) retransmit the prior-transmitted Acknowledge or Immediate-reply DLPDU immediately, and
- ii) discard the received DLPDU and not forward it to the DLS-user.

- b) If a) does not apply, the receiving DLE shall act according to the rules described in 24.1.2.5

24.3.2 Unconfirmed DLPDU

An Unconfirmed DLPDU is used

- a) by a requesting DLS-user to transfer a limited amount of transparent user data from the requesting DLS-user to one or more other DLS-users;
- b) by a responding DLS-user to transfer a limited amount of transparent user data from the responding DLS-user to the requesting, as a response to a received confirmed DLPDU;
- c) by a responding DLS-user to acknowledge the receipt of a limited amount of transparent user data from the requesting DLS-user, as a response to a received confirmed DLPDU.

24.3.2.1 Structure of Unconfirmed DLPDUs

The Structure of Unconfirmed DLPDUs is shown in Table 201.

Table 201 – Structure of Unconfirmed DLPDUs

P-NET-route format	Destination node-addresses	Last source node-address	Control-status	Data size	Data
Simple	= BNA	≠ 0	DLS-user info	> 2	user data
Extended	= BNA	≠ 0	DLS-user info	> 2	user data
Complex	= BNA	≠ 0	DLS-user info	> 2	user data
Complex	≠ BNA	= 0	DLS-user info	≥ 0	user data

24.3.2.2 Sending the Unconfirmed DLPDU

An unconfirmed DLPDU is selected for transmission on the Link when the DLPDU is the first in the queue, and the DLE receives the Virtual Link-access token. Once selected, the DLPDU is removed from the queue, and transmission of the DLPDU commences. If the receipt of the DLPDU must be acknowledged, according to the rules described in 24.1.2.1, the DLPDU shall be transmitted until either

- a) an Acknowledge DLPDU is received, or
- b) the original transmission and the permitted maximum number or transmission retries, V(MRC), have all failed to elicit one of the permissible reply DLPDUs.

In addition to the above, the transmitting DLE shall act according to the rules described in 24.1.2.7.

24.3.2.3 Receiving the Unconfirmed DLPDU

A received Unconfirmed DLPDU shall be treated as follows by the receiving DLE:

NOTE The next alternative is used to detect the reception of a duplicated Unconfirmed DLPDU resulting from an immediate retry by the current token-holding DLE, which itself was probably caused by an error detected during receipt of the earlier Acknowledge or Immediate-reply DLPDU.

- a) If
 - 1) the receipt of the DLPDU shall be acknowledged, according to the rules described in 24.1.2.1, and
 - 2) no PhE LINK-IDLE indication primitive, specifying the Link has been idle for 40 or more bit periods, has been received since the last DLPDU was received, and
 - 3) the contents of the P-NET-route field in the just received DLPDU is exactly the same as the contents of the P-NET-route field in the last DLPDU received

then the receiving DLE shall

- i) retransmit the prior-transmitted Acknowledge DLPDU immediately, and
 - ii) discard the received DLPDU and not forward it to the DLS-user.
- b) If a) does not apply, the receiving DLE shall act according to the rules described in 24.1.2.5

24.3.3 Acknowledge DLPDU

An Acknowledge DLPDU is used

- a) by a responding DLE to acknowledge the receipt of a Confirmed or Unconfirmed DLPDU;
- b) by a responding DLS-user to acknowledge the receipt of a Confirmed DLPDU (the Acknowledge DLPDU is transmitted by the DLE as a result of a request service primitive from the local DLS-user specifying DLSDU type acknowledge).

24.3.3.1 Structure of Acknowledge DLPDUs

The Structure of Acknowledge DLPDUs is shown in Table 202.

Table 202 – Structure of Acknowledge DLPDU

P-NET-route format	Destination node-addresses	Last source node-address	Control-status	Data size	Data
Immediate	Any	Any	= Wait/RCL/ACK	= 0	-

24.3.3.2 Sending the Acknowledge DLPDU

An Acknowledge DLPDU is transmitted as an immediate reply on the Link to acknowledge the receipt of a Confirmed or Unconfirmed DLPDU, according to the rules described in 24.1.2.5. Acknowledge DLPDUs shall never be retransmitted.

24.3.3.3 Receiving the Acknowledge DLPDU

A received Acknowledge DLPDU shall be treated according to the rules described in 24.1.2.7. The receipt of Acknowledge DLPDUs shall never be acknowledged.

24.3.4 Immediate-reply DLPDU

An Immediate-reply DLPDU is used

- a) by a responding DLS-user to transfer a limited amount of transparent user data from the responding DLS-user to the requesting, as a response to a received confirmed DLPDU;
- b) by a responding DLS-user to confirm the receipt of a limited amount of transparent user data from the requesting DLS-user, as a response to a received confirmed DLPDU.

24.3.4.1 Structure of Immediate-reply DLPDU

Table 203 – Structure of Immediate-reply DLPDU

P-NET-route format	Destination node-addresses	Last source node-address	Control-status	Data size	Data
Immediate	Any	Any	≠ Wait/RCL/ACK	≥ 0	user-data

24.3.4.2 Sending the Immediate-reply DLPDU

An Immediate-reply DLPDU is transmitted as an immediate reply on the Link to acknowledge the receipt of a Confirmed DLPDU, according to the rules described in 24.1.2.5. Immediate-reply DLPDUs shall never be retransmitted.

24.3.4.3 Receiving the Immediate-reply DLPDU

A received Acknowledge DLPDU shall be treated according to the rules described in 24.1.2.7. The receipt of Immediate-reply DLPDUs shall never be acknowledged.

24.4 DL-service elements of procedure

24.4.1 Receipt of a DL-UNITDATA request primitive

When the DLE receives a DL-UNITDATA request primitive from the local DLS-user, the DLE shall determine, whether the request primitive holds a response for immediate transmission, or a request or response to be queued.

a) if

- 1) the DLE is waiting for a request or response primitive from the local DLS-user, according to the rules described in 24.1.2.5, and
- 2) the contents of the user-specified Destination DL-route parameter in the request is the same as the contents of the Source DL-route parameter in the indication primitive generated by this DLE after receipt of the Confirmed DLPDU from the remote DLE,

the request primitive holds a response for immediate transmission. The DLE shall form and immediately transmit an Immediate-reply DLPDU.

- b) If 0 does not apply, then the request primitive holds a request or a response to be queued. The DLE shall form a Confirmed or Unconfirmed DLPDU.

If the request is accepted, the DLE shall append the DLPDU to the queue for transmission at the first opportunity. The DLPDU is appended to the queue at a position based on the user-specified priority. The specified value can be any integral number from 0 to 255. The DLPDU is placed in front of all DLPDUs currently in the queue having a lower priority, where 255 indicates the highest priority.

The DLE shall create and start a retry timer with duration based on the user-specified Maximum retry time. If the specified value is other than zero, then the duration of this timer shall be equal to that user-specified Maximum retry time; otherwise, the duration shall be 2000 ms. DL-management may override this default duration. The retry timer shall be associated with the DLPDU.

The DLE shall create and clear a retry counter. The retry counter shall be associated with the DLPDU.

The DLE shall associate the user-specified parameters DL-route and Local source ID with the DLPDU, for later use in DL-route conversion.

If the request is rejected, and the user-specified DL-route specifies Confirmed, the DLE shall immediately report the reason for failure in a DL-UNITDATA indication primitive.

24.4.1.1 Forming an Immediate-reply DLPDU

The forming of an Immediate-reply DLPDU is described in the following.

The first Node-address in the Destination DL-route is copied to the address subfield of the first P-NET-route-element, and the Source/Destination designator in that element is set to TRUE. The Node-address of the DLE is copied to the address field of the second element, and the Source/Destination designator in that element is set to FALSE.

The user-specified Control-status parameter is stored in the Control-status (C-S) field of the DLPDU.

The user-specified Data-field-format parameter is stored in the Data-field-format (DFF) field of the DLPDU.

The user-specified Data unit (DLSDU) (the size of which is indicated in bit 1-6 of the Data-field-format field) is stored in the Data field of the DLPDU.

24.4.1.2 Forming a Confirmed or Unconfirmed DLPDU

The forming of a Confirmed or Unconfirmed DLPDU is described in the following.

The DLE's Node-address shall be inserted in front of all other addresses in the source DL-route.

Request P-NET-route generation encodes the resulting DL-route into a P-NET-route with Simple, Extended or Complex format, and stores the result in the P-NET-route field of the DLPDU. Request P-NET-route generation is described in 24.5.1.

The user-specified Control-status parameter is stored in the Control-status (C-S) field of the DLPDU.

The user-specified Data-field-format parameter is stored in the Data-field-format (DFF) field of the DLPDU.

The user-specified Data unit (DLSDU) (the size of which is indicated in bit 1-6 of the Data-field-format field) is stored in the Data field of the DLPDU.

24.4.2 Receipt of a DL-UNITDATA response primitive

When the DLE receives a DL-UNITDATA response primitive from the local DLS-user, the DLE shall determine, whether the response primitive shall result in the immediate transmission of an Acknowledge DLPDU.

a) if

- 1) the DLE is waiting for a request or response primitive from the local DLS-user, according to the rules described in 24.1.2.5, and
- 2) the contents of the user-specified Destination DL-route parameter in the response is the same as the contents of the Source DL-route parameter in the indication primitive generated by this DLE after receipt of the Confirmed DLPDU from the remote DLE,

then the response primitive shall result in the immediate transmission of an Acknowledge DLPDU.

b) if 0 does not apply, then the response primitive is ignored by the DLE. In this situation, the DLE has already sent an autonomously generated Acknowledge DLPDU.

If the response primitive holds an acknowledge for immediate transmission:

The DLE shall form and immediately transmit an Acknowledge DLPDU.

24.4.2.1 Forming an Acknowledge DLPDU

The forming of an Acknowledge DLPDU is described in the following.

The parameters used to form an Acknowledge DLPDU are those from the indication causing the request specifying acknowledge.

The first Node-address in the Source DL-route parameter of the indication is copied to the address subfield of the first P-NET-route-element, and the Source/Destination designator in that element is set to TRUE. The Node-address of the DLE is copied to the address field of the second element, and the Source/Destination designator in that element is set to FALSE.

The Status subfield of the Control-status parameter of the indication is replaced with the code for "Wait" (4) if this DLE is of Simple class, and with the code for "RCL/ACK" (5) if this DLE is of Normal class. The result is stored in the Control-status (C-S) field of the DLPDU.

The value 0 is stored in the Data-field-format (DFF) field of the DLPDU. This value specifies the length of the data field, which is zero.

24.4.3 Autonomous DLE acknowledge

If

- a) the DLE has received an Unconfirmed DLPDU requiring acknowledge, or
- b) the DLE has received a Confirmed DLPDU requiring acknowledge, and does not receive a request or response primitive from the local DLE-user within "Max Indication Delay",

then the DLE shall autonomously form and immediately transmit an Acknowledge DLPDU. The DLPDU is formed as described in 24.4.2.1.

24.4.4 Generation of a DL-UNITDATA indication primitive

The DLE shall inform a DLS-user by generating a DL-UNITDATA indication primitive in the following situations:

- a) When the DLE has received a Confirmed, Unconfirmed or Immediate-reply DLPDU.
- b) When the transmission of a Confirmed DLPDU fails, typically because of a missing acknowledge from the receiving DLE.

The parsing of a received DLPDU into its DL-service parameters, and the generation of DL-service parameters as a result of a failed transmission is described in this subclause.

24.4.4.1 Parsing of a received Confirmed or Unconfirmed DLPDU

The parsing of a received Confirmed or Unconfirmed DLPDU is described in the following.

DL-route generation generates Destination DL-route and Source DL-route from the received Simple, Extended or Complex P-NET-route. DL-route generation is described in 24.5.2.

The value of Confirm is set to FALSE if the value of one of the addresses in the Destination DL-route is = BNA, or the value of the last address in the Source DL-route is = NCNA.

The first element of the Destination DL-route is removed.

The Control-status parameter shall be the Control-status (C-S) field of the DLPDU.

The Data-field-format parameter shall be the Data-field-format (DFF) field of the DLPDU.

The Data unit (DLSDU) parameter shall be the Data field of the DLPDU (the number of octets in DLSDU is indicated in bit 1-6 of the Data-field-format field).

24.4.4.2 Parsing of a received Immediate-reply DLPDU

The parsing of a received Immediate-reply DLPDU is described in the following.

The user-specified Destination DL-route parameter from the request causing this immediate reply is copied to the Source DL-route indication parameter. A DL-route-element is appended at the end, with the value of No-Confirm-Node-Address (0). The Source DL-route request parameter is copied to the Destination DL-route indication parameter.

The value of Confirm shall be FALSE.

The Control-status parameter shall be the Control-status (C-S) field of the DLPDU.

The Data-field-format parameter shall be the Data-field-format (DFF) field of the DLPDU.

The Data unit (DLSDU) parameter shall be the Data field of the DLPDU (the number of octets in DLSDU is indicated in bit 1-6 of the Data-field-format field).

24.4.4.3 Generation of indication primitive indicating error

When the transmission of a Confirmed DLPDU fails, typically because of a missing acknowledge from the receiving DLE, the transmitting DLE shall inform its local DLS-user by generating a DL-UNITDATA indication primitive, indicating the locally detected reason for error.

The generation of an indication primitive indicating error is described in the following.

The user-specified Destination DL-route parameter from the request causing the failed transmission is copied to the Source DL-route indication parameter. A DL-route-element is appended at the end, with the value of No-Confirm-Node-Address (0). The Source DL-route request parameter is copied to the Destination DL-route indication parameter.

The value of Confirm shall be FALSE.

The value of the Control-status parameter shall indicate the locally detected reason for failure.

The value of the Data-field-format parameter shall be zero, indicating that length of the DLSDU parameter is zero.

24.5 Route mechanism

The route mechanism provides the functionality of "guiding" a request from a requestor to the responder possibly through one or more gateways, and of "guiding" the response from the responder back to the requestor.

The request and response may pass up to 12 gateways.

Two of the parameters of the DL-services request and indication are Destination DL-route and Source DL-route. In the request from a requestor the Destination DL-route parameter describes the complete route to the responder, and the Source DL-route parameter describes the complete route the requestor.

As the request is on its way towards the responder, the Destination DL-route becomes shorter and shorter, and the Source DL-route becomes longer and longer, as a result of the route conversion that is performed in the gateways.

When the request reaches the responder, the Source DL-route describes the complete route back to the requestor. The responder "swaps" the Destination and Source DL-routes, giving the result that the Destination DL-route now describes the complete route back to the requestor.

When the response reaches the requestor, the Source DL-route describes the complete route to the responder (as did the Destination DL-route in the original request).

24.5.1 Request P-NET-route generation

Request P-NET-route generation is performed by the DLE as a part of forming a DLPDU when it has received a DL-UNITDATA request primitive from the local DLS-user.

Request P-NET-route generation encodes the Destination DL-route and Source DL-route parameters into a P-NET-route of Simple, Extended or Complex format.

Generally, a DL-route-element from the Destination DL-route is copied to a P-NET-route-element by copying the value of the DL-route-element to the address subfield of the P-NET-route-element, and setting the value of the Source/Destination designator to FALSE. A DL-route-element from the Source DL-route is copied to a P-NET-route-element by copying the value of the DL-route-element to the address subfield of the P-NET-route-element, and setting the value of the Source/Destination designator to TRUE.

Figure 100 illustrates the generation of the Source/Destination designator.

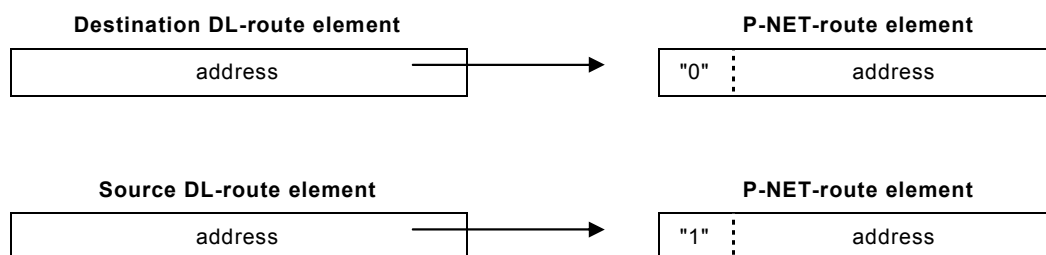


Figure 100 – Source / destination designator

The Destination DL-route is copied to the first elements of the P-NET-route, and the Source DL-route is copied to the succeeding elements.

One of three alternatives apply:

- a) If the Destination DL-route holds only one element, the P-NET-route shall be of Simple format.

The first element of the P-NET-route shall be a copy of the first element of the Destination DL-route. The second octet of the P-NET-route shall be a copy of the first element of the Source DL-route.

Figure 101 illustrates this.

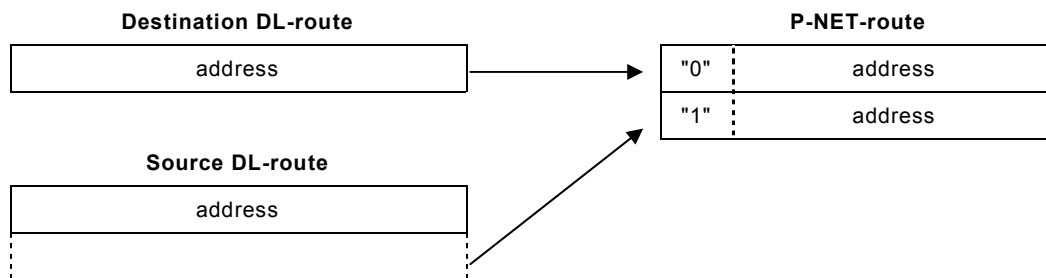


Figure 101 – Simple P-NET-route generation

- b) If the Destination DL-route and the Source DL-route hold only two elements each, the P-NET-route shall be of Extended format.

The first two elements of the P-NET-route shall be a copy of the two elements in the Destination DL-route. The next two elements of the P-NET-route shall be a copy of the two elements in the Source DL-route. Figure 102 illustrates this.

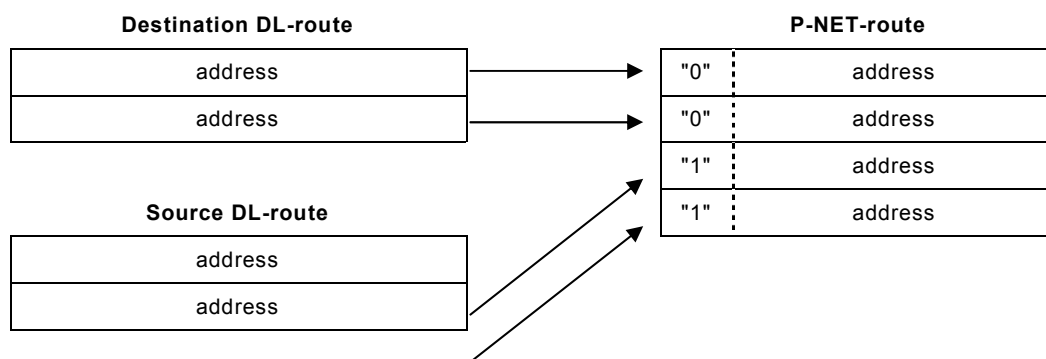


Figure 102 – Extended P-NET-route generation

- c) If the Destination DL-route holds at least two elements, and the total DL-route (the combination of Destination and Source DL-route) holds more than four elements, the P-NET-route shall be of Complex format.

The first two elements of the P-NET-route shall be a copy of the first two elements in the Destination DL-route. The third element shall be the total number of elements in the DL-route minus two. A copy of the remaining elements of the Destination DL-route (if any) shall follow the third element of the P-NET-route. They shall be followed by all of the elements of the Source DL-route. Figure 103 illustrates this.

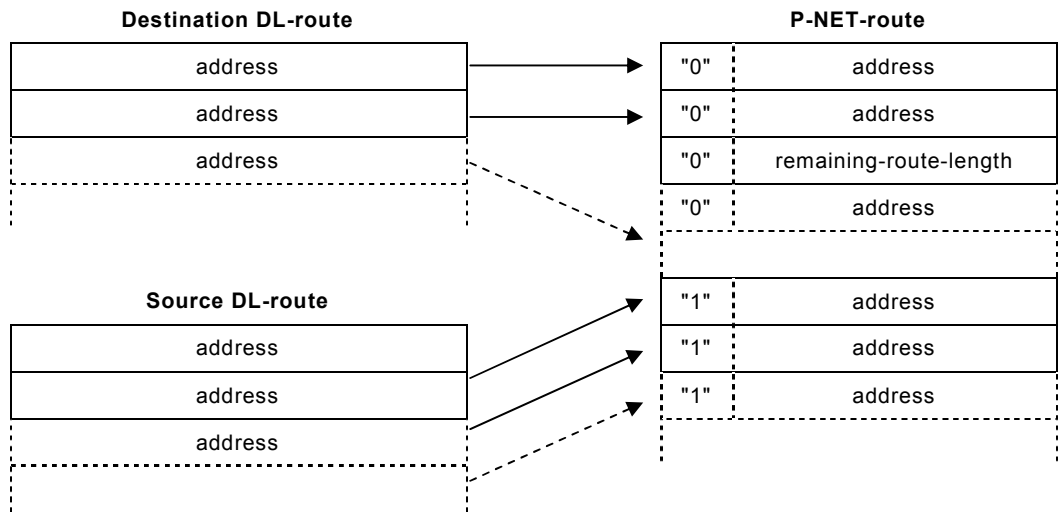


Figure 103 – Complex P-NET-route generation

24.5.2 DL-route generation

DL-route generation is performed by the DLE when it has received a Confirmed or Unconfirmed DLPDU, as part of generating an indication service primitive to inform the local DLS-user of the receipt.

The three different P-NET-route formats Simple, Extended and Complex are supported in Normal class DLEs. Only Simple route format is supported in Simple class DLEs.

Generally, a P-NET-route-element is copied to a DL-route-element by copying the value from the address subfield of the P-NET-route-element.

One of three alternatives apply:

- a) If the P-NET-route in the received DLPDU is of Simple format, the Destination DL-route shall be empty. The source element of the received P-NET-route shall be copied to the Source DL-route. Figure 104 illustrates this.

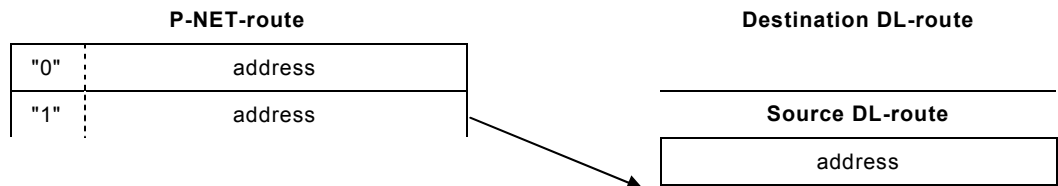


Figure 104 – Simple DL-route generation

- b) If the route in the received DLPDU is of Extended format, the second destination element of the received P-NET-route shall be copied to the first and only element in the Destination DL-route. The Source DL-route shall be a copy of the two source elements in the received P-NET-route. Figure 105 illustrates this.

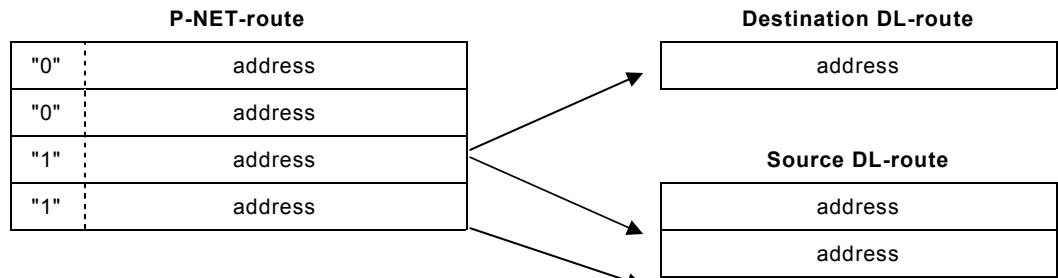


Figure 105 – Extended DL-route generation

- c) If the P-NET-route in the received DLPDU is of Complex format, the second element of the received P-NET-route shall be copied to the first element of Destination DL-route. The remaining destination elements of the received P-NET-route shall be copied to the Destination DL-route. All of the source elements of the received P-NET-route shall be copied to the Source DL-route. Figure 106 illustrates this.

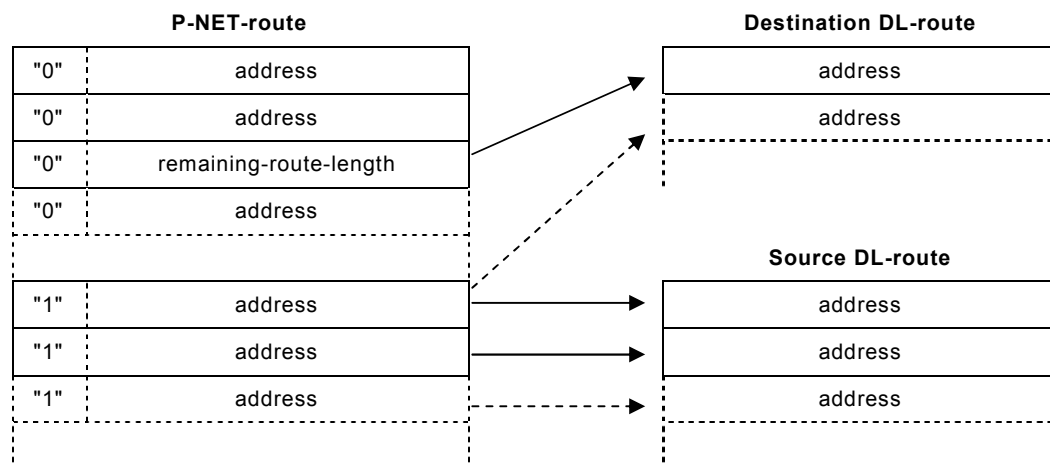


Figure 106 – Complex DL-route generation

24.6 Link-access system

The basic principles and most details of the Link-access system are described in 24.1.2.8. This subclause describes how to gain Link-access synchronization after DLE initialization, and after loss of synchronization.

After DLE-initialization and after loss of synchronization the DLE is declared OutOfSync. As long as the DLE is OutOfSync it is only allowed to transmit Immediate-reply and Acknowledge DLPDUs.

Synchronization can be gained in two different ways.

- a) When the DLE lost synchronization because of a difference between the Link Access Counter C(LAC) and the received Node-address in a Sync SPDU, the difference between C(LAC) and the received Node-address is saved.

If the received Node-address in the next Sync SPDU is equal to C(LAC), the DLE is in synchronization again.

If the received Node-address in the next Sync SPDU is different from C(LAC), but the difference is the same as the saved difference, C(LAC) is set equal to the received Node-address, and the DLE is in synchronization again. If the difference is not the same, the new difference is saved, and the DLE is still OutOfSync.

- b) If the DLE is obliged to send a Sync SPDU because the Link has been idle for 360 bit periods or more when the DLE receives the Virtual Link-access token, and the DLE is OutOfSync, it is not allowed to transmit, but shall instead declare itself in synchronization again. This gives the result, that the next time this DLE receives the Virtual Link-access token, and the Link is still idle, this DLE transmits a sync SPDU. This rule covers the situation of starting up, and situations where there is only one Normal class DLE on the Link.

24.7 Local variables, counters and queues

24.7.1 V(ACPDU) - Acknowledge Confirmed PDU

The value of this variable defines, whether this DLE should acknowledge the receipt of Confirmed DLPDUs, and expect remote DLEs to acknowledge the receipt after this DLE has transmitted Confirmed DLPDUs. The possible values of this variable are TRUE and FALSE. A value of TRUE indicates, that Confirmed DLPDUs shall be acknowledged.

24.7.2 V(AUPDU) - Acknowledge Unconfirmed PDU

The value of this variable defines, whether this DLE should acknowledge the receipt of Unconfirmed DLPDUs, and expect remote DLEs to acknowledge the receipt after this DLE has transmitted Unconfirmed DLPDUs. The possible values of this variable are TRUE and FALSE. A value of TRUE indicates, that Unconfirmed DLPDUs shall be acknowledged.

24.7.3 V(NA) - Node-address

This variable holds the Node-address of this DLE. The Node-address uniquely identifies this DLE on the Link. The range of V(NA) is 0 to 125. A value of 0 indicates that V(NA) has not been initialized.

24.7.4 V(NDLE) - Number of DLEs

This variable only exists in Normal class DLEs. It holds the maximum number of Normal class DLEs on the Link. The value of V(NA) must be lower than or equal to the value of V(NDLE). The value of V(NDLE) must not exceed 32. The value of V(NDLE) shall be the same in all DLEs on the Link. A value of 0 indicates that V(NDLE) has not been initialized.

24.7.5 V(PNR) - Permitted Number of Retries

This variable only exists in Normal class DLEs. It holds the maximum number of transmission retries allowed, when transmission of Confirmed or Unconfirmed DLPDUs fails. Its range is 0 to 10.

24.7.6 V(DC) - Device Class (Simple or Normal)

This variable indicates the class of the DLE. It can indicate Simple or Normal. The possible values of this variable are TRUE and FALSE. A value of TRUE indicates that the DLE is of Normal class.

24.7.7 V(BR) - Bit rate

This DLE variable is used by the DLE-user to set the desired bit rate. When the DLS-user changes the value of this variable, the DLE shall issue a PhE management service to change to bit rate in the PhL. The value of this variable is also the basis of calculating the value of Max Indication Delay. The possible values depend on the capability of the PhE.

24.7.8 V(MID) - Max Indication Delay

This DLE variable indicates to the DLS-user, how many microseconds the DLS-user is allowed to use to prepare a response after receiving an indication requiring that. If the DLS-user is unable to prepare a response within Max Indication Delay, the DLS-user must issue a DL-UNITDATA request with a DLSDU type indicating Acknowledge. The DLE will as a result transmit an Acknowledge DLPDU on the Link.

The value of Max Indication Delay is calculated by the DLE from the value of V(BR), as 30 bit periods. The range of Max Indication Delay is 3 μ s (corresponding to a bit rate of 10 Mbit/s) to 100 ms (corresponding to a bit rate of 300 bit/s).

24.7.9 V(DMRT) - Default Max retry time

This variable indicates to the DLE, for how many ms retransmission of the request may be performed, as a result of Wait acknowledges from the remote DLE or DLS-user. It is used if the DLS-user didn't specify a Max retry time parameter in the request. The default value of V(DMRT) is 2000 ms. DL-management may override this default value. The range of V(DMRT) is 10 to 30000 ms.

24.7.10 Q(UR) - User Request Queue

This variable only exists in Normal class DLEs. It is a multi-priority FIFO-within-priority queue, holding DLPDUs queued for transmission, the associated DL-route and Local source ID parameters, retry counter and retry timer. One such queue is associated with each DLE.

24.7.11 C(LAC) - Link Access Counter

This counter only exists in Normal class DLEs. It holds the Node-address of the DLE having the Virtual Link-access token.

24.7.12 C(LIC) - Link Idle Counter

This counter only exists in Normal class DLEs. It holds information on, for how long time the Link has been idle.

25 Type 5: <placeholder for the Type 5 DL-protocol definition>

A normative specification of this protocol was not available at the time of compilation of this edition.

The Type 5 protocol is an adaptation of the Type 1 protocol for a PhL consisting of high-speed counter-rotating prioritized buffer insertion rings. It provides the identical Type 1 DL-service. In many respects the Type 5 DLPDUs and corresponding elements of procedure are identical to those of the Type 1 protocol; the primary differences are as follows:

- a) The 4-octet LONG DL-addresses of Type 1 are unchanged in Type 5; they provide for addressing the entire extended link. The DLPDU FC-octet code points used with LONG DL-addresses, and the associated parameter field definitions, are identical in the Type 1 and Type 5 protocols. (See EC, DC, RC, CA, CD, ED and DT DLPDUs, 8.1 through 8.7.) These are the DLPDUs that are forwarded between Type 1 and Type 5 links.
- b) The Type 5 protocol replaces Type 1's 2-octet SHORT DL-address with a 6-octet STANDARD DL-address of the form used by ISO/IEC 8802 and other local area networks. These STANDARD DL-addresses are restricted to the Type 5 subset of the extended link. The DLPDU FC-octet code points used with SHORT DL-addresses in Type 1 are used with STANDARD DL-addresses in Type 5.

This addressing substitution applies to all EC, DC, RC, CA, CD, ED and DT DLPDUs (see 8.1 through 8.7). The maximum amount of DLS-user data conveyable by these DLPDUs when using STANDARD DL-addresses is increased to 4k octets, and the formats and sequence number moduli of the associated parameter fields are extended for computational efficiency and to permit a greater number of DLPDUs to be in transit between sender and receiver(s).

- c) The Type 5 protocol concatenates a second FCS to that used by the Type 1 protocol to provide the greater burst error detection required of a high-speed PhL. The second FCS was selected on a similar basis to that of Type 1 (see 5.1) — to maximize burst error detection for DLPDUs of lengths permitted by the Type 5 protocol when the statistics of the error burst have a Poisson distribution (as is the usual case).

When a bridge forwards a DLPDU (which will have 4-octet LONG DL-addresses) from a Type 1 link to a Type 5 link, it may transform the received FCS as specified in 5.1.3 and computes and appends the second FCS; it does not recompute the first FCS. When a bridge forwards a DLPDU from a Type 5 link to a Type 1 link, it discards the second FCS before forwarding the Type 1 DLPDU information with just the first FCS. This concatenated FCS construction provides for end-to-end error detection throughout the entire extended link, independent of the number or types of the intermediary links.

- d) Due to the intrinsic differences between Type 1 and Type 5 PhLs, the Type 5 protocol slightly modifies those Type 1 DLPDUs and corresponding elements of procedure which are used for
 - 1) determination of the presence of DLEs on the local link and distribution of basic link parameters (PN and PR, see 8.13 and 8.14);
 - 2) determination of the unique DLE, the Link Active Scheduler (LAS), that serves as local link administrator, schedule coordinator and source of DL-time (CL and TL, see 8.19 and 8.20);
 - 3) equitably regulating the sharing of unscheduled communications capacity among the DLEs of the local link (PT, see 8.15);
 - 4) determination of the approximate one-way delay of time distribution (TD) DLPDUs as received by any DLE from the current LAS DLE (RQ and RR, see 8.11 and 8.12);
 - 5) sharing the current sense of DL-time (CT and TD, see 8.9 and 8.10);
 - 6) responding with status but without DLS-user data to a DLPDU which in Type 1 requested an "immediate" reply (SR, see 8.8).

- e) All other Type 1 DLPDUs (ES, RT, RI, WK and IDLE, see 8.16 through 8.18, 8.21 and 8.22) and all instances of DT DLPDUs (see 8.7) which use VERY-SHORT addressing are not used by the Type 5 protocol. A small number of additional DLPDUs are defined to assist in failure detection and maintenance of the Type 5 PhL.
- f) The Type 5 protocol maintains clock synchronization across an extended link with a precision approximately equal to the transmission duration of one octet on the slowest interconnecting link.
- g) All execution of DL-schedules occurs within the individual Type 5 DLEs; unlike the Type 1 protocol it is never delegated to the LAS. All SPDUs and elements of procedure related to schedule delegation (see 12.3 and clause 11) are not used by the Type 5 protocol.

26 Type 6: Connection-oriented DL-protocol and related matters

This clause describes the real time Connection Oriented Data Transfer, Time and Coordination protocol, for which Type 6's design and performance are optimized. This clause also describes the DLM-connectionless protocol, which is used for configuration.

This protocol uses TDMA (time division multiple access) on a bus to provide dedicated high-speed low-jitter synchronous channels concurrently with demand-assigned channels. The dedicated channels are suitable for distribution or exchange of automation state data; as well as for pre-established low-latency messaging paths.

Depending on bus configuration, this protocol can support up to 120,000 distinct messages per second on a 5 Mbit/s PhL.

This clause also describes bridge operation; bus configuration and conformance.

26.1 Connection mode data transfer using TDMA principles

This protocol provides a slotted Time Division Multiple Access (TDMA) system. Time on the bus is divided into equal length intervals called **slots**. Each DLE listens to traffic, and may send traffic, only during a slot assigned to it. A local link has $2^{L_{BL}}$ slots during one bus cycle. TDMA operation is shown in Figure 107, including equal length slots, and **channels** that are composed of equally spaced sets of slots.

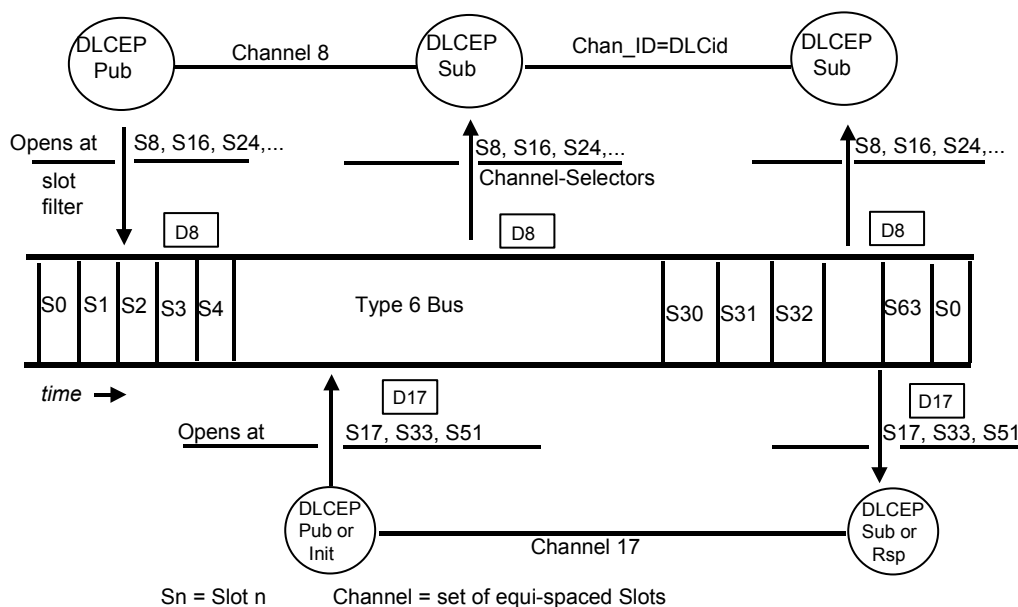


Figure 107 – Connection mode operation using TDMA

26.2 Bus access and data transmission

Bus access, data transmission and data reception is coordinated by each DLE based on a shared perception of which Channel and Slot now occupies the bus. This perception is derived from the Bus-Cycle-Counter (incremented in Slots) and Slot-clock (incremented in Bit-times) maintained by each DLE.

The Relationship of Channels, Slots and Bus Synchronization and operation of a TDMA bus, with a highly abbreviated bus cycle, is shown in Figure 108.

Start abbreviated Bus Cycle 1 End															Start of Bus Cycle 2	
shown for an artificial Bus-Cycle with 16 Slots and 3 Scan Classes the typical fieldbusBus-Cycle has 64K (65536) Slots and 15 Scan Classes																
Ch A		Ch A		Ch A		Ch A		Ch A		Ch A		Ch A	<i>Bus Sync period</i>	Ch A	Scan-Class 1, Channel A @ max. Scan Rate	
	Ch B			Ch C				Ch B				Ch C	<i>Bus Sync period</i>		Scan-Class 2, Channels B,C @ 25% of max. Scan Rate	
			Ch D				Ch E Syn Ch				Ch F		<i>Bus Sync period</i>		Scan-Class 3, Channels D,E,F @ 12,5% of max.Scan Rate	

Figure 108 – Fundamental access concepts

26.2.1 Bus-cycle-length

Bus-cycle-length is a conformance-class issue. Its form is 2^{LBL} and its range is 2^8 to 2^{56} , inclusive.

NOTE In an extended link, Bus-cycle-length is typically larger on or near the TOP link.

26.2.2 Channels

Each variable measured by a sensor, or stimulus sent to an actuator, uses a dedicated channel, which is configured for, and occupies, one or more slots per Bus-Cycle.

When one of the slots which is assigned to one of the channels in a DLE is noted by that DLE's Bus-Cycle clock, it either transmits or receives (as assigned) the data which is present on the bus during that slot.

NOTE The slots on a local bus can be distributed by configuration among typically 10 s or 100 s or 1000 s (up to Bus-Cycle-Length-1) of channels to achieve the scan rates and variable coverage needed for the transducers and higher-level protocols used in the specific real world system.

26.2.3 Slots

All slots on the bus are the same length for one bus configuration and mode of operation. Each slot may be assigned to only one channel. Each Channel may be scanned one to many times during each Bus-Cycle. These restrictions are enforced by the Bus Configuration process.

26.2.4 Restrictions on slot assignments

At the slot level, potential conflicts between various channels are resolved in advance by the Bus Configuration process.

26.2.5 Master and slave DLCEPs

On peer channels, only one DLCEP may transmit in any given slot. One DLCEP is designated the Master. It controls the direction of transmission by either delegating the right to transmit to itself or to the Slave DLCEP. This delegation process is accomplished by the setting of the CD-Semaphore.

26.2.6 Publisher and subscriber DLCEPs

On multi-peer channels, only one DLCEP (the Publisher) may transmit in any given slot, but any number of DLCEPs (the Subscriber(s)), except the Publisher, may listen to the data. This is equivalent to stating that each Channel may have one Publisher with one or more Subscribers.

26.2.7 Buffers and queues

Buffers and Queues are used for communicating DLS-user data (DLSDU) between the DLS-user and the DLE at one DLSAP. Queues of depth(1) are used for communicating DLMS-user data (DLMSDU) between the DLMS-user and the DLE at one DLSAP.

- a) If the Buffer-or-Queue is a non-retentive buffer or a Queue of depth(1), as defined in Part 3, the DLE removes the DLSDU (or DLMSDU) after completion of the transfer procedure.
- b) If the Buffer-or-Queue is a retentive buffer, as defined in Part 3, the DLE does not remove the DLSDU. This action is the responsibility of the associated DLS-user.

26.3 Overview of bus synchronization

The Conductor's DLE and each other node's DLE cooperate to Synchronize all DLEs on a fieldbuslocal link or extended link.

The Conductor DLE starts-up and synchronizes bus operation. The Conductor is the one Host configured or chosen by the contention protocol as the (active) Conductor.

Two levels of bus synchronization are used:

- a) Bus Cycle Synchronization secures the basic sequencing of the bus and establishes overall bus timing.
- b) Slot clock Synchronization ensures precise inter-DLE synchronization, in the presence of hardware vagaries.

NOTE The Conductor does not actively schedule traffic on the link, as is done by a Type 1 LAS.

26.3.1 Method used on local link

Once per Bus-Cycle the Conductor DLE emits a DLPDU with a special set of delimiters and Sync-type = TRUE, as defined in 27.1. These delimiters are highly secure. They allow each other DLE on the local link to synchronize its Bus-Cycle-counter with the Conductor's Bus-Cycle-counter at the SYN bit of the Bus-End-Sync DLPDU, as shown in IEC 61158-2.

26.3.2 Method used on extended link

Each bridge in an extended link has one **Upper Port Bridge** DLE and may have one or more **Lower-Port Bridge** DLE's.

A Bridge DLE may be either:

- a) a **Synchronizer**, when all links in the extended link operate synchronously, but there is no data transfer between links.

In a Synchronizer one Upper Port Bridge-DLE synchronizes multiple Lower Port Bridge DLEs, by means of zero latency Bus Sync, Slot Sync, Slot Clock and Bit Clock signals.

EXAMPLE An example of a Synchronizer is shown in Figure 109.

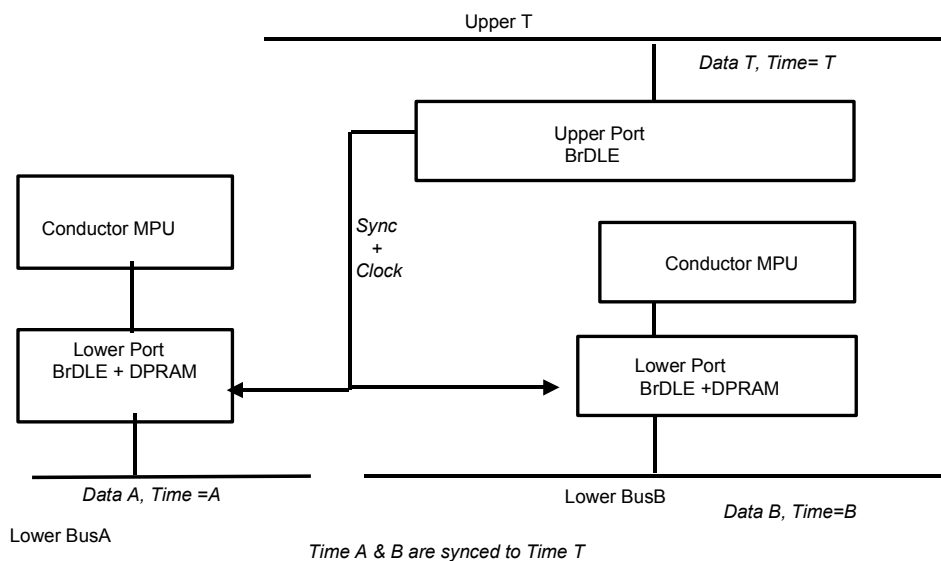


Figure 109 – EXAMPLE: Architecture of a synchronizer showing one upper port bridge-DLE synchronizing multiple lower port bridge-DLEs

- b) a **Forwarding Bridge** when all links in the extended link operate synchronously and there is data transfer between DLEs on the links of the extended link.

In a Forwarding Bridge one Upper Port Bridge-DLE synchronizes exactly one Lower Port Bridge-DLE, and may transfer data between these two links.

EXAMPLE An example of a Forwarding Bridge is shown in Figure 110.

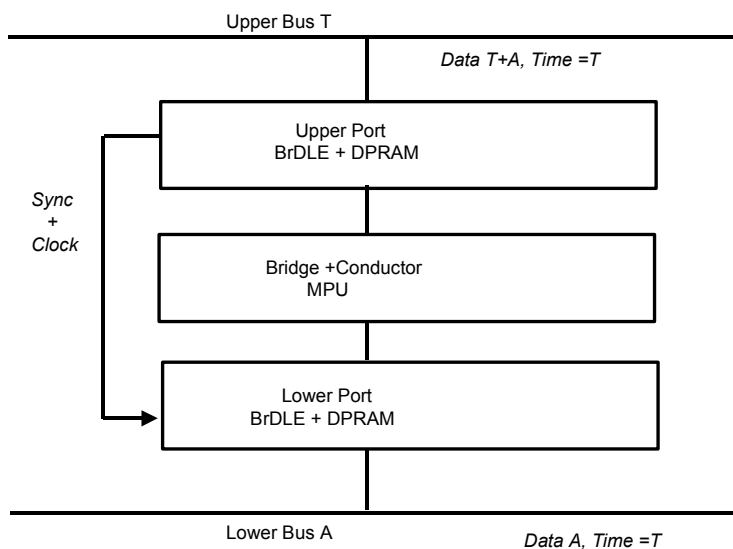


Figure 110 – EXAMPLE: Architecture of a forwarding bridge showing one upper port bridge-DLE synchronizing exactly one lower port bridge-DLE

26.3.3 Bus-cycle synchronization

Bus-cycle Synchronization is defined in 26.3.1.

26.3.4 Slot-clock synchronization

26.3.4.1 Local link

When a Slot-sync channel selector is present on the bus, the Conductor DLE emits a Bus-Sync DLPDU with a special set of delimiters and Sync-type = FALSE, as defined in 27.1. These delimiters are highly secure. They allow each other DLE on the local link to synchronize its Slot-clock with the Conductor's Slot-clock at SYN bit of the Bus-Sync DLPDU, as shown in IEC 61158-2.

26.3.4.2 Extended link

As shown in Figure 109 through Figure 111, Sync and crystal clock are passed by a zero latency interface between the Upper-Port Bridge DLE and the Lower-Port Bridge DLE's of a Bridge. This assures synchronism of the Lower-Port Bridge DLE's Slot and Bit clocks with those of the upper link.

The Bus Cycle Synchronization mechanism and the Slot Clock Synchronization mechanism operates on each local link. These mechanisms synchronize all the slot clocks on each lower link, including any lower level Upper-Port Bridge DLEs.

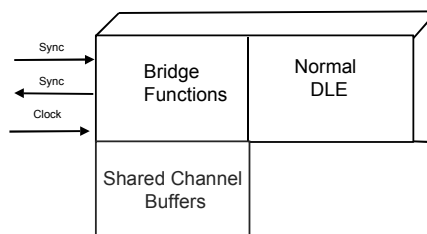


Figure 111 – EXAMPLE: Lower-port bridge DLE

EXAMPLE Figure 112 shows an example bridged network using Forwarding Bridges. In this figure, bridge DLEs D, E, O and P are Upper Port Bridge DLEs and DLEs G, V A1, and B1 are the associated Lower Bridge DLEs. Each of these Lower Bridge DLEs, and thus links A, B, R and S, are synchronized by the associated Upper Bridge DLE. Port Bridge DLE D is the Conductor on the Top link T and thus Port Bridge DLE D ultimately synchronizes the entire extended link.

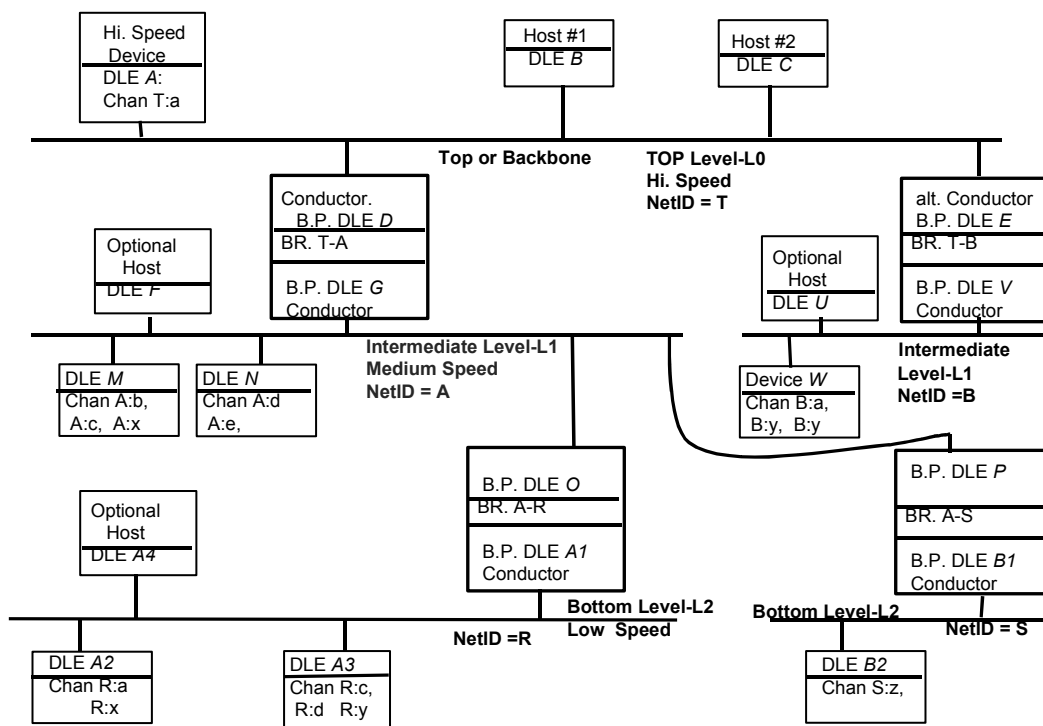


Figure 112 – EXAMPLE: Bridged network using forwarding bridges

26.4 Connection mode data transfer

This subclause provides a conceptual definition of the services provided by the DLS-provider to the DLS-user(s). Nothing in this subclause shall be construed to constrain the actual implementations of the interactions at the DLS-provider to DLS-user interface.

26.4.1 Three level model of connection mode operation

Connection mode architecture follows a three level model.

- The first level, Channels, are composed of slots which are formed as described in IEC 61158-2.
- The second level, Channel Methods, are elements of procedure for transmitting DLSDUs, described in 26.4.3, that are, or could be, common to more than one Channel Class.
- The third level, Channel Classes, described in 26.4.2, combine the elements of procedure for Channel Methods, to perform real-time connection-mode data transfers and provide the specified QoS to the DLS-users.

Channel identification is described in 26.4.4.

26.4.2 DLS-user interface and channel operation related to channel-class

Connection mode architecture uses four Channel-classes:

- SCAN – for state DLSDUs that are short relative to max-data-length
- ExSCAN – for state DLSDUs that may be long relative to max-data-length
- GPA – for use by DLS-users that do not need bi-directional DLSDU flow
- GPC – for use by DLS-users that may need bi-directional DLSDU flow

26.4.2.1 SCAN class

The operation of the SCAN class and its interface between the DLE and the DLS-user is shown in Figure 113.

The number of SCAN channels per DLSAP is a conformance-class issue. The minimum is 0 and the maximum is 248.

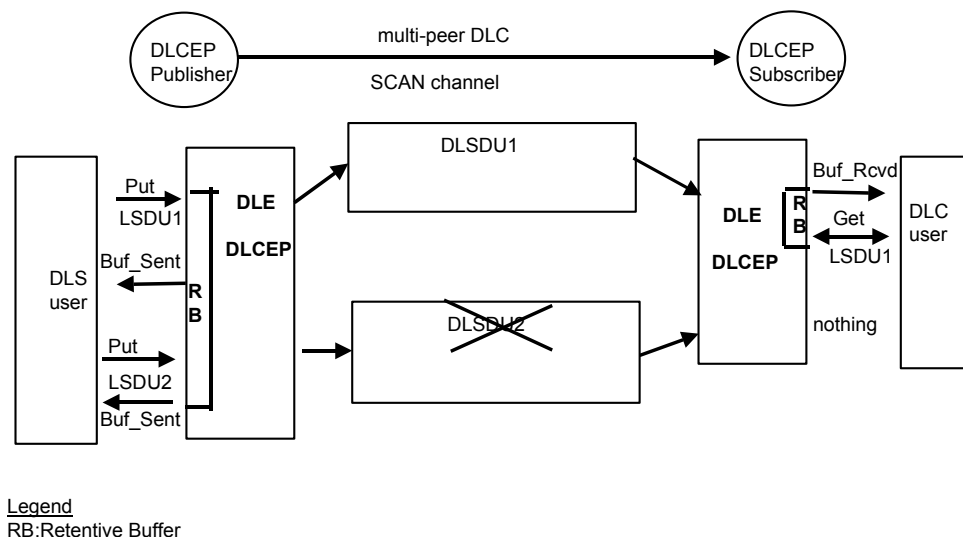


Figure 113 – Operation of the SCAN channel-class and its associated primitives

26.4.2.1.1 Primitives and parameters

The primitives and parameters used on SCAN channels are shown in Table 204.

Table 204 – Primitives and parameters used on SCAN channels

Function	Location	Primitive	Direction	Parameters
submit DLSDU	Publisher	DL-PUT	to DLE	DLC-identifier DLSDU-buffer-identifier DLSDU-length
confirm to publishing DLS user	Publisher	DL-BUFFER-SENT	from DLE	DLC-identifier DLSDU-buffer-identifier Status
notify subscribing DLS user	Subscriber(s)	DL-BUFFER-RECEIVED	from DLE	DLC-identifier
retrieve DLSDU	Subscriber(s)	DL-GET	to/from DLE	DLC-identifier DLSDU-buffer-identifier DLSDU-length
NOTE In this table, time increases from top to bottom.				

26.4.2.1.2 Parameter handling

On SCAN channels DLEs handle parameters passed by/to the DLS-user as follows:

26.4.2.1.2.1 DLC-identifier

The Publisher DLE transmits DLPDUs via the channel identified by DLC-identifier in the DL-PUT primitive. Publisher DLEs notify their local DLS-user of which channel has transmitted a DLPDU by the DLC-identifier of the DL-BUFFER-SENT primitive.

Subscriber DLEs notify their local DLS-user of which channel has received a DLPDU via the DLC-identifier of the DL-BUFFER-RECEIVED primitive. The local DLS-user notifies the Subscriber DLE as to which channel's data it wishes to retrieve by the DLC-identifier of the DL-GET primitive.

26.4.2.1.2.2 DLSDU-buffer-identifier

The Publisher DLE retrieves DLSDUs submitted by the local DLS-user from the retentive buffer identified by the DLSDU-buffer-identifier of the DL-PUT primitive. Publisher DLEs notify their local DLS-user of which DLSDU has been transmitted by the DLSDU-buffer-identifier of the DL-BUFFER-SENT primitive.

Subscriber DLEs notify their local DLS-user as to which DLSDU has been received on DLC-identifier via the DLSDU-buffer-identifier in the DL-GET primitive.

26.4.2.1.2.3 DLSDU length and status

The Publisher DLE uses the DLSDU-length parameter to decide whether to transmit the DLSDU identified by the DLSDU-buffer-identifier:

```

IF DLSDU-length > max-data-length
THEN
    Return status = "data-too-long"
ELSE
    Return status = "data-accepted"
    The Publisher prepares to transmit the DLSDU using the UNITARY method
ENDIF
    
```

26.4.2.2 ExSCAN class

The operation of the ExSCAN class and its interface between the DLE and the DLS-user is shown in Figure 114.

The number of ExSCAN channels per DLSAP is a conformance-class issue. The minimum is 0 and the maximum is 248.

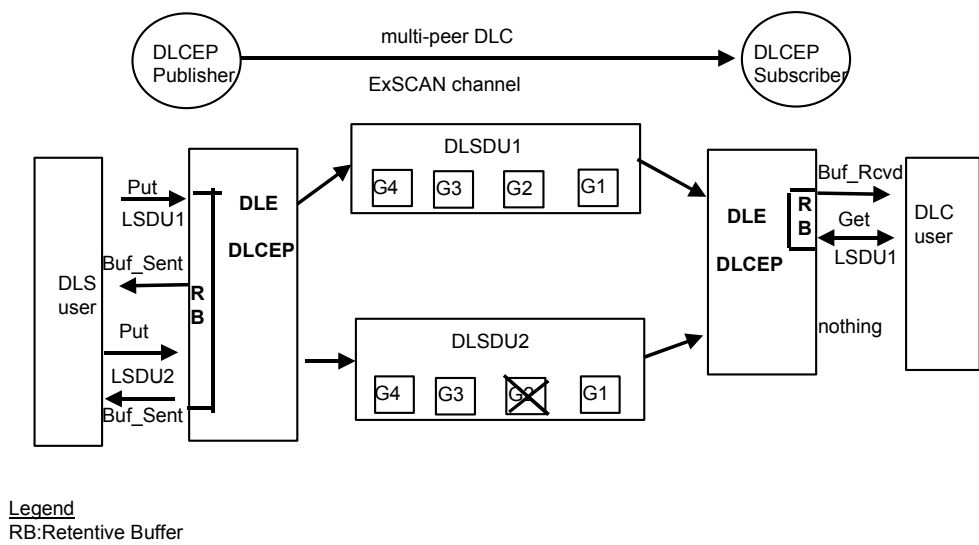


Figure 114 – Operation of the ExSCAN channel-class and its associated primitives

26.4.2.2.1 Primitives and parameters

The primitives and parameters used on ExSCAN channels are shown in Table 205.

Table 205 – Primitives and parameters used on ExSCAN channels

Function	Location	Primitive	Direction	Parameters
submit DLSDU	Publisher	DL-PUT	to DLE	DLC-identifier DLSDU-buffer-identifier DLSDU-length
confirm to publishing DLS user	Publisher	DL-BUFFER-SENT	from DLE	DLC-identifier DLSDU-buffer-identifier Status
notify subscribing DLS user	Subscriber(s)	DL-BUFFER-RECEIVED	from DLE	DLC-identifier
retrieve DLSDU	Subscriber(s)	DL-GET	to/from DLE	DLC-identifier DLSDU-buffer-identifier DLSDU-length
NOTE In this table, time increases from top to bottom.				

26.4.2.2.2 Parameter handling

On ExSCAN channels DLEs handle parameters passed by/to the DLS-user as follows:

26.4.2.2.2.1 DLC-identifier

The Publisher DLE transmits DLPDUs via the channel identified by DLC-identifier in the DL-PUT primitive. Publisher DLEs notify their local DLS-user of which channel has transmitted a DLPDU by the DLC-identifier of the DL-BUFFER-SENT primitive.

Subscriber DLEs notify their local DLS-user of which channel has received a DLPDU via the DLC-identifier of the DL-BUFFER-RECEIVED primitive. The local DLS-user notifies the Subscriber DLE as to which channel's data it wishes to retrieve by the DLC-identifier of the DL-GET primitive.

26.4.2.2.2.2 DLSDU-buffer-identifier

The Publisher DLE retrieves DLSDUs submitted by the local DLS-user from the retentive buffer identified by the DLSDU-buffer-identifier of the DL-PUT primitive. Publisher DLEs notify their local DLS-user of which DLSDU has been transmitted by the DLSDU-buffer-identifier of the DL-BUFFER-SENT primitive.

Subscriber DLEs notify their local DLS-user as to which DLSDU has been received on DLC-identifier via the DLSDU-buffer-identifier of the DL-GET primitive.

26.4.2.2.2.3 DLSDU length and status

The Publisher DLE uses the DLSDU-length parameter to decide whether to transmit the DLSDU identified by the DLSDU-buffer-identifier and the number-of-grains to be transmitted:

IF DLSDU-length > max-DLSDU-length as given below

THEN

Return status = "data-too-long"

ELSE the Publisher DLE calculates the number-of-grains to be transmitted

NOTE The following equations apply to all GRANULAR transfers. The notation used is from Excel5.

$$\text{max-grain-size} = [\text{max-data-length} - 1] \quad (60)$$

$$\text{number-of-grains} = \text{ROUNDUP} [\text{DLSDU-length} / \text{max-grain-size}] \quad (61)$$

$$\text{max-DLSDU-length} = \text{MIN} [\text{max-DLSDU-size from Table 206}, \text{max-DLSDU-buffer-size}] \quad (62)$$

Return status = "data-accepted"

ENDIF

Table 206 – Correspondence of max-DLSDU-size and max-data-length for GPA, GPC and ExSCAN channels

Max-data-length (octets)	Max-DLSDU-size (octets)
2	64
3	128
4 or 5	256
6 to 9	512
10 to 17	1024
18 to 33	1792

Max-DLSDU-buffer-size is set by the conformance-class in use. Its range is 64 to 1792 octets in the same increments as shown for max-DLSDU-size in Table 206.

26.4.2.3 GPA class

The operation of the GPA class and its interface between the DLE and the DLS-user is shown in Figure 115.

The number of GPA channels per DLSAP is a conformance-class issue. The minimum is 0 and the maximum is 248.

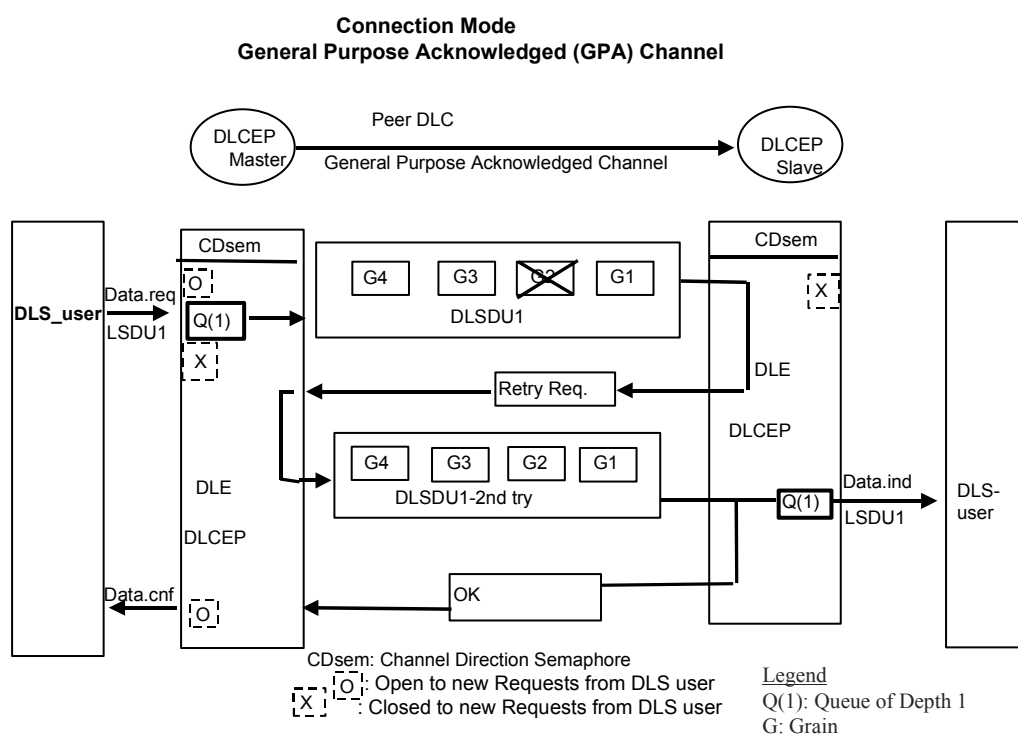


Figure 115 – Operation of the GPA channel class and its associated primitives

26.4.2.3.1 Primitives and parameters

The primitives and parameters used on GPA channels are shown in Table 207.

Table 207 – Primitives and parameters used on GPA channels

Function	Location	Primitive	Direction	Parameters
submit DLSDU	Master	DL-DATA request	to DLE	DLC-identifier DLSDU-queue-identifier DLSDU-length QoS-set
notify calling peer DLS user	Master	possible DL-DATA confirm	from DLE	DLC-identifier DLSDU-queue-identifier Status
notify called peer and provide DLSDU	Slave	DL-DATA indication	from DLE	DLC-identifier DLSDU-queue-identifier DLSDU-length QoS-set
confirm to calling peer DLS user	Master	DL-DATA confirm	from DLE	DLC-identifier DLSDU-queue-identifier Status

NOTE In this table, time increases from top to bottom.

NOTE In this table, time increases from top to bottom.

26.4.2.3.2 Parameter handling

On GPA channels DLEs handle parameters passed by/to the DLS-user as follows:

26.4.2.3.2.1 DLC-identifier

The Calling DLE transmits DLPDUs via the channel identified by DLC-identifier of the DL-DATA request primitive. Calling DLEs notify their local DLS-user of the status of the transmission of the DLC-identifier of the DL-DATA confirm primitive.

Called DLEs notify their local DLS-user of which channel has received a DLPDU via the DLC-identifier of the DL-DATA indication primitive.

26.4.2.3.2.2 DLSDU-queue-identifier

The Calling DLE retrieves DLSDUs submitted by the local DLS-user from the queue of depth 1 identified by the DLSDU-queue-identifier of the DL-DATA request primitive. Calling DLEs notify their local DLS-user of which DLSDU has been transmitted by the DLSDU-queue-identifier of the DL-DATA confirm primitive.

Called DLEs notify their local DLS-user as to which DLSDU has been received on DLC-identifier via the DLSDU-queue-identifier of the DL-DATA indication primitive.

26.4.2.3.2.3 DLSDU length

The calling DLE uses the DLSDU-length parameter to decide whether to transmit the DLSDU identified by the DLSDU-queue-identifier and to calculate the number-of-grains to be transmitted:

```

IF DLSDU-length > max-DLSDU-length as given in 26.4.2.2.2.3
THEN
    Return status = "data-too-long"
ELSE
    The calling DLE calculates the number-of-grains to be transmitted as specified in 26.4.2.2.2.3
ENDIF
    
```

Max-DLSDU-buffer-size is set by the conformance-class in use. Its range is 64 to 1792 octets in the same increments as shown for max-DLSDU-size in Table 206.

26.4.2.3.2.4 QoS-set

The QoS-set parameter allows

- a) matching of the configured usage of the two peer DLCEPs associated with a GPA or GPC channel and
- b) matching of the calling DLCEP associated with a GPA or GPC channel with its associated DLS-user.
- c) matching by the called DLS-user with its associated DLCEP on a GPA or GPC channel.

Its value may be assigned as 0 to 15 inclusive.

The semantics associated with values of QoS-set are determined by the conformance-class in use rather than by this standard.

NOTE All negotiations of DLS QoS-set are conducted by matching values of QoS-set. There is no concept of one QoS-set value being "included" within another QoS-set value, or of further examination of QoS-set values.

26.4.2.3.2.4.1 At the calling DLE

The calling DLE performs the following logic:

```

IF the Qos-set parameter in the DL-DATA request ≠ quasi-static value of QoS-set for the local DLCEP
associated with the DLC identifier
THEN
    Return status = "QoS-set-mismatch" in the DL-DATA confirm
ELSE
    Copy Qos-set parameter in the DL-DATA request to the QoS-set field of the Initial-grain-header
ENDIF
/* wait for the reverse direction DL-Status DLPDU */
IF other-status = "QoS-set-mismatch" in the reverse direction Status DLPDU
THEN return status = "QoS-set-mismatch" in the DL-DATA confirm
ENDIF

```

26.4.2.3.2.4.2 At the called DLE

The called DLE performs the following logic:

```

IF the Qos-set parameter in the the Initial-grain-header ≠ quasi-static value of QoS-set for the local DLCEP
associated with the DLC identifier of the channel on which the Initial-grain DLPDU was just received
THEN
    Set other-status = "QoS-set-mismatch" in the reverse direction DL-Status DLPDU
ELSE
    Copy the QoS-set parameter in the Initial-grain-header to the QoS-set parameter of the DL-DATA indication
ENDIF

```

NOTE 1 The called DLS-user may check that the value the QoS-set parameter of the DL-DATA indication meets its requirements.

NOTE 2 Detection of DL-status = "Timeout by the calling DLE may indicate a failure to match the QoS-set of called DLS-user and its associated DLCEP

26.4.2.3.2.5 Max-confirm-delay

Max-confirm-delay is a quasi-static parameter that sets an upper bound on the maximum time duration permitted for the completion of a sequence of connection-oriented DLS primitives on a GPA or GPC channel.

It may be configured to the values 10 to 31 max-transfer-periods, inclusive.

NOTE The minimum value allows for both passage through a 4-level bridged network and the round down of max-transfer-period.

Max-transfer-period is the period required to transfer one maximum length DLSDU on a GPC channel on the local link, including the time for transfer of the reverse direction DL-Status DLPDU. Its value is rounded down to 64 Scan-periods.

26.4.2.3.2.6 Status

Status values are determined as follows

- "Data-transferred-correctly" – described in 26.4.2.3.3.1
- "Data-too-long" – described in 26.4.2.1.2.3
- "QoS-set-mismatch" – described in 26.4.2.3.2.4
- "Timeout" – described in 26.4.2.4.2.5.1 and 26.4.2.3.3.1

26.4.2.3.3 Retransmission

Retransmission is used to attempt to guarantee correct transfer of all DLSDUs sent on GPA or GPC channels.

NOTE 1 No DLSDU will be transferred to the called DLS-user in a DL-DATA indication if the called DLE detects any error in the formation or sequence of any of the Grains used to transfer the DLSDU.

NOTE 2 The procedure used will attempt retransmission indefinitely in the presence of errors. This choice was made since the bandwidth allocated to each channel may not be reallocated during normal operation of the Bus.

26.4.2.3.3.1 At the calling DLE

The calling DLE performs the following logic:

```

IF a reverse direction DL-Status DLPDU is received and its transfer-status = FALSE
THEN
    Return status = "Data-transferred-correctly" to the local DLS-user in the DL-DATA confirm
ELSE

```

```

    IF no reverse direction DL-Status DLPDU is received
        WAIT [max-confirm-delay * 3] max-transfer-times, as described in 26.4.2.3.2.5, for receipt of the
        reverse direction DL-Status DLPDU

```

NOTE The quasi-static max-confirm-delay used is that of the DLCEP associated with DLC-identifier of the DL-DATA request.

```

        Return Status = "Timeout" to the local DLS-user in the DL-DATA confirm
    ENDIF
    Retry granulation and transmission of the current DLSDU as described in 26.4.3.1.2.1
ENDIF

```

26.4.2.3.3.2 At the called DLE

The called DLE performs the following logic:

```

IF the just received DLPDU is properly formed, as described in IEC 61158-2
AND the residual of the FCS polynomial is correct

```

NOTE In a properly formed DLPDU, the DLPDU user-data field length is an integral number of octets.

```

THEN
    Extract the DLPDU user-data field from the just received DLPDU
    Save the DLPDU user-data field contents in the queue of depth 1 identified by DLSDU-queue-identifier, as
    described in 26.4.2.3.2.2 and 26.4.2.4.2.2.

```

```

ELSE
    Discard the just received DLPDU

```

```

ENDIF
IF there is any perturbation in the proscribed sequence of received grains, as described in 26.4.3.1.2
THEN
    Set transfer-status = TRUE in the reverse direction DL-Status DLPDU
ENDIF

```

26.4.2.4 GPC class

Data transfers on the GPC class and its interface between the DLE and the DLS-user is shown in Figure 116.

The number of GPC channels per DLSAP is a conformance-class issue. The minimum is 0 and the maximum is 248.

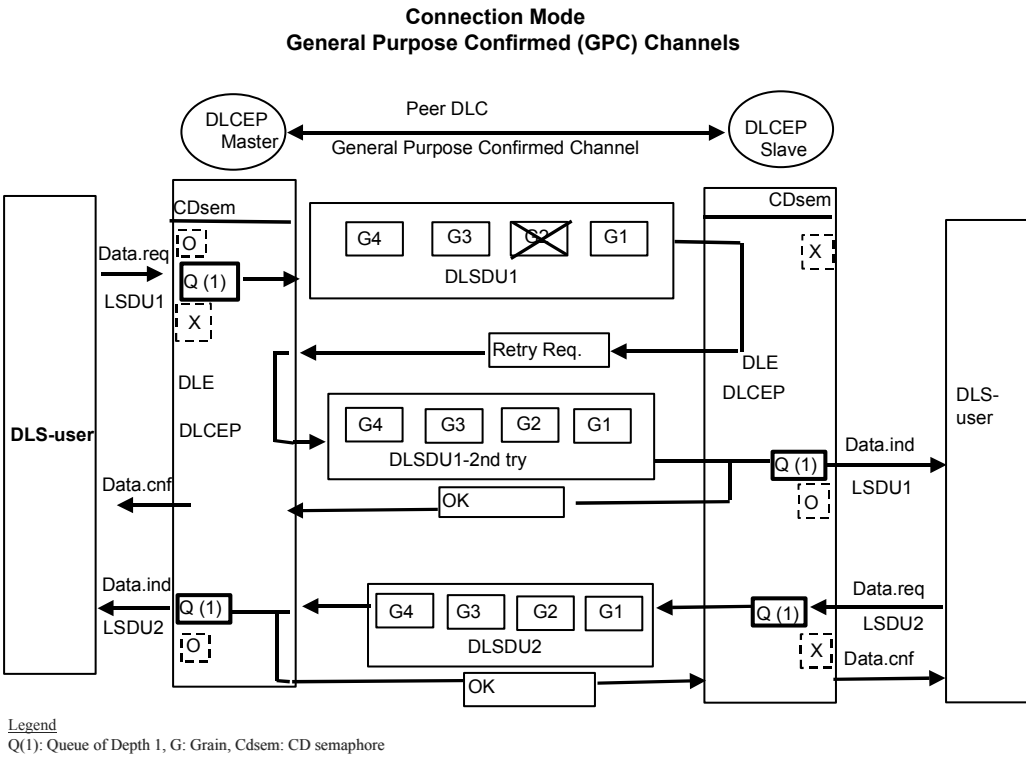


Figure 116 – Data Transfer on the GPC channel-class and its associated primitives

26.4.2.4.1 Primitives and parameters

The primitives and parameters used on GPC channels are shown in Table 208 and Table 209.

Table 208 – Primitives and parameters used for data transfer on GPC channels

Function	Location	Primitive	Direction	Parameters
submit DLSDU and request reply	Master	DL-DATA request	to DLE	DLC-identifier DLSDU-queue-identifier DLSDU-length QoS-set response-request
notify calling peer DLS user	Master	possible DL-DATA confirm	from DLE	DLC-identifier DLSDU-queue-identifier Status
notify called peer, provide DLSDU and request response	Slave	DL-DATA indication	from DLE	DLC-identifier DLSDU-queue-identifier DLSDU-length QoS-set response-request
confirm to calling peer DLS user	Master	DL-DATA confirm	from DLE	DLC-identifier DLSDU-queue-identifier Status

Function	Location	Primitive	Direction	Parameters
submit DLSDU	Slave	DL-DATA request	to DLE	DLC-identifier DLSDU-queue-identifier DLSDU-length QoS-set
notify calling peer DLS user	Slave	possible DL-DATA confirm	from DLE	DLC-identifier DLSDU-queue-identifier Status
notify called peer and provide DLSDU	Master	DL-DATA indication	from DLE	DLC-identifier DLSDU-queue-identifier DLSDU-length QoS-set
confirm to calling peer DLS user	Slave	DL-DATA confirm	from DLE	DLC-identifier DLSDU-queue-identifier Status
NOTE In this table, time increases from top to bottom.				

Table 209 – Primitives and parameters used to disconnect GPC channels

Function	Location	Primitive	Direction	Parameters
submit Disconnect request	Master	DL-DISCONNECT request	to DLE	DLC-identifier
notify peer DLS user	Slave	DL-DISCONNECT indication	from DLE	DLC-identifier
submit Disconnect request	Slave	DL-DISCONNECT request	to DLE	DLC-identifier
notify peer DLS user	Master	DL-DISCONNECT indication	from DLE	DLC-identifier
submit Disconnect request	Slave	DL-DISCONNECT request	to DLE	DLC-identifier
notify peer DLS user	Master	DL-DISCONNECT indication	from DLE	DLC-identifier
submit Disconnect request	Master	DL-DISCONNECT request	to DLE	DLC-identifier
notify peer DLS user	Slave	DL-DISCONNECT indication	from DLE	DLC-identifier
NOTE In this table, time increases from top to bottom.				

26.4.2.4.2 Parameter handling

26.4.2.4.2.1 DLC-identifier

The Calling DLE transmits DLPDUs via the channel identified by DLC-identifier in the DL-DATA request primitive. Calling DLEs notify their local DLS-user of the status of the transmission by the DLC-identifier of the DL-DATA confirm primitive.

Called DLEs notify their local DLS-user of which channel has received a DLPDU via the DLC-identifier in the DL-DATA indication primitive.

The Calling DLE transmits DL-DISCONNECT requests via the channel identified by DLC-identifier in the DL-DISCONNECT request primitive.

Called DLEs notify their local DLS-user of which channel has received a DL-DISCONNECT indication via the DLC-identifier in the DL-DISCONNECT indication primitive.

26.4.2.4.2.2 DLSDU-queue-identifier

Handling of this parameter on GPC channels is described in 26.4.2.3.2.2

26.4.2.4.2.3 DLSDU length

Handling of this parameter on GPC channels is described in 26.4.2.3.2.3

26.4.2.4.2.4 QoS-set

Handling of this parameter on GPC channels is described in 26.4.2.3.2.4

26.4.2.4.2.5 Response-request

This parameter is used by the DLS-user associated with the Master DLCEP on a GPC channel to request and allow the DLS-user associated with the Slave DLCEP to send a reply DLSDU.

Its value may be

- TRUE: a reply DLSDU is requested
- FALSE: no reply DLSDU is requested or allowed

26.4.2.4.2.5.1 At the master DLE

The Master DLE performs the following logic:

```

IF this is a GPC channel
THEN
    Copies the response-request parameter in the DL-DATA request to the CD-semaphore field of the Initial-
    grain-header and to its local CD-semaphore
ELSE /* this is a GPA channel */
    Sets CD-semaphore field of the Initial-grain-header and its local CD-semaphore = FALSE
ENDIF
Granulates and transmits the associated DLSDU as described in 26.4.3.1.2
/* Wait for a reply DLPDU */
IF the Master DLE's local CD-semaphore = TRUE
THEN
    IF a properly formed reply DLSDU arrives from the Slave within max-confirm-delay max-transfer-times.
NOTE   The quasi-static max-confirm-delay used is that of the DLCEP associated with DLC-identifier of the
DL-DATA request.
    THEN
        Set its local CD-semaphore = FALSE
        Pass the reply DLSDU to the local DLS-user in the DL-DATA indication
    ELSE
        Wait again max-confirm-delay max-transfer-times
        Set its local CD-semaphore = FALSE
        Return status = "Timeout" to the local DLS-user in the DL-DATA confirm
    ENDIF
ENDIF
ENDIF

```

26.4.2.4.2.5.2 At the slave DLE

The Slave DLE performs the following logic:

```

IF a properly formed and sequenced DLSDU as described in 26.4.2.3.3.2 is received
THEN
    Copy the CD-semaphore value from the Initial-grain-header to the response-request parameter in the
    DL-DATA indication and to its local CD-semaphore
    Pass this DLSDU to the local DLS-user in the DL-DATA indication
ENDIF
Form and transmit the reverse direction DL-Status DLPDU
IF the CD-semaphore value from the Initial-grain-header = TRUE
THEN
    Prepare to accept a DL-DATA request from the local DLS-user
    IF a DL-DATA request is submitted within [max-confirm-delay - 9] max-transfer-periods
    THEN
        Granulate and transmit the associated DLSDU as described in 26.4.3.1.2
    ELSE
        Set its local CD-semaphore = FALSE
        Do not accept any DL-DATA request from the local DLS-user
    ENDIF
ELSE
    Do not accept any DL-DATA request from the local DLS-user
ENDIF

```

26.4.2.4.2.6 Max-confirm-delay

Max-confirm-delay is a quasi-static parameter that specifies an upper bound on the maximum time duration permitted for the completion of a sequence of connection-oriented DLS primitives on a GPC channel. :

It may be configured to the values 10 to 31 max-transfer-periods, inclusive.

NOTE The minimum value allows for passage through a 4 level Bridged network and the round down of max-transfer-period.

Max-transfer-period is the period required to transfer one maximum length DLSDU on a GPC channel on the local link, including the time for transfer of the reverse direction DL-Status DLPDU. Its value is rounded down to 64 Scan-periods.

26.4.2.4.2.7 Status

Status values are determined as follows

```

"Data-transferred-correctly" – described in 26.4.2.3.3.1
"Data-too-long" – described in 26.4.2.1.2.3
"QoS-set-mismatch" – described in 26.4.2.3.2.4
"Timeout" – described in 26.4.2.4.2.5.1, 26.4.2.3.3.1 and 26.4.2.4.2.5.1

```

26.4.2.4.3 Retransmission

Retransmission on GPC channels is described in 26.4.2.3.3

26.4.3 DLPDU structure, usage and encoding related to channel-methods

Channel Methods define the conveyance of DLS-user data that are common to one or more Channel classes. There are two categories of channel methods:

- a) DLSDU packaging by channels and
- b) DLPDU transfer by channel direction

26.4.3.1 DLSDU packaging channel-methods

Two methods of packaging DLSDUs are available:

- a) UNITARY packaging method – for DLSDUs that are known to be short relative to max-data-length
- b) GRANULAR packaging method – for DLSDUs that may be long relative to max-data-length

26.4.3.1.1 UNITARY method:

The UNITARY method of DLSDU transmission is used exclusively on SCAN and startup channels.

26.4.3.1.1.1 Transmission and reception of short DLSDUs

Short DLSDUs, when transmitted over UNITARY or startup channels, are transmitted in a single Transfer DLPDU.

NOTE 1 With the UNITARY Method, max-data-length is configured during the Bus-configuration process to be $\geq$ the length of all DLSDUs to be transferred on SCAN channels.

As shown in Figure 117, one Transfer DLPDU on a UNITARY or startup channel, transfers the complete DLSDU within a single DLPDU user-data field and a single DLPDU.

NOTE 2 The DLSDU and DLPDU user-data field may be shorter than max-data-length.

Transfer DLPDU composition
single DLPDU user-data field
complete DLSDU

Figure 117 – Transfer DLPDU composition on a UNITARY channel

26.4.3.1.1.1.1 At the publisher DLE

DLPDUs are exclusively transmitted by the Publisher DLE of the associated channel.

When a Publisher DLE transmits a UNITARY DLPDU it notifies its local DLS-user via a DL-BUFFER-SENT confirm.

```

IF Bus-state = NORMAL
  AND No-comm-node for the DLE = FALSE
  AND No-comm-channel for the DLCEP associated with DLC-identifier = FALSE
  AND From-device for the DLCEP associated with DLC-identifier = TRUE
  AND the Bus-cycle-counter is incremented and then matches one of the channel-selectors, as shown
  26.4.3.1.1.1.1.1, for the channel associated with DLC-identifier of the DL-DATA request

```

THEN

Form the single DLPDU with DLPDU user-data field = the entire DLSDU

NOTE The low-order bit of the DLPDU user-data field = the low-order bit of the DLSDU

Transmit the just formed DLPDU

ENDIF

26.1.4.3.1.1.1.1 Channel selectors

Channel-selectors = Channel-offset + [1, 2, ... n]* Scan-period (63)

26.4.3.1.1.1.2 At the subscriber DLE

All Subscriber DLEs associated with a channel attempt to receive DLPDUs on that channel .

When a Subscriber DLE receives a UNITARY DLPDU it notifies its local DLS-user via a DL-BUFFER-RECEIVED indication.

```
IF Bus-state = NORMAL
  AND No-comm-node for the DLE = FALSE
  AND No-comm-channel for the DLCEP associated with DLC-identifier = FALSE
  AND To-device for the DLCEP associated with DLC-identifier = TRUE
  AND the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in
  26.4.3.1.1.1.1, for the channel associated with DLC-identifier of the DL-PUT
```

THEN

Attempt to receive the DLPDU which is present on the link during this channel-selector slot

```
IF the just received DLPDU is properly formed, as described in IEC 61158-2
  AND the residual of the FCS polynomial is correct
```

NOTE 1 In a properly formed DLPDU, the DLPDU user-data field length is an integral number of octets.

THEN

Extract the DLPDU user-data field from the just received DLPDU

Save the DLPDU user-data field contents in the buffer identified by DLSDU-buffer identifier, as described in 26.4.2.1.2.2.

NOTE 2 The low-order bit of the DLPDU user-data field = the low-order bit of the DLSDU

ELSE

Discard the just received DLPDU

ENDIF

ENDIF

26.4.3.1.2 GRANULAR method

The GRANULAR method of data transmission is used for channels where some DLSDUs may not fit within the DLPDU user-data field, which is limited to max-data-length.

DLPDU composition is determined by the length of the DLSDU relative to max-data-length.

If required (on a dynamic basis) DLSDUs are granulated and transmitted in multiple DLPDUs known as Grains.

26.4.3.1.2.1 Granulation and transmission of long DLSDUs

With the GRANULAR method, which is used on ExSCAN, GPA and GPC channels, the DLSDU field is transferred in a sequence of two to 64 Grains.

A reverse direction DL-Status DLPDU is transferred on GPA and GPC channels.

The Grains are potentially:

- a) the Initial-grain DLPDU
- b) zero to 62 Middle-Grain DLPDUs
- c) the Final-Grain DLPDU

The last transmitted Grain is followed on GPA and GPC channels by a:

- d) reverse direction DL-Status DLPDU

26.4.3.1.2.1.1 One Initial-grain

The calling DLE transmits exactly one Initial-grain DLPDU to begin the transfer of a DLSDU passed by its local DLS-user in a DL-DATA request or DL-PUT.

26.4.3.1.2.1.2 Initial-grain composition and transmission

The Initial-grain DLPDU is a sequence of the Initial-grain-header and the Initial-grain.

The Initial-grain DLPDU is shown in Figure 118. If sufficient octets are available, the Initial-grain is the low order [max-data-length -1] octets of the DLSDU.

NOTE The Initial-grain DLPDU may be any permitted length.

The Initial-grain-header is shown in Figure 119.

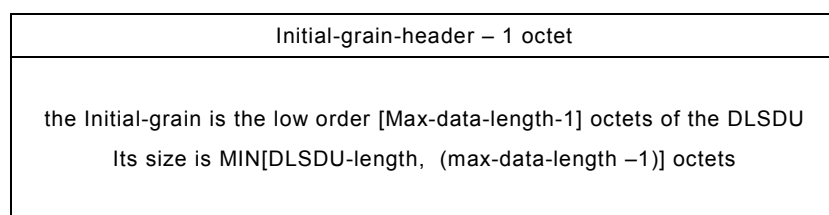


Figure 118 – Initial-grain DLPDU composition

26.1.4.3.1.2.1.2 On GPA and GPC channels

The Calling DLE performs the following logic:

```

IF Bus-state = NORMAL
  AND No-comm-node for the DLE = FALSE
  AND No-comm-channel for the DLCEP associated with DLC-identifier = FALSE
  AND the Calling DLE receives a DL-DATA request /*on a GPA or GPCchannel*/
THEN
  local TGC = 0
  local Final flag = FALSE
  IF [DLSDU-length, as specified in the DL-DATA request, ≤ (max-data-length -1)]
  THEN
    Copy the DLSDU identified by DLSDU-queue-identifier in the DL-DATA request to the DLPDU user-
    data field of the Initial-grain as described in 26.4.2.3.2.2.
    Set the local Final flag = TRUE.
  ELSE
    Copy the low order [max-data-length -1] octets of the DLSDU identified by DLSDU-queue-identifier in
    the DL-DATA request to the DLPDU user-data field of the Initial-grain as described in 26.4.2.3.2.2.
    Set the local Final flag = FALSE.
  ENDIF

```

NOTE The low-order bit of the DLPDU user-data field = the low-order bit of the DLSDU

local TGC = [local TGC + 1]

Build the Initial-grain-header as described below

Form the Initial-grain DLPDU

IF the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in 26.4.3.1.1.1.1.1, for the channel associated with DLC-identifier of the DL-DATA request

THEN

Transmit the Initial-grain in this slot.

ENDIF

ENDIF

26.2.4.3.1.2.1.2 On ExSCAN channels

IF Bus-state = NORMAL

AND No-comm-node for the DLE = FALSE

AND No-comm-channel for the DLCEP associated with DLC-identifier = FALSE

AND From-device for the DLCEP associated with DLC-identifier = TRUE

AND the Publisher DLE receives a DL-PUT /*on an ExSCAN channel*/

THEN

local TGC = 0

local Final flag = FALSE

IF [DLSDU-length, as specified in the DL-PUT, ≤ (max-data-length -1)]

THEN

Copy the low-order octets of the DLSDU identified by DLSDU-buffer-identifier in the DL-PUT to the DLPDU user-data field of the Initial-grain as described in 26.4.2.2.2.

Set the local Final flag = TRUE.

ELSE

Copy the low order [max-data-length -1] octets of the DLSDU identified by DLSDU-buffer-identifier in the DL-PUT to the DLPDU user-data field of the Initial-grain as described in 26.4.2.2.2.

Set the local Final flag = FALSE.

ENDIF

NOTE The low-order bit of the DLPDU user-data field = the low-order bit of the DLSDU

Build the Initial-grain-header as described below

Form the Initial-grain DLPDU

local TGC = [local TGC + 1]

IF the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in 26.4.3.1.1.1.1.1, for the channel associated with DLC-identifier of the DL-PUT

THEN

Transmit the Initial-grain in this slot.

ENDIF

ENDIF

NOTE 5 The low order octet of the DLSDU will be the 2nd octet transmitted in the DLPDU user-data field of the Initial-grain DLPDU

26.4.3.1.2.1.3 Initial-grain-header formation and usage and encoding

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PDU-type		Disconnect request	CD-semaphore	QoS-set			
0	0	TRUE = Disconnect requested FALSE = no request	TRUE = Slave DLS-user may transmit a reply DLPDU FALSE = Slave DLS-user may not transmit a reply DLPDU	0 to 15			

Figure 119 – Initial-grain header composition

The Initial-grain-header is formed by the Calling DLE and conveys the DLPDU-type, CD-semaphore and QoS-set.

a) DLPDU-type

DLPDU-type = 0 0

b) CD-semaphore (allowed for GPC channels only)

The setting and usage of the CD-semaphore is described in 26.4.2.4.2.5

TRUE – Slave DLS-user may transmit a reply DLPDU as described in 26.4.2.4.2.5.

FALSE – Slave DLS-user may not transmit any DLPDU

NOTE 1 To avoid collisions, the Slave DLE enforces the CD-semaphore setting by preventing a Slave DLE that does not hold the CD-semaphore from transmitting DLPDUs.

NOTE 2 The setting of the CD-semaphore does not influence whether a reverse direction DL-Status DLPDU will be sent by the called DLE after receipt of the Final-grain.

c) QoS-set

The setting, values and usage of the QoS-set is described in 26.4.2.3.2.4

d) Disconnect request (allowed for GPC channels only)

TRUE – Calling DLS-user has requested a DL-DISCONNECT as described in 26.4.2.4.1.

FALSE – Calling DLS-user has not requested a DL-DISCONNECT

NOTE 3 Receipt of a DL-DISCONNECT request disables all other reporting associated with the same DLPDU.

26.4.3.1.2.1.4 Zero to 62 middle-grains

zero to 62 Middle-grain DLPDUs are transmitted by the calling DLE to transfer the middle part of a DLSDU.

26.4.3.1.2.1.5 Middle-grain composition and transmission

The intermediate Transfer DLPDUs for a DLSDU contain Middle-grains, each of which is a sequence of the Middle-grain-header and the Middle-grain.

The Middle-grain DLPDU is shown in Figure 120. If sufficient octets are available, the Middle-grain is the low order [Max-data-length-1] octets of the remainder of the DLSDU, continuing from low order octet.

The Middle-grain-header is shown in Figure 121.

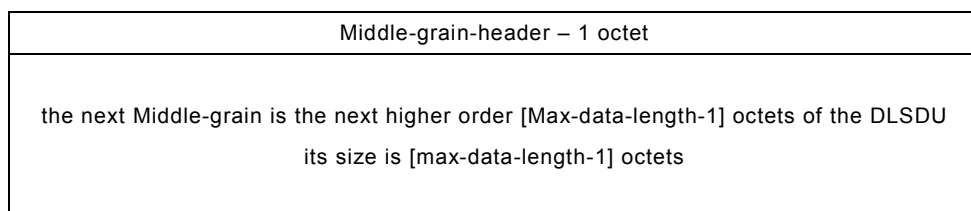


Figure 120 – Middle-grain DLPDU composition

The Calling DLE performs the following logic:

IF local Final flag = FALSE
AND local TGC ≥ 1

THEN

IF [remaining DLSDU-length ≤ (max-data-length - 1)]

THEN

Set the local Final flag = TRUE.

NOTE 1 A Middle-grain DLPDU is always of maximal length for its slot.

ELSE

Copy the low order (max-data-length -1) octets of the current DLSDU to the DLPDU user-data field of the Middle-grain.

Set the local Final flag = FALSE.

ENDIF

Build the Middle-grain-header as described below.

Form the Middle-grain DLPDU

local TGC = [local TGC + 1]

Copy local TGC to the TGC field of the Middle-grain-header

Transmit a Middle-grain DLPDU one Scan-Period after transmission of the previous grain.

ENDIF

NOTE 2 No Middle-grains are transmitted when number-of-grains is two or less.

26.4.3.1.2.1.6 Middle-grain-header formation and usage and encoding

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PDU-type		Transmitted-grain-count (TGC)					
0	1	values 1 to 62 are allowed					

Figure 121 – Middle-grain header composition

The Middle-grain-header is formed by the Calling DLE and conveys DLPDU-type and Transmitted-grain-count.

a) DLPDU-type

DLPDU-type = 0 1

b) Transmitted-grain-count (TGC)

Transmitted-grain-count TGC = the number of Grains previously transmitted from this DLSDU, excluding this grain.

NOTE TGC's allowed range of values is 1 to 62 in the Middle-grain-header

26.4.3.1.2.1.7 One final-grain

The calling DLE transmits exactly one Final-grain DLPDU to complete the transfer of a DLSDU.

26.4.3.1.2.1.8 Final-grain composition and transmission

The Final-grain is a sequence of the Final-grain-header and the Final-grain; it is shown in Figure 122.

The Final-grain conveys the high order remainder of the DLSDU, if any remainder exists after the previous grain.

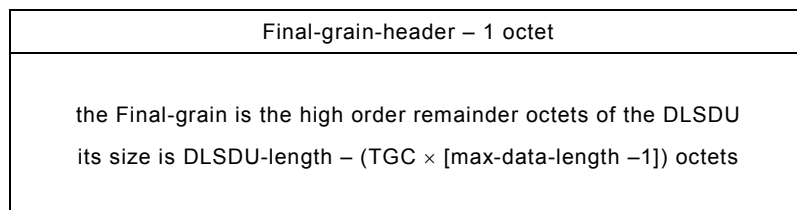


Figure 122 – Final-grain DLPDU composition

The Calling DLE performs the following logic:

IF local Final flag = TRUE
AND local TGC $\geq$ 1

THEN

Copy the remaining octets of the current DLSDU to the DLPDU user-data field of the Final-grain.

NOTE 1 The Final-grain DLPDU may be any permitted length.

local TGC = [local TGC + 1]

Copy the local TGC to the TGC field of the Final-grain-header

Build the Final-grain-header as described below.

Form the Final-grain DLPDU

NOTE 2 Local TGC now = [number-of-grains, from 26.4.2.2.2.3, - 1].

Transmit the Final-grain DLPDU one Scan-period after transmission of the previous grain.

ENDIF

26.4.3.1.2.1.9 Final-grain-header formation and usage and encoding

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PDU-type		Transmitted-grain-count (TGC)					
1	0	values 1 to 63 are allowed					

Figure 123 – Final-grain header composition

The Final-grain-header is formed by the Calling DLE and conveys DLPDU-type and Transmitted-grain-count, as shown in Figure 123.

a) DLPDU-type

DLPDU-type = 1 0

NOTE 1 This is the ending grain of this multi-grain DLSDU. It is a "marker" for the End Of Transfer.

b) Transmitted-grain-count

NOTE 2 TGC's allowed range of values is 1 to 63 in the Final-grain-header

26.4.3.1.2.2 Reverse direction DL-status DLPDU

A reverse direction DL-Status DLPDU follows the Final-grain on GPA and GPC channels.

This DLPDU conveys the status of the just completed transmission.

26.4.3.1.2.2.1 DL-status DLPDU composition and transmission

The DL-Status DLPDU consists solely of a DL-Status-header

The DL-Status DLPDU is transmitted by the called DLE one Scan-period after receipt of the Final-grain.

The DL-Status DLPDU, and DL-Status-header are shown in Figure 124.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
PDU-type		Transfer-status	Disconnect request	Other-status			
1	1	Ack = FALSE Nak = TRUE	TRUE = Disconnect requested FALSE = no request	defined by conformance-class			

Figure 124 – DL-status-header and DL-status DLPDU

The Called DLS executes the following logic

IF the Final-grain is received on a GPA or GPC channel

THEN

IF there is any perturbation in the proscribed sequence of the just received grains, as described in 26.4.3.1.2.

THEN

set transfer-status = TRUE in the DL-Status-header

ELSE

Set transfer-status = FALSE in the DL-Status-header

ENDIF

Build the DL-Status-header as described below.

Form the DL-Status DLPDU .

NOTE The length of an unexpanded DL-Status DLPDU is one octet.

Transmit the DL-Status DLPDU one Scan-Period after reception of the Final grain.

ENDIF

26.4.3.1.2.2.2 DL-status-header formation and usage and encoding

The DL-Status-header is formed by the Called DLE and conveys DLPDU-type, transfer-status and other-status

a) DLPDU-type

DLPDU-type = 1 1

b) transfer-status

FALSE — the just received DLSDU was received correctly

TRUE — There has been a perturbation in the sequential receipt of the Grains of the just received DLSDU

NOTE 1 This is the transfer-status of the just received DLSDU.

c) other-status

This encoding is established by conformance-class. The specific choice of values does not affect interoperability.

NOTE 2 This is the other-status of the just received DLSDU.

d) Disconnect request (allowed for GPC channels only)

TRUE – Called DLS-user has requested a DL-DISCONNECT as described in 26.4.2.4.1.

FALSE – Called DLS-user has not requested a DL-DISCONNECT

NOTE 3 Receipt of the DL-DISCONNECT request disables all other reporting associated with the same DLPDU.

26.4.3.1.2.3 Reception and reassembly of long DLSDUs

26.4.3.1.2.3.1 On GPA and GPC channels

The Called DLE on a GPA or GPC channel performs the following logic:

```

IF Bus-state = NORMAL
  AND No-comm-node for the DLE = FALSE
  AND No-comm-channel for the DLCEP associated with DLC-identifier = FALSE
  AND local Partial flag = FALSE
  AND local Error flag = FALSE
  AND the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in
  26.4.3.1.1.1, for the channel associated with the DLC-identifier of the DL-DATA request

```

NOTE 1 At Power-on and Reset the local Partial flag is initialized = FALSE and the local Error flag is initialized = FALSE

THEN

Attempt to receive the DLPDU which is present on the link during this channel-selector slot

IF the just received DLPDU is properly formed, as described in IEC 61158-2
AND the residual of the FCS polynomial is correct

NOTE 2 In a properly formed DLPDU, the DLPDU user-data field length is an integral number of octets.

THEN

Extract the DLPDU user-data field from the just received DLPDU

Save the DLPDU user-data field contents in the queue identified by DLSDU-queue identifier, as described in 26.4.2.3.2.2.

NOTE 3 The low-order bit of the DLPDU user-data field = the low-order bit of the DLSDU

ELSE

Discard the just received DLPDU

ENDIF

IF the just received DL PDU is an Initial-grain DLPDU, as described in 26.4.3.1.2.1.1, is received

THEN

Set local Partial flag = TRUE

Set local First flag = TRUE

Copy QoS-set from the Initial-grain-header to the local QoSet variable

Copy the 2nd through last octets of the Initial-grain DLPDU to the low-order position in the queue of depth one which will be used hold the received DLSDU.

NOTE 4 The 2nd through last octets of the Initial-grain DLPDU correspond to the first grain, which was extracted from the DLSDU being transmitted, as described in 26.4.3.1.2.1.1.

IF this is a GPC channel

Copy CD_semaphore from the Initial-grain-header to the local response-request variable

ENDIF

ELSE

Discard the DLPDU

Set local Partial flag = FALSE

Set local First flag = FALSE

Set local Error flag = TRUE

ENDIF

ENDIF /* end of Initial grain*/

IF local Partial flag = TRUE

AND local Error flag = FALSE

AND a Middle-grain DLPDU, as described in 26.4.3.1.2.1.5, is received, as described above for the Initial-grain

THEN

IF the local First flag = TRUE
 AND TGC from the Middle-grain-header = 1
 OR the local First flag = FALSE
 AND TGC from the Middle-grain-header > 1
 AND TGC from the Middle-grain-header = [local TGC variable + 1]

THEN

Copy TGC from the Middle-grain-header to the local TGC variable

 Copy the 2nd through last octets of the Middle-grain DLPDU to the next-lowest-order position in the queue of depth one which will be used hold the received DLSDU.

 Set local First flag = FALSE

NOTE 5 The 2nd through last octets of the Middle-grain DLPDU correspond to the next grain that was extracted from the DLSDU being transmitted, as described in 26.4.3.1.2.1.5.

ELSE

Discard the DLPDU

 Set local Partial flag = FALSE

 Set local First flag = FALSE

 Set local Error flag = TRUE

ENDIF

ELSE

Discard the DLPDU

 Set local Partial flag = FALSE

 Set local First flag = FALSE

 Set local Error flag = TRUE

ENDIF /* end of Middle grain*/

IF local Partial flag = TRUE
 AND local Error flag = FALSE
 AND a Final-grain DLPDU, as described in 26.4.3.1.2.1.8, is received, as described above for the Initial-grain

THEN

IF the local First flag = TRUE
 AND TGC from the Final-grain-header = 1
 OR the local First flag = FALSE
 AND TGC from the Final-grain-header > 1
 AND TGC from the Final-grain-header = [local TGC variable + 1]

THEN

Copy the 2nd through last octets of the Final-grain DLPDU to the next-lowest order position in the queue of depth one which will be used hold the received DLSDU.

 Set local First flag = FALSE

NOTE 6 The 2nd through last octets of the Final-grain DLPDU correspond to the last grain that was extracted from the DLSDU being transmitted, as described in 26.4.3.1.2.1.8.

ELSE

Discard the DLPDU

 Set local Partial flag = FALSE

 Set local First flag = FALSE

 Set local Error flag = TRUE

ENDIF

ELSE

Discard the DLPDU

 Set local Partial flag = FALSE

 Set local First flag = FALSE

 Set local Error flag = TRUE

ENDIF /* end of Final grain*/

IF local Error flag = FALSE

THEN

Place the DLC-identifier for the DLC on which this DLSDU was received in the DLC-identifier parameter of the upcoming DL-DATA indication

Place the DLSDU-queue-identifier for the queue of depth one in which this DLSDU was received in the DLSDU-queue-identifier parameter of the upcoming DL-DATA indication

Calculate the length of the just received DLSDU

Place the DLSDU-length of the just received DLSDU in the DLSDU-length parameter of the upcoming DL-DATA indication

Place the local QoSet variable in the QoS-set parameter of the upcoming DL-DATA indication

IF this is a GPC channel

Place the local response-request variable in the response-request parameter of the upcoming DL-DATA indication

NOTE 7 The procedure by which a Slave (Called) DLE may send a reply DLSDU is described in 26.4.2.4.2.4.2.

ENDIF

Pass the DL-DATA indication to the local DLS-user

Set transfer-status = FALSE in the DL-Status-header

ELSE

Discard the DLSDU

Set transfer-status = TRUE in the DL-Status-header

ENDIF

Transmit the DL-Status DLPDU one Scan-period after reception of the Final-grain

Set local Error flag = FALSE

Set local Partial flag = FALSE

26.4.3.1.2.3.2 On ExSCAN channels

The Called DLE on an ExSCAN channel performs the following logic:

IF Bus-state = NORMAL

AND No-comm-node for the DLE = FALSE

AND No-comm-channel for the DLCEP associated with DLC-identifier = FALSE

AND To-device for the DLCEP associated with DLC-identifier = TRUE

AND local Partial flag = FALSE

AND local Error flag = FALSE

AND the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in 26.4.3.1.1.1.1, for the channel associated with the DLC-identifier of the DL-PUT

NOTE 1 At Power-on and Reset the local Partial flag is initialized = FALSE and the local Error flag is initialized = FALSE

THEN

Attempt to receive the DLPDU which is present on the link during this channel-selector slot

IF the just received DLPDU is properly formed, as described in IEC 61158-2

AND the residual of the FCS polynomial is correct

NOTE 2 In a properly formed DLPDU, the DLPDU user-data field length is an integral number of octets.

THEN

Extract the DLPDU user-data field from the just received DLPDU

Save the DLPDU user-data field contents in the buffer identified by DLSDU-buffer identifier, as described in 26.4.2.2.2.2.

NOTE 3 The low-order bit of the DLPDU user-data field = the low-order bit of the DLSDU

ELSE

Discard the just received DLPDU

ENDIF

IF the just received DLPDU is an Initial-grain DLPDU, as described in 26.4.3.1.2.1.3

THEN

Set local Partial flag = TRUE

Set local First flag = TRUE

Copy the 2nd through last octets of the Initial-grain DLPDU to the low-order position in the buffer which will be used to hold the received DLSDU.

NOTE 4 The 2nd through last octets of the Initial-grain DLPDU correspond to the first grain that was extracted from the DLSDU being transmitted, as described in 26.4.3.1.2.1.1.

ELSE

Discard the DLPDU

Set local Partial flag = FALSE

Set local First flag = FALSE

Set local Error flag = TRUE

ENDIF

ENDIF /* end of Initial Grain*/

IF local Partial flag = TRUE

AND local Error flag = FALSE

AND a Middle-grain DLPDU, as described in 26.4.3.1.2.1.5.5, is received, as described above for the Initial-grain

THEN

IF the local First flag = TRUE

AND TGC from the Middle-grain-header = 1

OR the local First flag = FALSE

AND TGC from the Middle-grain-header > 1

AND TGC from the Middle-grain-header = [local TGC variable + 1]

THEN

Copy TGC from the Middle-grain-header to the local TGC variable

Copy the 2nd through last octets of the Middle-grain DLPDU to the next-lowest-order position in the buffer which will be used to hold the received DLSDU.

Set local First flag = FALSE

NOTE 5 The 2nd through last octets of the Middle-grain DLPDU correspond to the next grain that was extracted from the DLSDU being transmitted, as described in 26.4.3.1.2.1.5.

ELSE

Discard the DLPDU

Set local Partial flag = FALSE

Set local First flag = FALSE

Set local Error flag = TRUE

ENDIF

ELSE

Discard the DLPDU

Set local Partial flag = FALSE

Set local First flag = FALSE

Set local Error flag = TRUE

ENDIF /* end of Middle grain*/

IF local Partial flag = TRUE

AND local Error flag = FALSE

AND a Final-grain DLPDU, as described in 26.4.3.1.2.1.8, is received, as described above for the Initial-grain

THEN

IF the local First flag = TRUE

AND TGC from the Final-grain-header = 1

OR the local First flag = FALSE

AND TGC from the Final-grain-header > 1

AND TGC from the Final-grain-header = [local TGC variable + 1]

THEN

Copy the 2nd through last octets of the Final-grain DLPDU to the next-lowest order position in the buffer which will be used to hold the received DLSDU.

Set local First flag = FALSE

NOTE 6 The 2nd through last octets of the Final-grain DLPDU correspond to the last grain that was extracted from the DLSDU being transmitted, as described in 26.4.3.1.2.1.8.

ELSE

Discard the DLPDU

Set local Partial flag = FALSE

Set local First flag = FALSE

Set local Error flag = TRUE

ENDIF

ELSE

Discard the DLPDU

Set local Partial flag = FALSE

Set local First flag = FALSE

Set local Error flag = TRUE

ENDIF /* end of Final grain*/

IF local Error flag = FALSE

THEN

Place the DLC-identifier for the DLC on which this DLSDU was received in the DLC-identifier parameter of the upcoming DL-GET and DL-BUFFER-RECEIVED primitives.

Place the DLSDU-buffer-identifier for the buffer in which this DLSDU was received in the DLSDU-buffer-identifier parameter of the upcoming DL-GET

Calculate the length of the just received DLSDU

Place the DLSDU-length of the just received DLSDU in the DLSDU-length parameter of the upcoming DL-GET

Pass a DL-BUFFER-RECEIVED to the local DLS-user

IF the local DLS-user requests the just received DLSDU

Pass the just prepared DL-GET to the local DLS-user

ENDIF

ELSE

Discard the DLSDU

ENDIF

Set local Error flag = FALSE

Set local Partial flag = FALSE

26.4.3.2 Channel direction method and DLPDU transmission order

Two methods of controlling the direction of DLPDU transmission are available.

a) 1WAY method

b) 2WAY method

26.4.3.2.1 1WAY method

In the 1WAY method, all Transfer DLPDUs that are associated with one DLSDU flow in the same direction.

The DLPDU transmission format shown for both the 1 WAY and the 2 WAY Transmission method is shown in Figure 125.

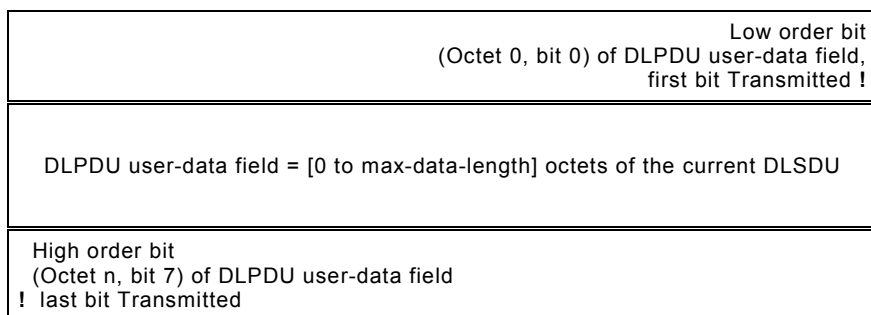


Figure 125 – DLPDU transmission format for 1 WAY or 2 WAY transmission method

The 1WAY method allows configurable unidirectional communication to or from a “field device”.

26.4.3.2.1.1 Direction of DLPDU transmission

On 1WAY channels the direction of DLPDU transmission is:

- a) Constant
- b) either From-device, for the Publisher, or To-device, for a Subscriber
- c) Controlled by channel configuration

26.4.3.2.2 2WAY method

In the 2WAY method Transfer DLPDUs may flow in both directions on a channel , as directed by

- a) the Calling and Called DLEs on GPA or GPC channels
- b) the Master and Slave DLEs, and indirectly by their local DLS-users, on GPC channels

The DLPDU transmission format shown in Figure 125 applies to both the 1 WAY and 2 WAY Transmission Methods

The 2WAY method is required:

- c) on GPA and GPC channels for transmission of the DL-Status DLPDU and
- d) on GPC channels for transmission of reply DLPDUs by the Slave DLE.

26.4.3.2.2.1 Direction of DLPDU and grain flow

On 2WAY channels the direction of DLPDU transmission may be viewed in two ways:

- a) Constant during the transmission of all Grains of a DLSDU, then reversed for the DL-Status DLPDU, when controlled by the Calling and Called DLEs on GPA or GPC channels, as described in 26.4.3.1.2.
- b) Starts from the Master DLE when controlled by the Master DLE on a GPC or GPA channel. The Master DLE may grant the right to transmit to the Slave DLE on a GPC channel by setting the CD-semaphore = TRUE, as described in 26.4.2.4.2.5.
- c) Master status is established by channel configuration.

26.4.4 DLC-identifiers (channel-IDs)

DLCs are accessed by transmitting or listening, as described in 26.2 and shown in Figure Figure 107, during the slot time that corresponds to each of the slots that compose it. This sequence of slots, the channel-selectors, is represented by a channel-identifier as shown in Figure 107.

The DLC-identifier (for example, Chan-ID=8 in Figure 107) identifies one Channel (DLC) from within all Channels on a any link.

26.4.4.1 Local link form of DLC-identifier

The DLC-identifier of any channel on a local link is the first slot selector (after the Bus-Cycle-End slot) assigned to this channel.

DLEs assigned to a channel send or receive DLPDUs, as described in 26.4.3.1.1.1.0 and shown in Figure 108, in each slot selected by that channel's channel-selectors, which are established during the Bus Configuration process.

NOTE No DL-addresses are used with the connection-mode protocol once a DLC has been established. The actual establishment uses DLM-connectionless-mode protocol elements, which do use DL-addresses.

The relationship between each DLC-identifier and the DLCEP-addresses used for configuration of this channel is known to, and managed by, the Bus Configurator.

26.4.4.2 Extended link form of DLC-identifier

The DLC-identifier, of any (local) link on the extended link, is the slot number of the first slot selector (after the Bus-Cycle-End slot) assigned to this channel on this link.

The relationship between each DLC-identifier and the DLCEP-addresses used for configuration of this channel is known to, and managed by, the Bus Configurator.

26.4.4.2.1 Virtual network topology

Virtual networks are networks where some lower level DLCs are not visible on the TOP or other higher levels of the extended link and vica-versa. Virtual topologies apply only to connection mode operation. Virtual topologies are invisible to the operation of non-involved DLEs.

EXAMPLE Figure 126 shows an example of a real network and two possible superimposed virtual networks, VN1 and VN2. In these two virtual networks, DLCEPs in N6, N2 and N3 are visible to each other, but not to DLCEPs in any other DLEs; also DLCEPs in N4 and N5 are visible to each other, but not to DLCEPs in any other DLEs.

Virtual topologies are set up during the Bus-configuration process.

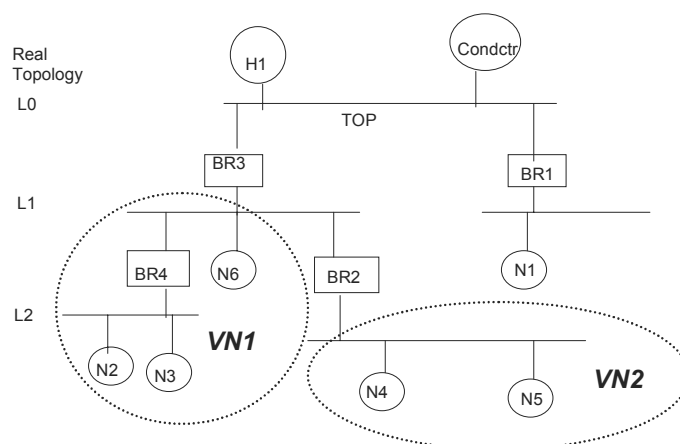


Figure 126 – EXAMPLE: Three level real network with two superimposed virtual networks

NOTE 1 Max-net-Level limits the depth of the maximum nest of links in the real topology. It does not limit the depth of the maximum nest of virtual topology channels.

NOTE 2 Virtual topologies are useful for increasing the total number of channels serviced by an extended-link.

26.5 DLM-connectionless service

The DLM-connectionless service is an adaptation to the constraints of this protocol of the DLM-connectionless service defined in ISO/IEC 8886 and in IEC 61158-3, Type 1. It is restricted to DL-management and is used solely during the START and CHECK states, where a standardized Bus-Config (a standardized set of bus timing parameters) that supports DLM-connectionless service exists. It is used for configuration of all DLEs, channels, triggers and half-forwarding-ports.

NOTE Future versions of this standard may define additional uses of the DLM-connectionless service.

26.5 provides a conceptual definition of the services provided by the DLS-provider to the DLMS-user(s). Nothing in this section shall be construed to constrain the actual implementations of the interactions at the DLS-provider to DLMS-user interface.

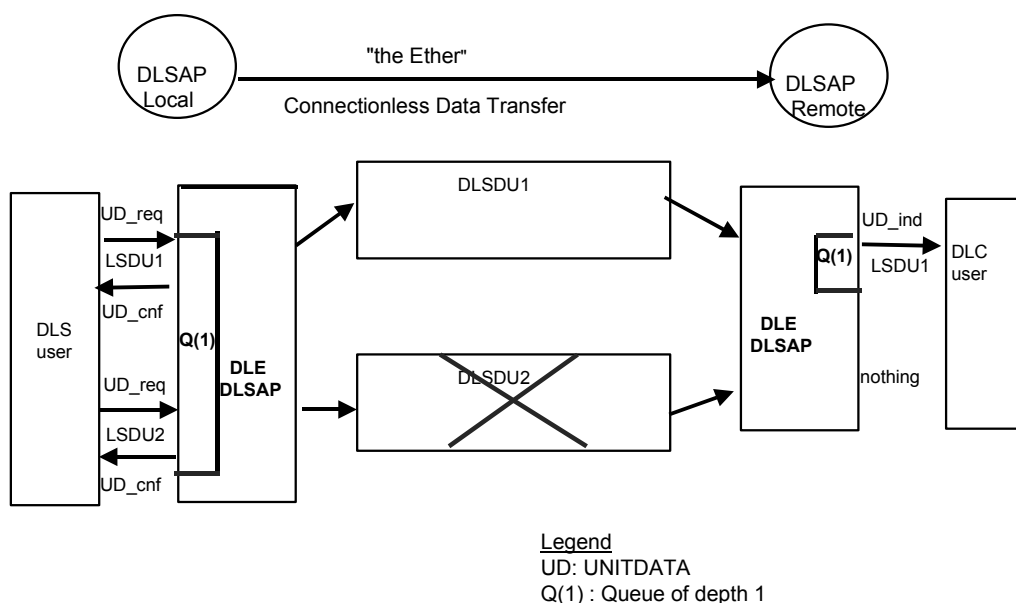


Figure 127 – Operation of DLM-connectionless service and its primitives

The operation and primitives of the DLM-connectionless service are shown in Figure 127.

26.5.1 DLMS-user interface and related operation for DLM-connectionless service

26.5.1.1 Primitives and parameters

The primitives and parameters used for DLM-connectionless service are shown in Table 210 and Figure 127:

Table 210 – Primitives and parameters used for DLM-connectionless service

Function	Location	Primitive	Direction	Parameters
submit DLMSDU	Local DLSAP	DLM-UNITDATA request	to DLE	DLC-identifier DLMSDU-queue-identifier DLMSDU-length Calling Address Called Address
notify calling peer DLMS user	Local DLSAP	DLM-UNITDATA confirm	from DLE	DLC-identifier DLMSDU-queue-identifier Status
notify called peer and provide DLMSDU	Remote DLSAP	DLM-UNITDATA indication	from DLE	DLC-identifier DLMSDU-queue-identifier DLMSDU-length Calling Address Called Address
NOTE In this table, time increases from top to bottom.				

26.5.1.2 Parameter handling

26.5.1.2.1 DLC-identifier

The Calling DLE transmits DLPDUs via the channel identified by DLC-identifier in the DLM-UNITDATA request primitive. Calling DLEs notify their local DLMS-user of the status of the transmission by the DLC-identifier in the DLM-UNITDATA confirm primitive.

Called DLEs notify their local DLMS-user of which channel has received a DLPDU via the DLC-identifier in the DLM-UNITDATA indication primitive.

NOTE Startup channels are used to convey DLM-connectionless DLPDUs, as described in 26.5.2.3

26.5.1.2.2 DLMSDU-queue-identifier

The Calling DLE retrieves DLMSDUs submitted by the local DLMS-user from the queue of depth 1 identified by the DLMSDU-queue-identifier of the DLM-UNITDATA request primitive. Calling DLEs notify their local DLMS-user of which DLMSDU has been transmitted by the DLMSDU-queue-identifier of the DLM-UNITDATA confirm primitive.

Called DLEs notify their local DLMS-user as to which DLMSDU has just been received on DLC-identifier via the DLMSDU-queue-identifier of the DLM-UNITDATA indication primitive.

26.5.1.2.3 DLMSDU length

The calling DLE uses the DLMSDU-length parameter to decide whether to transmit the DLMSDU identified by the DLMSDU-queue-identifier :

```

IF DLMSDU-length > (max-data-length – 12)
THEN
    Return status = "data-too long"
ENDIF

```

26.5.1.2.4 Calling address

The calling DLE maps the calling address of the just passed DLM-UNITDATA request to the SA of the DLPDU it is forming.

The called DLE maps the SA of the just received DLPDU to the calling address of the upcoming DLM-UNITDATA indication.

26.5.1.2.5 Called address

The calling DLE maps the called address of the just passed DLM-UNITDATA request to the DA of the DLPDU it is forming.

The called DLE maps the DA of the just received DLPDU to the called address of the upcoming DLM-UNITDATA indication.

26.5.2 DLPDU structure, usage and encoding related to DLM-connectionless operation

The structure of DLM-connectionless DLPDUs is shown in Figure 128.

Item	Length (octets)	Encoding	Allowed Values
Destination Address (DA)	4 to 6	OCTETSTRING	see 26.5.3
Source Address (SA)	4 to 6	OCTETSTRING	see 26.5.3
DLPDU user-data field	0 to max-data-length – 12	OCTETSTRING	any
High order bit (Octet n, bit 7) of DLPDU ! last bit Transmitted!			

Figure 128 – DLM-connectionless DLPDU composition and encoding

26.5.2.1 Transmission and reception of DLM-connectionless DLMSDUs

DLM-connectionless DLMSDUs are transmitted and received using the procedures for short DLSDUs, as described in 26.4.3.1.1.1.

26.5.2.1.1 At the calling DLE

The Local/Calling DLE performs the following logic for the DLM-connectionless service.

IF the Calling DLE receives a DLM-UNITDATA request from the local DLMS-user of the DLM-connectionless service

AND the Bus-cycle-counter is incremented and then matches one of the startup-channel-selectors, described in 26.4.3.1.1.1.1, for the startup-channel temporarily associated with the DLE.

NOTE 1 Startup-channels are temporarily assigned to match DLE UUIDs during the startup process , as described in 26.5.2.3

THEN

Form the single DLM-connectionless DLPDU, as described in 26.5.2, with:

DA = called address from the just passed DLM-UNITDATA request

SA = calling address from the just passed DLM-UNITDATA request

DLPDU user-data field = the entire DLMSDU from the just passed DLM-UNITDATA request

NOTE 2 The low-order bit of the DLPDU user-data field = the low-order bit of the DLMSDU

Transmit the just formed DLPDU on the above startup-channel.

ENDIF

26.5.2.1.2 At the called DLE

The Remote/Called DLE performs the following logic for the DLM-connectionless service:

IF the Bus-cycle-counter is incremented and then matches one of the startup-channel-selectors, described in 26.5.2.2, for the startup-channel temporarily associated with the DLE.

NOTE 1 Startup-channels are temporarily assigned to match DLE UUIDs during the startup process, as described in 26.5.2.3

THEN

Attempt to receive the DLM-connectionless DLPDU, as described in 26.5.2, which is present on startup-channel during this startup-channel-selector slot

IF the just received DLPDU is properly formed, as described in IEC 61158-2
AND the residual of the FCS polynomial is correct

NOTE 2 In a properly formed DLM-connectionless DLPDU, the DA length, SA length and DLPDU user-data field length are all an integral number of octets.

THEN

Extract the DA, SA and DLPDU user-data field from the just received DLPDU

Save the:

DA as the called address of the upcoming DLM-UNITDATA indication

SA as the calling address of the upcoming DLM-UNITDATA indication

DLPDU user-data field contents in the queue of depth 1 identified by the DLMSDU-queue-identifier of the upcoming DLM-UNITDATA indication

NOTE 3 The low-order bit of the DLPDU user-data field = the low-order bit of the DLMSDU

Pass the DLM-UNITDATA indication to the local DLM-connectionless service DLMS-user

ELSE

Discard the just received DLPDU

ENDIF

ENDIF

26.5.2.2 Standardized bus timing parameters for DLM-connectionless operation

During the START and CHECK states, the Conductor on the each link establishes standardized bus timing parameters for DLM-connectionless operation. This set of conditions constitutes the "ether" needed for the DLM-connectionless service. They are:

a) InterFrameGap = 35 bit-times

b) max-data-length = 33 octets

c) Bus-sync-period = 32 slot-times

NOTE 1 The Bus-sync channel is not offered, and may not be claimed, as a startup-channel

d) 31 startup-channels are simultaneously, implicitly offered by the Conductor to non-Conductor DLEs

e) startup-scan-period = 32 slot-times

f) Startup-channel-selectors = Startup-channel-offset + [1, 2, ... n]* startup-scan-period

g) During the START and CHECK states, each non-Conductor DLE attempts to claim the startup-channel that uses the Offset-slot that equals the lower four bits of its UUID.

NOTE 2 Thus there is at least one startup-channel that may be claimed by each DLE

NOTE 3 The procedures of this subclause prevent collisions during DLM-connectionless operation.

The Node and Node-check DLPDUs are transmitted and received using the same procedures that are used for other DLM-connectionless services, as described below in this subclause.

26.5.2.3 Assignment of startup-channels for DLM-connectionless operation

Startup-channels are temporarily and sequentially assigned to a specific non-Conductor node by cooperative action of the Conductor and each non-Conductor node.

26.5.2.3.1 At the conductor

The Conductor performs the following logic on each of the startup channels during the START and CHECK states.

NOTE 1 The activity below may be concurrent over several or all offered startup-channels.

IF the Conductor enters the START state

THEN

the Conductor sets its Node-configuration variable for each startup-channel = EMPTY

the Conductor sets its Node-address variable for this startup-channel = NULL

The Bus-configuration DLMS-user forms a Node DLMSDU for the first node associated, in the Node-list, with each offered startup-channel, as described in 26.5.2.3.1.1.

The Bus-configuration DLMS-user passes the just formed Node DLMSDU to the DLE in a DLM-UNITDATA request with:

Called-address = all 0s

Calling-address = a local identifier for the the DLE's own Individual SA

DLMSDU-queue-identifier pointing to the just formed DLMSDU

The DLE forms a Node DLPDU for each newly passed DLM-UNITDATA request. The contents of this DLM-connectionless DLPDU, described in 26.5.2, are:

DA = all 0s.

SA = the DLE's own Individual SA

DLPDU user-data field = Node DLMSDU

The DLE transmits each just formed Node DLPDU on all the above startup-channels, as described in 26.5.2.1.1

ENDIF

IF the Conductor has just completed configuration of a node, as indicated by exhaustion of the entries for this node in the Node-list

THEN

the Conductor sets its Node-configuration variable for the startup-channel assigned to this node = EMPTY

the Conductor sets its Node-address variable for this startup-channel = NULL

NOTE 2 The above actions release the startup-channel that had been temporarily assigned to the just configured node.

The Bus-configuration DLMS-user forms a Node DLMSDU for the next node associated, in the Node-list, with each offered startup-channel, as described in 26.5.2.3.1.1.

The Bus-configuration DLMS-user passes the just formed Node DLMSDU to the DLE in a DLM-UNITDATA request with:

Called-address = all 0s

Calling-address = a local identifier for the the DLE's own Individual SA

DLMSDU-queue-identifier pointing to the just formed DLMSDU

The DLE forms a Node DLPDU for each newly passed DLM-UNITDATA request. The contents of this DLM-connectionless DLPDU, described in 26.5.2, are:

DA = all 0s.

SA = the DLE's own Individual SA

DLPDU user-data field = Node DLMSDU

The DLE transmits the just formed Node DLPDU on each startup-channels which shares the low-order 4 bits of the UUID with the just-released startup-channel, as described in 26.5.2.1.1

ENDIF

/* The Conductor waits for a Node-check DLPDU */

IF the Conductor DLE receives a DLM-connectionless DLPDU, as described in 26.5.2, on one of the startup-channel used to transmit the just sent Node DLMSDU

THEN

the DLE sets the local Continue flag = FALSE

the DLE attempts to receive the DLPDU which is present on the link during this channel-selector slot, as described in 26.5.2.1.2

IF the just received DLM-connectionless DLPDU is properly formed, as described in IEC 61158-2

AND the residual of the FCS polynomial is correct

AND the DLPDU was received within 128 startup-scan-periods following the immediately previous DLPDU transmission

NOTE 3 In a properly formed DLPDU, the DLPDU user-data field length, DA length and SA length are each an integral number of octets.

THEN

the DLE extracts the DLPDU user-data field from the just received DLPDU

the DLE forms a DLM-UNITDATA indication, as described in 26.5.2.1.2.

the DLE sets the local Continue flag = TRUE

ELSE

Discard the just received DLPDU

ENDIF

ENDIF

IF the local Continue flag = TRUE

THEN

The DLE passes the DLMSDU to the Bus-configuration DLMS-user via the DLMSDU-queue-identifier of a DLM-UNITDATA indication, as described in 26.5.2.1.2

IF the Bus-configuration DLMS-user determines that this is a Node-check DLMSDU

THEN

the Conductor's Bus-configuration DLMS-user copies the just received Node-check DLMSDU to the Configuration-results-list

the Conductor's Bus-configuration DLMS-user sets its local No-comm-node flag = TRUE

IF UUID from the Node-check DLMSDU = UUID from the just transmitted Node DLMSDU

AND Net-ID from the Node-check DLMSDU = Net-ID from the just transmitted Node DLMSDU

AND Node-VID from the Node-check DLMSDU = Node-VID from the just transmitted Node DLMSDU

AND NCAP-revision-number from the Node-check DLMSDU = NCAP-revision-number from the just transmitted Node DLMSDU

AND Protocol-revision-number from the Node-check DLMSDU = Protocol-revision-number from the just transmitted Node DLMSDU

AND the configuration-status flag from the just received Node-check DLMSDU = FALSE

THEN

the Conductor's Bus-configuration DLMS-user sets its local No-comm-node flag = FALSE

the Conductor's Bus-configuration DLMS-user sets its Node-configuration variable for this startup-channel = ASSIGNED

the Conductor's Bus-configuration DLMS-user sets its Node-address variable for this startup-channel = Net_Id.Node_VID from the just received Node-check DLMSDU

the Conductor's Bus-configuration DLMS-user begins the bus-configuration process for this node, as described in 26.8.

/* the Conductor waits for completion of node configuration */

IF the Conductor completes configuration of a node, as described in 26.8.1.1

THEN

The Conductor's Bus-configuration DLMS-user forms a Drop-channel DLMSDU with:

Called-address =

Net-ID (of the just configured DLE) . Node-VID(of the just configured DLE) . 0 . 0

Calling-address = the individual address of the conductor's DLE

DLMSDU = 1 octet – INT8 = 255

The Conductor's Bus-configuration DLMS-user passes this DLMSDU to the conductor's DLE in a DL-DATA indication

The Conductor's DLE forms a Drop-channel DLPDU

The Conductor's Bus-configuration DLMS-user transmits the Drop-channel DLPDU on the startup-channel that was used to configure the just configured node

The Conductor's Bus-configuration DLMS-user and DLE repeat the above process, starting from formation of the Drop-channel DLMSDU

NOTE 4 The above process de-assigns the startup-channel previously in use from the just configured node

ENDIF

IF the Node-list shows that any unconfigured node should be using the offered startup-channel

THEN

the Conductor attempts to configure another node, as described above, starting with the first step of 26.5.2.3.1

ENDIF

ELSE

The Bus-configuration DLMS-user sets the local No-comm-node = TRUE for the node to which it attempted to assign a startup-channel

IF the Node-list shows that any unconfigured node should be using the offered startup-channel

THEN

the Conductor's Bus-configuration DLMS-user enters the UUID of the unconfigured node in the Configuration-results-list

the Conductor's Bus-configuration DLMS-user marks the above node as a "Missing-node" in the Configuration-results-list

ENDIF

ENDIF

ELSE

Discard the DLMSDU

ENDIF

ELSE

the Conductor attempts to configure another node, as described above, starting with the first step of 26.5.2.3.1

ENDIF

26.5.2.3.1.1 Formation and usage and encoding of the node DLMSDU

The Node DLMSDU, as shown in Figure 129, conveys UUID, Net-ID, Node-VID, NCAP-revision-number and Protocol-revision-number for one non-conductor node.

The Bus-configuration DLMS-user assigns Net-IDs and Node-VIDs so that each DLE's Link-address.DLE-address is unique on the extended link

Item	Length (octets)	Encoding	Allowed values
DLMSDU-type	1	USINT8	1 = Node DLMSDU
UUID	12	OCTETSTRING	any
Net-ID	3	OCTETSTRING	see Table 213
Node-VID	1	USINT8	2 to 255
NCAP-revision-number	1	USINT8	any
Protocol-revision-number	1	USINT8	any
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

Figure 129 – Node DLMSDU composition and encoding

DLMSDU-type

Node DLMSDU = 1

UUID

the Universal Unit Identifier of a NCAP/node. This 12 OCTETSTRING is unique on a worldwide basis. Its low order bits vary between all NCAPs manufactured at a given location.

Net-ID

hierarchical identification of the subnetwork on which this DLE resides in the real network topology. It identifies the local link as seen from the TOP level (L0) of the extended link looking down.

Node-VID

the individual DL-address of the DLE as seen from its local link

NCAP-revision-number

the revision-number of the NCAP associated with a node

Protocol-revision-number

the revision-number of the protocol used by a node

26.5.2.3.2 At each non-conductor

Each non-Conductor performs the following logic on the startup-channels whose Offset-slot equals the value of the lower four bits of its UUID.

IF the non-Conductor enters the START state

THEN

 The Node-configuration DLMS-user sets its Node-configuration variable = EMPTY

 The Node-configuration DLMS-user sets its Node-address variable = NULL

ENDIF

/* During the START and CHECK states each non-Conductor DLE initially listens for a DLPDU */

IF a DLM-connectionless DLPDU is received when the Bus-cycle-counter is incremented and then matches one of the channel-selectors, whose Offset-slot = the lower four bits of this node's UUID

THEN

 the DLE sets the local Continue flag = FALSE

 the DLE attempts to receive the DLPDU which is present on the link during this channel-selector slot

 IF the just received DLM-connectionless DLPDU is properly formed, as described in IEC 61158-2
 AND the residual of the FCS polynomial is correct

NOTE 1 In a properly formed DLPDU, the DLPDU user-data field length, DA length and SA length are each an integral number of octets.

THEN

 the DLE extracts the DA, SA and DLPDU user-data field from the just received DLPDU

 the DLE forms a DLM-UNITDATA indication and passes it to the Node-configuration DLMS-user, as described in 26.5.2.1.2.

 the DLE sets the local Continue flag = TRUE

ELSE

 Discard the just received DLPDU

ENDIF

ENDIF

IF the local Continue flag = TRUE

THEN

IF the Node-configuration DLMS-user determines that this is a Node DLMSDU
AND its local Node-configuration variable = EMPTY

THEN

The Node-configuration DLMS-user extracts the UUID from the associated Node DLMSDU.

IF UUID from the Node DLMSDU matches the non-Conductor node's local UUID

NOTE 2 Since UUIDs are unique on a worldwide basis only one node on the extended link will meet the above criteria

THEN

The Node-configuration DLMS-user sets its Individual DLE-address = Node-VID from the Node DLMSDU

The Node-configuration DLMS-user sets its Individual Link-address = Net-ID from the Node DLMSDU

NOTE 3 The above actions temporarily assign the related startup channel to the non-Conductor DLE whose UUID matches the UUID in the Node DLMSDU

The Node-configuration DLMS-user assigns the startup-channel on which the DLPDU was received to the DLE

NOTE 4 Individual or group addressing may now be used on this startup channel.

The Node-configuration DLMS-user assigns the startup-channel to the DLC-identifier of the upcoming DLM-UNITDATA request

The Node-configuration DLMS-user marks its Node-configuration variable = ASSIGNED

The Node-configuration DLMS-user extracts all remaining parameters from the Node DLMSDU

IF UUID from the just transmitted Node DLMSDU = local UUID

AND NCAP-revision-number from the just transmitted Node DLMSDU = local NCAP-revision-number

AND Protocol-revision-number from the just transmitted Node DLMSDU = local Protocol-revision-number

NOTE 5 The above local parameters are expected to be held in non-volatile memory

THEN

The Node-configuration DLMS-user sets its local configuration-status flag = FALSE
AND copies this value to the upcoming Node-check DLMSDU

The Node-configuration DLMS-user sets its local other-status and low-order-failed-octet-number in the upcoming Node-check DLMSDU as described in 26.5.2.3.2.1

ELSE

The Node-configuration DLMS-user sets its local configuration-status flag = TRUE
AND copies this value to the upcoming Node-check DLMSDU

The Node-configuration DLMS-user sets its local other-status and low-order-failed-octet-number in the upcoming Node-check DLMSDU as described in 26.5.2.3.2.1

ENDIF

The Node-configuration DLMS-user forms a Node-check DLMSDU as described in 26.5.2.3.2.1

The Node-configuration DLMS-user forms DLM-UNITDATA request for the Node-check DLMSDU
The parameters of this DLM-UNITDATA request are:

Called-address = the SA from the just received DL Node DLPDU

Calling-address = the newly assigned Link-address . DLE-address . 0 . 0.

DLPDU user-data field = DLMSDU-queue-identifier of the Node-check DLMSDU

The Node-configuration DLMS-user passes the DLM-UNITDATA request to the DLE

The DLE forms a DLM-connectionless Node-check DLPDU for the Node-check DLMSDU, as described in 26.5.2. Its contents are:

DA = the SA from the just received DL Node DLPDU

SA = the newly assigned Link-address . DLE-address . 0 . 0.

DLPDU user-data field = the newly formed Node-check DLMSDU

The DLE transmits the Node-check DLPDU on its assigned startup channel, as described in 26.5.2.1.1.

The DLE waits for additional DLPDUs on the assigned startup-channel

NOTE 6 These DLPDUs will be associated with additional Bus-configuration DLMSDUs that will be handled by the Node-configuration DLMS-user.

IF the Node-configuration DLMS-user is passed a Drop-channel DLMSDU, identified by:

Called-address = Net-ID (of the node's DLE) . Node-VID (of the node's DLE) . 0 . 0

DLMSDU = 1 octet – INT8 = 255

THEN

The node's Node-configuration DLMS-user de-assigns the just used startup-channel

ENDIF

ELSE

Repeat the procedure of 26.5.2.3.2 starting at its first step.

NOTE 7 This will allow the node to claim the startup-channel at a later time

ENDIF

ELSE

the DLE discards the DLMSDU

ENDIF

ELSE

the DLE continues to wait for a DLM-connectionless DLPDU on any of the startup-channels which are implicitly offered to it by the Conductor

ENDIF

26.5.2.3.2.1 Formation and usage and encoding of the node-check DLMSDU

The Node-check DLMSDU, shown in Figure 130 conveys UUID, Net-ID, Node-VID, NCAP-revision-number, Protocol-revision-number and Configuration-status for one non-conductor DLE.

Item	Length (octets)	Encoding	Allowed values
DLMSDU-type	1	USINT8	2 = Node-check DLMSDU
UUID	12	OCTETSTRING	any
Net-ID	3	OCTETSTRING	see Table 213
Node-VID	1	USINT8	2-255
NCAP-revision-number	1	USINT8	any
Protocol-revision-number	1	USINT8	any
Configuration-status-field	1	OCTETSTRING	see below
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

Figure 130 – Node-check DLMSDU composition and encoding

DLMSDU-type

Node-check DLMSDU = 2

UUID

the Universal Unit IDentifier of a NCAP/node. This 12 OCTETSTRING is unique on a worldwide basis. Its low order bits vary between all NCAPs manufactured at a given location.

Net-ID

hierarchical identification of the subnetwork on which this DLE resides in the real network topology. It identifies the local link as seen from the TOP level (L0) of the extended link looking DOWN.

Node-VID

the individual DL-address of the DLE as seen from its local link

NCAP-revision-number

the revision-number of the NCAP associated with a node

Protocol-revision-number

the revision-number of the protocol used by a node

Configuration-status field is shown in Figure 131. It conveys the configuration-status flag, other-status and low-order-failed-octet-number.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Configuration-status flag		Other-status		Low-order-failed-octet-number			
Success = FALSE Failure = TRUE		defined by conformance-class		0 to 20			

Figure 131 – Configuration-status parameter composition

configuration-status flag

FALSE – the requested configuration was completed successfully

TRUE – the requested configuration was not completed successfully

other-status

use or non-use of other-status, and its encoding, is defined by conformance-class. The specific choice of values does not affect interoperability.

low-order-failed-octet-number

Its format = USINT5

Its range is 0 to 20

0 – no configuration failure occurred

1 – the just received DLMSDU type does not apply to this node

2 to 20 – the low-order octet number of the first item that could not be configured properly

use or non-use of low-order-failed-octet-number is a conformance-class issue

26.5.3 DLM-connectionless DL-addresses

Addressing in DLM-connectionless mode, is shown in Figure 132.

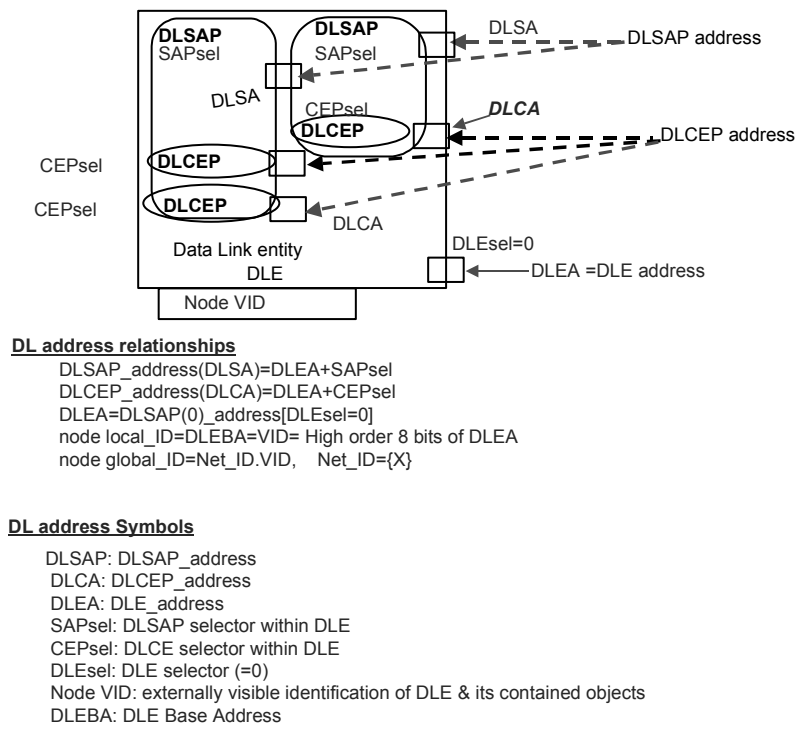


Figure 132 – DLM-connectionless DL-addresses, symbols and their relationships

26.5.3.1 Hierarchical address regimes

The four main DLM-connectionless address regimes are:

- a) DLE-addresses – (Node-VIDs)
- b) DLSAP-selector
- c) DLCEP-selector
- d) Link-addresses – (Link topology identifiers, Net-ID)

26.5.3.2 General form of DLM-connectionless DL-addresses

DLM-connectionless DL-addresses are of the form and encoding shown in Figure 133.

The General form of DLM-connectionless DL-addresses is:

Link-address . DLE-address . DLSAP-selector . DLCEP-selector (64)

! first bit transmitted			last bit ! transmitted
Link-address = Net-ID	DLE address = Node-VID	DLSAP-selector	DLCEP-selector
1 to 3 octets	1 octet	1 octet	1 octet

Figure 133 – General form and encoding of DLM-connectionless DL-addresses

26.5.3.3 DL-addresses - individual, predefined and group addresses

The individual, group and predefined DL-addresses supported are:

- a) the selector of all object in the referenced address regime
- b) the selector(s) of a user defined group or functional group of objects in the referenced address regime
- c) the selector of one (individual) object in the referenced address regime
- d) the selector for all links in the extended link

NOTE The groupings supported are appropriate to the Configuration process.

26.5.3.3.1 DLE-addresses and node VIDs

The format of all DLE-addresses is USINT8

26.5.3.3.1.1 Individual object selectors

The range of Individual DLE-addresses per link is 2 to 255.

The value of an individual DLE's Node-VID = the value of its DLE-address, as shown in Figure 132.

26.5.3.3.1.2 All object selector

The all object selector for all DLEs on the local link is 0.

26.5.3.3.1.3 Functional selectors

The functional object selectors for groups of DLEs on the local link are 2 to 7, inclusive. Predefined functional addresses are shown in Table 211.

Table 211 – Predefined functional and group addresses

Value	Functional group
2	the Active Conductor
3	all Hosts
4	all non-Conductors
5	all Devices
6, 7	reserved

26.5.3.3.1.4 User defined group selectors

The User defined group object selectors for groups of DLEs on the local link are 8 to max-group-DLE, inclusive.

The value of max-group-DLE is a conformance-class issue. Its range is 8 to 15.

26.5.3.3.2 DLSAP-selectors

The format of all DLSAP-selectors is USINT8

26.5.3.3.2.1 Individual object selectors

The range of Individual DLE DLSAP-selectors per DLE is 2 to 255, inclusive. The maximum number and semantics of these individual addresses is a conformance-class issue.

26.5.3.3.2.2 All object selector

The all object selector for all DLSAPs in the (local) DLE is 0.

26.5.3.3.3 DLCEP-selectors

The format of all DLCEP-selectors is USINT8

Predefined ranges of DLCEP-selectors apply to channels and triggers.

26.5.3.3.3.1 Individual channel or half-forwarding-port selectors

The range of Individual DLCEP-selectors per DLSAP that is assigned to channels or half-forwarding-ports is 8 to 255. The maximum number and semantics of these individual selectors is a conformance-class issue.

26.5.3.3.3.2 Individual trigger selectors

The range of Individual DLCEP-selectors per DLSAP that is assigned to triggers is 2 to 7. The maximum number and semantics of these trigger selectors is a conformance-class issue. Table 212 shows Predefined Trigger assignments.

Table 212 – Predefined trigger assignments

Value	Trigger assignments
2	the Global Trigger
3 to 7	Individual Triggers

26.5.3.3.3.3 All object selector

The all object selector for all DLCEPs in one (local) DLSAP is 0.

26.5.3.3.4 Link address

The length of the Link-address depends on the depth of the topology of links in the extended link, as shown in Table 213. This table shows:

- the length and specific form of Link-address vs. depth of the real Nest of links in the extended -link.
- the range of allowed values of Net-ID, the component of Link-address that identifies a specific, individual link within the extended -link
- the range of allowed values of the selector for individual links at this level, and
- the selector for all links on the extended link.

Table 213 – Format of link-addresses

Depth of real nest of links	Length of link-address octet string	Specific form of concatenated link-address	Real topology of extended-link	Range of allowed values of Net-ID	Selector for all links on the extended link	Selector for individual links at this level
0	1	[z = 0]	local link only	0, 1	0	1
1	1	[z =2 to 255]		2 to 255	0	2 to 255
2	2	[y =2 to 255] [z =2 to 255]	example shown in Figure 112	2 to 255	0	2 to 255
3	3	[x =2 to 255] [y =2 to 255] [z =2 to 255]	maximum depth of nest of real links	2 to 255	0	2 to 255

The actual depth of the real nest of links on the extended link is 0 to max-Net-level.

The value of max-Net-level is a conformance-class issue. Its range is 0 to 3.

EXAMPLE Figure 112 shows one example real network topology. In this figure, Net-ID T = [0], Net-ID A may be = [2 to 255], Net-ID B may be = [2 to 255], Net-ID R may be = [2 to 255][2 to 255], Net-ID S may be = [2 to 255][2 to 255]. One set of allowed assignments is T=[0], A = [2], R = [2][2], S = [2][3], B = [3].

26.5.3.3.4.1 Link-address selectors

The format of all Link-address-selectors is an OCTETSTRING of the length shown in Table 213, column 2

26.5.3.3.4.2 Individual link selectors

The range of Individual DLE Link-address-selectors per level of the extended-link is 2 to 255, except when referencing the local-link. The maximum number and semantics of these individual selectors is a conformance-class issue.

The selector for the local link = 1

26.5.3.3.4.3 All object selector

The all object selector for all links on the extended-link is all 0s.

26.6 Other real-time services

The following real-time services, which are above those related to data transmission and reception, are offered.

26.6.1 Common sense of bus time

The Conductor enforces a common sense of Bus-time (local DL-Time-offset) on all nodes on a local link or a lower link in an extended link.

The local parameters and their formats and allowed range of values that support the DL-TIME-OFFSET primitive are shown in Table 214.

Table 214 – DL-TIME-OFFSET parameter composition

DL-TIME-OFFSET Parameter	Components	Format	Length of OCTET STRING	Range of values
high-order bit of local-DL-Time	NcycleLO	USINT1		0 to 1
local-DL-Time-mid-order	Bus-cycle-counter	OCTET STRING	7	0 to 2 ^{LBL} -1
local-DL-Time-low-order	Slot-clock/2	USINT8		0 to 129

The procedure to calculate the Local DL-Time-offset (of an Event) from the local parameters of this table is :

$$\begin{aligned} \text{local DL-Time-offset} = & \\ & (\text{ABS}(\text{low order bit of conductor's Ncycle-counter} - \text{NcycleLO}) \\ & + (\text{Bus-Cycle-counter} / \text{Bus-cycle-length}) \times \text{Bus-cycle-time}) \\ & + (\text{Slot-clock} / \text{Slot-length} \times \text{Slot\_time}) \quad (\text{in } \mu\text{s}) \end{aligned} \quad (65)$$

The procedure to calculate the time of an event from Local-DL-Time-offset:

$$\begin{aligned} \text{Event-time} = & \text{System-time at the Conductor's latest bus cycle rollover} \\ & + \text{local-DL-Time-offset} \quad (\text{in } \mu\text{s}) \end{aligned} \quad (66)$$

NOTE The source and external synchronization of System-time is a user choice. It might be UTC, or the current time in the local time zone.

The primitives and parameters to support the Time offset function are shown in Table 215.

Table 215 – Time-Offset primitive and parameters

DL-TIME-OFFSET Parameter name	Request output
local-offset	M

26.6.1.1 DL-time classes

The values of the DL-Time-Class attribute, the aliasing of common time sense across various DLEs, relates primarily to Bit-rate and the number of Bridges traversed in an extended link, is shown in Table 216.

Table 216 – DL-time classes

Data rate	Number of bridges traversed	Possible DL-Time-Class
≥ 1.25 Mbit/s	0	1 μs
≥ 5 Mbit/s	3	1 μs
≥ 0.625 Mbit/s	3	10 μs
≥ 0.0785 Mbit/s	3	100 μs

The actual DL-Time-class achieved across all nodes on a local or extended link is a Bus-configuration issue.

26.6.1.2 Transmission delay

Transmission delay on the local link can be less than or equal to two bit-times, the resolution at which a Trigger can be synchronized to the Scan Period of a channel.

Transmission delay on an extended link depends on the allocation of slots on the upper levels of the extended link and the ratio of Data-rates from the TOP to the local levels of the real or virtual network used by the associated channel or trigger.

The minimum delay is one lower Slot-time per bridge traversed.

26.6.2 Time stamping of events

A node may support time-stamping of Events. Its DLE inserts the local DL-Time-offset and NcycleLO, when notified by the DLS-user, into a time-stamp-buffer selected by Time-stamp-ID,. This buffer may be appended to the associated Event message by the DLS-user before that message is submitted to the DLE for transmission.

NOTE 1 Time-stamping eliminates the extra latency that DLS-user activity would introduce in recording time-stamps. Thus the time stamp appended to the message may be approximately concurrent with the time the relevant Event occurred, even if the Event message is transmitted much later.

NOTE 2 The DLS-user may notify the DLE to record Local-DL-Time and NcycleLO via a zero latency mechanism.

Each DLSAP may have 0 to max-time-stamp non-retentive time-stamp-buffers. Max-time-stamp is a conformance-class issue and may range from 0 to 1 inclusive.

Inter-DLE errors in the time-stamp are a function of inter-DLE errors in Bus-time.

The primitives and parameters to support the Time-Stamp function are shown in Table 217.

Table 217 – Time stamp primitives and parameters

DL-TIME-STAMP	Request	Indication
Parameter Name	input	output
Time-Stamp-ID	M	
Local-DL-Time-offset		M

The local parameters and their formats and allowed range of values to support the Time-Stamp function are shown in Table 218.

Table 218 – Time stamp parameter composition

DL-TIME Parameter	Components	Format	Length of OCTET STRING	Range of values
Time-Stamp-ID		USINT1		0 to 1
high-order bit of local-DL-Time	NcycleLO	USINT1		0 or 1
local-DL-Time-mid-order	Bus-cycle-counter	OCTET STRING	7	0 to 2 ^{LBL} - 1
local-DL-Time-low-order	Slot-clock/2	USINT8		0 to 129

The procedure to calculate the Conductor's-System-time of an Event from these parameters is:

IF NcycleLO of the Conductor = NcycleLO of the node that reports the local-DL-Time-offset (time stamp) of an Event as described in 26.6.2

THEN

Conductor's-System-time-of-the-Event=
the Conductor's-System-time at the last rollover of the Bus-cycle-counter
+ the local DL-Time-offset of the Time-stamp (in μs) (67)

ELSE

Conductor's-System-time-of-the-Event=
the Conductor's-System-time at the last rollover of the Bus-cycle-counter
+ the local DL-Time-offset, of the Time-stamp
+ (Bus-cycle-length * Slot-time) (in μs) (68)

ENDIF

26.6.3 Event time

The primitives and parameters to support the Event-time function are shown in Table 219.

Table 219 – DL-Event-time primitive and parameters

DL-Event-TIME	Request
Parameter Name	output
Conductor's System-time-of-the-Event	M

26.6.4 Triggers

Each DLE can provide synchronized Trigger signals, which may trigger a variety of node internal or external actions.

DLEs derive trigger signals from their own Bus-Cycle counter and Slot-clock.

NOTE Since triggers do not consume a slot on the bus, any number of actions may be triggered simultaneously without slowing down data transfers. Also, provision of these trigger signals does not impact data transfers.

The maximum number of Trigger signals per DLSAP is a conformance-class issue. Its range is 0 to max-trigger. The range of max-trigger is 0 to 7, inclusive.

There are no primitive exchanges between the DLS-provider and DLS-user related to Triggers, although there may be hardware interactions between these entities.

IF Bus-state = NORMAL
AND No-comm-node for the DLE = FALSE
AND No-comm-trigger for the DLCEP associated with DLC-identifier = FALSE
AND the Bus-cycle-counter is incremented and then matches one of the trigger-selectors, described in below,
for one of the DLSAP's configured triggers
AND Slot-time = the configured value for the above trigger

THEN

Assert a trigger signal, as described below, on the "I/O channel" specified by the trigger-number associated with the trigger DLCEP which activated the trigger.

ENDIF

Trigger-selectors = Trigger-offset + [1, 2, ... n]* Scan-period (69)

26.6.4.1 Trigger "channel"-activation

When a DLE senses that a Trigger has been activated, it triggers the associated signal,

As specified in IEEE 1451.2, a Trigger signal can be applied directly to a device port by the DLE, or by the associated NCAP to either:

- all the "I/O channels" in the device (TEDS value =0); – the Global Trigger or
- a specific "I/O channel" defined by the device TEDS – an individual Trigger.

NOTE 1 The above definition of "channel" is taken from IEEE 1451.2. It differs from the definition used in the balance of this clause.

NOTE 2 Routing of trigger signals is not a DLE responsibility.

26.7 Bridge architecture and operation

Figure 134 shows the architecture of a forwarding bridge, including one or more sets of an upper-half-forwarding-port, a lower-half-forwarding-port and a shared forwarding-buffer.

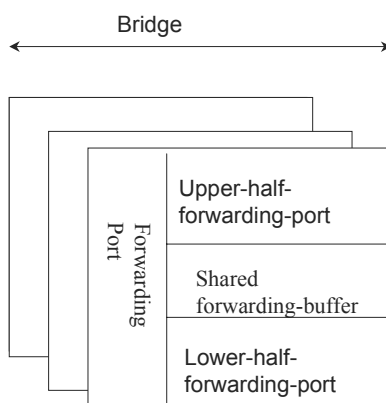


Figure 134 – Architecture of a forwarding bridge

26.7.1 Forwarding by slot matching

EXAMPLE Figure 135 shows forwarding through a forwarding-bridge by Slot matching. This figure represents the operation of the extended link shown in Figure 112. In this figure all of the slots shown appear on the TOP link, with Bit-rate = 4X; 75 % of the slots shown appear on lower link A, with Bit-rate = 3X; 25 % of the slots shown appear on lower link B, with Bit-rate = X and 25 % of the slots shown are originated by non-Conductors on the TOP link.

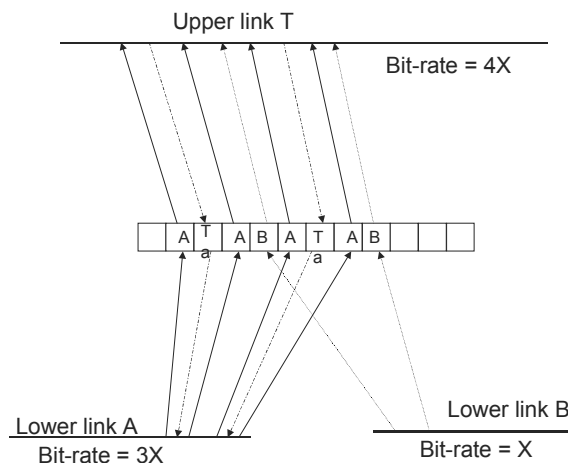


Figure 135 – EXAMPLE: Forwarding through a bridge by slot matching

26.7.2 Forwarding-port direction

Forwarding-ports may support 1WAY data transmission, as described in 26.4.3.2.1, or 2WAY data transmission, as described in 26.4.3.2.2.

The types of channels supported by each of these types of transmission are:

1WAY – SCAN and ExSCAN

2WAY – SCAN, ExSCAN, GPA and GPC

The maximum number of forwarding-ports per Bridge and their directionality is a conformance-class issue.

26.7.3 Forwarding-port data packaging

A half-forwarding-port treats each DLPDU as an independent entity. Thus its data packaging is modeled on that for a UNITARY channel, as described in 26.4.3.1.1.1.

26.7.4 Reception and transmission via a forwarding-port

A forwarding-port operates by:

- a) Receiving a DLPDU via one half forwarding-port
- b) Storing the just received DLPDU in a shared forwarding-buffer
- c) Later retrieving the just stored DLPDU from the shared forwarding-buffer
- d) Transmitting the just retrieved DLPDU on the opposite half forwarding-port

26.7.4.1 At the receiving half forwarding-port

The receiving half-forwarding-port performs the following logic:

```

IF Bus-state = NORMAL
  AND No-comm-node for the DLE = FALSE
  AND No-comm-forwarding-port = FALSE
  AND the upcoming DLPDU is inbound
  AND the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in
  26.4.3.1.1.1.1.1, for the channel associated with this half-forwarding-port

```

THEN

Attempt to receive the DLPDU which is present on the link during this half-forwarding-port-selector slot

```

IF the just received DLPDU is properly formed, as described in IEC 61158-2
  AND the residual of the FCS polynomial is correct

```

NOTE 1 In a properly formed DLPDU, the DLPDU user-data field length is an integral number of octets.

THEN

Extract the user-data field from the just received DLPDU

Save the user-data field contents in the shared-forwarding-buffer associated with this forwarding-port.

NOTE 2 The low-order bit of the DLPDU user-data field = the low-order bit of the shared forwarding buffer

ELSE

Discard the just received DLPDU

ENDIF

ENDIF

26.7.4.2 At the transmitting half forwarding-port

The transmitting half-forwarding-port performs the following logic:

```

IF Bus-state = NORMAL
  AND No-comm-node for the DLE = FALSE
  AND No-comm-forwarding-port = FALSE
  AND the saved DLPDU is outbound
  AND the Bus-cycle-counter is incremented and then matches one of the channel-selectors, described in
  26.4.3.1.1.1.1.1, for the channel associated with this forwarding-port
THEN
  Form a single DLPDU with its user-data field equal to the entire contents of the shared-forwarding-buffer
  NOTE The low-order bit of the DLPDU user-data field equals the low-order bit of the shared-forwarding-buffer
  Transmit the just formed DLPDU
ENDIF

```

26.8 Bus configuration

Following temporary assignment of a startup-channel to a specific non-conductor node, as described in 26.5.2.3, the Conductor configures each of that node's configuration-objects

NOTE 1 The configuration-objects defined are channels, triggers and half-forwarding-ports.

The configuration proceeds by an exchange of Bus-configuration DLMSDUs:

- a) a Channel DLMSDU and Channel-check DLMSDU for each channel listed in the Node-list for the above node
- b) a Trigger DLMSDU and Trigger-check DLMSDU for each trigger listed in the Node-list for the above node
- c) a Half-forwarding-port DLMSDU and Half-forwarding-port-check DLMSDU for each Forwarding-port listed in the Node-list for the above node

NOTE 2 Forwarding-ports are listed only once in the Node-list. This entry is related to the Lower-Port Bridge DLE of the associated Bridge.

NOTE 3 The configuration process for the Conductor is not specified by this standard.

26.8.1 Configuration process

For each instance of one the categories of configuration-objects contained in the Node-list for the node now being configured, the process below is followed.

26.8.1.1 At the conductor

The Conductor performs the following logic on the active startup channel during the START state.

```

IF the Conductor has just begun configuration of a newly assigned node
  OR the Conductor has just completed configuration of one configuration-object
THEN
  The Bus-configuration DLMS-user forms a Configuration-object DLMSDU, as described in 26.8.2, for the
  next configuration-object in the Node-list for the associated node.
  The Bus-configuration DLMS-user passes the just formed Configuration-object DLMSDU to the DLE in a
  DLM-UNITDATA request with:
    Called-address = the individual address of the DLCEP to be configured
    Calling-address = a local identifier for the DLE's own Individual SA
    DLMSDU-queue-identifier pointing to the just formed DLMSDU

```

The DLE forms a configuration-object DLPDU for each newly passed DLM-UNITDATA request. The contents of this DLM-connectionless DLPDU, described in 26.5.2, are:

DA = the individual address of the DLCEP to be configured.

SA = the Conductor DLE's own Individual SA

DLPDU user-data field = Configuration-object DLMSDU

The DLE transmits the just formed configuration-object DLPDU on the startup-channel assigned to the node being configured, as described in 26.5.2.1.1

ENDIF

/* The Conductor waits for a configuration-object-check DLPDU */

IF the Conductor DLE receives a DLM-connectionless DLPDU, as described in 26.5.2, on the startup-channel used to transmit the just sent Configuration-object DLMSDU

THEN

the DLE sets the local Continue flag = FALSE

the DLE attempts to receive the DLPDU which is present on the link during this channel-selector slot, as described in 26.5.2.1.2

IF the just received DLM-connectionless DLPDU is properly formed, as described in IEC 61158-2

AND the residual of the FCS polynomial is correct

AND the DLPDU was received within eight startup-scan-periods following the immediately previous DLPDU transmission

NOTE In a properly formed DLPDU, the DLPDU user-data field length, DA length and SA length are each an integral number of octets.

THEN

the DLE extracts the DA, SA and DLPDU user-data field from the just received DLPDU

the DLE forms a DLM-UNITDATA indication, as described in 26.5.2.1.2.

the DLE sets the local Continue flag = TRUE

ELSE

Discard the just received DLPDU

ENDIF

ENDIF

IF the local Continue flag = TRUE

THEN

The DLE passes the DLMSDU to the Bus-configuration DLMS-user via the DLMSDU-queue-identifier of a DLM-UNITDATA indication, as described in 26.5.2.1.2

the Conductor's Bus-configuration DLMS-user copies the just received Configuration-object-check DLMSDU to the Configuration-results-list

IF the Bus-configuration DLMS-user determines that this is a correct type of Configuration-object-check DLMSDU

AND each parameter of the just received Configuration-object-check DLMSDU = the corresponding parameter from the just transmitted Configuration-object DLMSDU

AND the configuration-status-flag = FALSE in the just received Configuration-object-check DLMSDU

THEN

the Conductor's Bus-configuration DLMS-user sets the local No-comm-configuration-object flag of this configuration-object = FALSE

ELSE

the Conductor's Bus-configuration DLMS-user sets the local No-comm-configuration-object flag of this configuration-object = TRUE

ENDIF

IF the Node-list shows that any unconfigured configuration-object for this node

THEN

The Bus-configuration DLMS-user repeats this subclause starting from the first step

ELSE

/* Configuration of this node is complete */

The Conductor's Bus-configuration DLMS-user begins configuration of the next node on the Node-list that uses the same low-order 4 bits of the UUID as the just configured node, by repeating 26.8.1.1 starting at the first step.

ENDIF

ENDIF

26.8.1.2 At the non-conductor

IF a DLM-connectionless DLPDU is received on the startup-channel assigned to this node

THEN

the DLE sets the local Continue flag = FALSE

the DLE attempts to receive the DLPDU which is present on the link during this channel-selector slot

IF the just received DLM-connectionless DLPDU is properly formed, as described in IEC 61158-2
AND the residual of the FCS polynomial is correct

NOTE 1 In a properly formed DLPDU, the DLPDU user-data field length, DA length and SA length are each an integral number of octets.

THEN

the DLE extracts the DA, SA and DLPDU user-data field from the just received DLPDU

the DLE forms a DLM-UNITDATA indication, as described in 26.5.1.1, and passes it to the Node-configuration DLMS-user. The parameters of this DLM-UNITDATA indication are:

Calling-address = the SA used by the Conductor in the just received Node DLPDU

Called-address = the temporarily assigned

address-to-be-configured . DLE-address-to-be-configured

Link-

DLSAP-to-be-configured . DLCEP-to-be-configured.

DLPDU user-data field = DLMSDU-queue-identifier of the Node DLMSDU

the DLE sets the local Continue flag = TRUE

ELSE

Discard the just received DLPDU

ENDIF

ENDIF

IF the local Continue flag = TRUE

AND the Node-configuration DLMS-user determines that this is a Configuration-object DLMSDU

THEN

The Node-configuration DLMS-user extracts each object in-turn from the Configuration-object DLMSDU

The Node-configuration DLMS-user attempts to configure the non-Conductor DLCEP, specified in the Called-address of the DLM-UNITDATA indication, as specified by the parameter just extracted from the Configuration-object DLMSDU

The Node-configuration DLMS-user forms a Configuration-object-check DLMSDU, as described in 26.8.3, which gives the result of the above configuration process.

The Node-configuration DLMS-user forms DLM-UNITDATA request for the Node-check DLMSDU. The parameters of this DLM-UNITDATA request are:

Called-address = the SA from the just received DL Node DLPDU

Calling-address = the individual address of the DLCEP which was just configured.

DLPDU user-data field = DLMSDU-queue-identifier of the Node-check DLMSDU

The Node-configuration DLMS-user passes the DLM-UNITDATA request to the DLE

The DLE forms a DLM-connectionless Node-check DLPDU for the Node-check DLMSDU, as described in 26.5.2. Its contents are:

DA = the SA from the just received DL Node DLPDU

SA = the individual address of the DLCEP which was just configured.

DLPDU user-data field = the newly formed Configuration-object-check DLMSDU

The DLE transmits the Configuration-object-check DLPDU on its assigned startup channel, as described in 26.5.2.1.1.

The DLE waits for additional DLPDUs on the assigned startup-channel

NOTE 2 These DLPDUs will be associated with additional Bus-configuration DLMSDUs, which will be handled by the Node-configuration DLMS-user.

ELSE

Repeat the procedure of 26.8.1.2 starting at its first step.

NOTE 3 This will allow the node to receive Configuration-object DLPDUs at a later time

ENDIF

26.8.2 Formation and usage and encoding of configuration-object DLMSDUs

For references to this subclause, the reader should make a macro substitution of the appropriate one of Channel DLMSDU or Trigger DLMSDU or Half-forwarding-port DLMSDU for the text "configuration-object DLMSDU".

All values in Configuration-object DLMSDUs, except DLMSDU-type, are extracted from the Node-list by the Bus-configuration DLMS-user and copied to the associated parameter of the appropriate Configuration-object DLMSDU.

26.8.2.1 Formation and usage and encoding of channel DLMSDUs

The Channel DLMSDU, as shown in Table 220, conveys Channel-number, Scan-class, Offset-slot, Channel-mode, Channel-type, DLCEP-QoS and max-confirm-delay for one DLCEP in a non-conductor node.

Table 220 – Channel DLMSDU composition and encoding

Item	Length (octets)	Encoding	Allowed Values
DLMSDU-type	1	USINT8	3 = Channel DLMSDU
Channel-number	(LBL+7)/8	OCTETSTRING	see below
Scan-class	1	OCTETSTRING	see below
Offset-slot	(LBL+7)/8	OCTETSTRING	see below
Channel-mode	1	OCTETSTRING	see below
Channel-type	1	ENUMERATED	see below
DLCEP-QoS	1	USINT4	0 to 15
max-confirm-delay	1	USINT5	10 to 31
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

DLMSDU-type

Channel DLMSDU = 3

Channel-number

the visible identification of this channel

Its range is 0 to $2^{LBL} - 1$

Scan-class

the base 2 logarithm of the number of slots between slot-selectors for this channel

Its range is 1 to LBL

This implies that the Scan-period has $2^{\text{Scan-class}}$ slots.

Offset-slot

the first slot-selector following the Bus-cycle-end slot for this channel

Its range is 0 to $[2^{L_{BL}} - 2]$

Channel-mode is shown in Figure 136. It conveys num-time-stamp, DLCEP role and DLCEP-direction

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
num-time-stamp		Reserved				DLCEP role	DLCEP-direction
0 to 1						TRUE = Master FALSE = Slave	TRUE = Publisher FALSE = Subscriber

Figure 136 – Channel-mode composition and encoding

num-time-stamp

the number of time-stamp buffers present in this DLCEP

Its range is 0 to max-time-stamp, whose range is 0 to 1

DLCEP role

specifies the role of this DLCEP on a GPA or GPC channel

TRUE – this DLCEP is the Master

FALSE – this DLCEP is the Slave

DLCEP-direction

specifies the direction of transmission on a SCAN or ExSCAN channel

TRUE – this DLCEP is the Publisher, DLCEP-direction = From-device

FALSE – this DLCEP is a Subscriber, DLCEP-direction = To-device

Channel-type

0 – SCAN channel

1 – ExSCAN channel

2 – GPA channel

3 – GPC channel

DLCEP-QoS

a selector that identifies, for one DLCEP, a specific set of QoS options that is specified in the Conformance-class now in use.

Its range is 0 to 15, inclusive

max-confirm-delay

sets the upper bounds on the maximum time duration permitted for the completion of a sequence of connection-oriented DLS primitives on a GPA or GPC channel. :

Its range is 10 to 31 max-transfer-periods, inclusive.

26.8.2.2 Formation and usage and encoding of trigger DLMSDUs

The Trigger DLMSDU, as shown in Table 221, conveys Trigger-number, Scan-class, Offset-slot, Offset-bit and Trigger-type for one DLCEP in a non-conductor node.

Table 221 – Trigger DLMSDU composition and encoding

Item	Length (octets)	Encoding	Allowed Values
DLMSDU-type	1	USINT8	5 = Trigger DLMSDU
Trigger-number	1	OCTETSTRING	see below
Scan-class	1	OCTETSTRING	see below
Offset-slot	(LBL+7)/8	OCTETSTRING	see below
Offset-bit	2	USINT9	see below
Trigger-type	1	ENUMERATED	see below
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

DLMSDU-type

Trigger DLMSDU = 5

Trigger-number

The local identification of the trigger, with a range of 0 to 255.

Scan-class

See 26.8.2.1, substituting “trigger” for “channel”.

Offset-slot

See 26.8.2.1, substituting “trigger” for “channel”.

Offset-bit

the bit-position within the trigger-selector-slot at which the trigger is issued, measured relative to the rollover of the Slot-clock..

Its range is 0 to $[2^9 - 1]$

NOTE Implementors may elect to round the internal value to the USINT8 format.

Trigger-type

1 – Global trigger

2 – Individual trigger

26.8.2.3 Formation and usage and encoding of half-forwarding-port DLMSDUs

The half-forwarding-port DLMSDU, as shown in Table 222, conveys Upper-half-forwarding-port-number, Scan-class, Offset-slot, Half-forwarding-port-mode, Forwarding-port-type, for one DLCEP in a Bridge node.

Table 222 – Half-forwarding-port DLMSDU composition and encoding

Item	Length (octets)	Encoding	Allowed Values
DLMSDU-type	1	USINT8	7 = Half-forwarding-port DLMSDU
Upper-half-forwarding-port-number	1	OCTETSTRING	see below
Scan-class	1	OCTETSTRING	see below
Offset-slot	(LBL+7)/8	OCTETSTRING	see below
Half-forwarding-port-mode	1	OCTETSTRING	see below
Forwarding-port-type	1	ENUMERATED	see below
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

DLMSDU-type

Half-forwarding-port DLMSDU = 7

Upper-half-forwarding-port-number

the local identification of the Upper-half-forwarding-port

Its range is 0 to 255

Scan-class

See 26.8.2.1, substituting “lower forwarding port” for “channel”

Offset-slot

See 26.8.2.1, substituting “lower forwarding port” for “channel”

Half-forwarding-port-mode is shown in Figure 137. It conveys half-forwarding-port-direction

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
Reserved							half-forwarding-port-direction
							TRUE = Transmitter FALSE = Receiver

Figure 137 – Half-forwarding-port mode composition and encoding

DLCEP-direction

specifies the direction of transmission on a SCAN or ExSCAN channel

TRUE – this DLCEP is the Transmitter, DLCEP-direction = From-device

FALSE – this DLCEP is a Receiver, DLCEP-direction = To-device

Channel-type

0 – 1WAY forwarding-port

2 – 2 WAY forwarding-port

26.8.3 Formation and usage and encoding of configuration-object-check DLMSDUs

For references to this subclause, the reader should make a macro substitution of the appropriate one of Channel-check DLMSDU or Trigger-check DLMSDU or Half-forwarding-port-check DLMSDU for the text "configuration-object-check DLMSDU"

All values in Configuration-object-check DLMSDUs, except DLMSDU-type and configuration-status, are copied from the actual, local configured value by the Node-configuration DLMS-user.

26.8.3.1 Formation and usage and encoding of channel-check DLMSDUs

The Channel-check DLMSDU, as shown in Table 223, conveys Channel-number, Scan-class, Offset-slot, Channel-mode, Channel-type, DLCEP-QoS, max-confirm-delay and configuration-status for one DLCEP in one non-conductor node.

Table 223 – Channel check DLMSDU composition and encoding

Item	Length (octets)	Encoding	Allowed Values
DLMSDU-type	1	USINT8	4 = Channel-check DLMSDU
Channel-number	(LBL+7)/8	OCTETSTRING	see below
Scan-class	1	OCTETSTRING	see below
Offset-slot	(LBL+7)/8	OCTETSTRING	see below
Channel-mode	1	OCTETSTRING	see below
Channel-type	1	ENUMERATED	see below
DLCEP-QoS	1	USINT4	0 to 15
max-confirm-delay	1	USINT5	10 to 31
Configuration-status-field	1	OCTETSTRING	see below
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

DLMSDU-type

Channel-check DLMSDU = 4

Channel-number

See 26.8.2.1.

Scan-class

See 26.8.2.1.

Offset-slot

See 26.8.2.1.

Channel-mode

See 26.8.2.1.

Channel-type

See 26.8.2.1.

DLCEP-QoS

See 26.8.2.1.

max-confirm-delay

See 26.8.2.1.

Configuration-status field is shown in Figure 138. It conveys the configuration-status flag, the low-order octet number of the first item which could not be configured, and other status.

bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
configuration-status flag	other-status		low-order-failed-octet-number				
Success = FALSE Failure = TRUE	defined by conformance- class		0 to 20				

Figure 138 – Configuration-status field composition

configuration-status flag

FALSE – the requested configuration was completed successfully

TRUE – the requested configuration was not completed successfully

other-status

use or non-use of other-status, and its encoding, is defined by conformance-class. The specific choice of values does not affect interoperability.

low-order-failed-octet-number

Its format = USINT5

Its range = 0 to 20

0 – no configuration failure occurred

1 – the just received DLMSDU type does not apply to this node

2 to 20 – the low-order octet number of the first item that could not be configured properly

use or non-use of low-order-failed-octet-number is a conformance-class issue

26.8.3.2 Formation and usage and encoding of trigger-check DLMSDUs

The Trigger DLMSDU, as shown in Table 224, conveys Trigger-number, Scan-class, Offset-slot, Offset-bit, Trigger-type and configuration-status for one DLCEP in a non-conductor node.

Table 224 – Trigger-check DLMSDU composition and encoding

Item	Length (octets)	Encoding	Allowed Values
DLMSDU-type	1	USINT8	6 = Trigger-check DLMSDU
Trigger-number	1	OCTETSTRING	see below
Scan-class	1	OCTETSTRING	see below
Offset-slot	(LBL+7)/8	OCTETSTRING	see below
Offset-bit	2	USINT9	see below
Trigger-type	1	ENUMERATED	see below
Configuration-status-field	1	OCTETSTRING	see below
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

DLMSDU-type

Trigger-check DLMSDU = 6

Trigger-number

See 26.8.2.2.

Scan-class

See 26.8.2.2.

Offset-slot

See 26.8.2.2.

Offset-bit

See 26.8.2.2.

Trigger-type

- 1 – Global trigger
- 2 – Individual trigger

Configuration-status field

See 26.8.3.1.

26.8.3.3 Formation and usage and encoding of half-forwarding-port-check DLMSDUs

The half-forwarding-port-check DLMSDU, as shown in Table 225, conveys Upper-half-forwarding-port-number, Scan-class, Offset-slot, Half-forwarding-port-mode, Forwarding-port-type, and configuration-status for one DLCEP in a Bridge.

Table 225 – Half-forwarding-port DLMSDU composition and encoding

Item	Length (octets)	Encoding	Allowed Values
DLMSDU-type	1	USINT8	8 = Half-forwarding-port-check DLMSDU
Upper-half-forwarding-port-number	1	OCTETSTRING	see below
Scan-class	1	OCTETSTRING	see below
Offset-slot	(LBL+7)/8	OCTETSTRING	see below
Half-forwarding-port-mode	1	OCTETSTRING	see below
Forwarding-port-type	1	ENUMERATED	see below
Configuration-status-field	1	OCTETSTRING	see below
High order bit (Octet n, bit 7) of DLMSDU ! last bit Transmitted			

DLMSDU-type

Half-forwarding-port-number DLMSDU = 8

Upper-half-forwarding-port-number

See 26.8.2.3.

Scan-class

See 26.8.2.3.

Offset-slot

See 26.8.2.3.

Half-forwarding-port-mode

See 26.8.2.3.

DLCEP-direction

specifies the direction of transmission on a 1WAY half-forwarding-port

TRUE – this DLCEP is the Transmitter, DLCEP-direction = From-device

FALSE – this DLCEP is a Receiver, DLCEP-direction = To-device

Forwarding-port-type

See 26.8.2.3, Channel-type.

Configuration-status-field

See 26.8.3.1.

26.9 Conformance

Conformance of a Node to the Type 6 protocol specifications is determined in three steps:

- a) conformance to the general requirements of this Type
- b) conformance to the additional requirements of a specific type of node specified for this Type
- c) conformance to the additional requirements of a specific conformance-class specified for this Type

NOTE Specification of a clause below includes all of its subsidiary subclauses

26.9.1 General requirements of this Type

- a) Connection Mode Data Transfer using Time Division Multiple Access (TDMA) Principles, specified in 26.1
- b) Bus Access and Data Transmission, specified in 26.2
- c) the non-Conductor role in Bus Synchronization on the local link, specified in 26.3.1
- d) the non-Conductor role in Slot-clock Synchronization on the local link, specified in 26.3.4.1
- e) DLM-connectionless data transmission and reception specified in 26.5.2.1
- f) the non-conductor role in Bus Startup and Operation specified in 27.1
- g) the non-conductor role in assignment of startup-channels and Node configuration for DLM-connectionless operation specified in 26.5.2.3.2
- h) Support of DLM-connectionless addresses specified in 26.5.3
- i) Support of the Media Access Control specified in 27.2

26.9.2 Additional requirements of a specific type of node specified for this Type

26.9.2.1 Host nodes

- a) the Conductor role in Bus Synchronization on the local link, specified in 26.3.1
- b) the Conductor role in Slot-clock Synchronization on the local link, specified in 26.3.4.1
- c) the Conductor role in Bus Startup and Operation specified in 27.1
- d) the Conductor role in assignment of startup-channels and Node configuration for DLM-connectionless operation as specified in 26.5.2.3.1
- e) the Conductor role in Configuration of channels, triggers and half-forwarding-ports specified in 26.8

26.9.2.2 Bridge nodes

- a) Bus Synchronization on the extended link, specified in 26.3.2
- b) Slot-clock Synchronization on the extended link, specified in 26.3.4.2
- c) Forwarding by Slot matching specified in 26.7.1
- d) the non-Conductor role in Configuration of half-forwarding-ports specified in 26.8

26.9.2.3 Nodes that support SCAN channels

- a) UNITARY Method of data-packaging specified in 26.4.3.1.1
- b) 1WAY data transmission direction method specified in 26.4.3.2.1
- c) the non-Conductor role in Configuration of channels specified in 26.8

26.9.2.4 Nodes that support ExSCAN channels

- a) GRANULAR Method of data-packaging specified in 26.4.3.1.2
- b) 1WAY data transmission direction method specified in 26.4.3.2.1
- c) the non-Conductor role in Configuration of channels specified in 26.8

26.9.2.5 Nodes that support GPA channels

- a) GRANULAR Method of data-packaging, including QoS negotiation, specified in 26.4.3.1.2
- b) 2WAY data transmission direction method specified in 26.4.3.2.2
- c) Retransmission, specified in 26.4.2.3.3
- d) Common sense of Bus-time specified in 26.6.1
- e) Time-stamping of Events specified in 26.6.2
- f) the non-Conductor role in Configuration of channels specified in 26.8

26.9.2.6 Nodes that support GPC channels

- a) GRANULAR Method of data-packaging, including QoS negotiation, response-requests and Reply DLMSDUs, specified in 26.4.3.1.2
- b) 2WAY data transmission direction method specified in 26.4.3.2.2
- c) Retransmission, specified in 26.4.2.3.3
- d) Common sense of Bus-time specified in 26.6.1
- e) Time-stamping of Events specified in 26.6.2
- f) the non-Conductor role in Configuration of channels specified in 26.8

26.9.2.7 Nodes that support triggers

- a) Triggers, either Global or I/O triggers or both, specified in 26.6.3
- d) the non-Conductor role in Configuration of triggers specified in 26.8

26.9.3 Additional requirements of a specific conformance-class specified for this Type

- a) The number and channel-type of channels supported, as described in 26.4
- b) The number and trigger-type of triggers supported, as described in 26.6.3
- c) The maximum depth of the real network topology supported, and the resultant length of the DA and SA fields, as described in 26.5.3.3.4.
- d) The maximum number of user-defined group-DLE-addresses per local link supported, as described in 26.5.3.3.1.4
- e) the Bus-cycle-length, as described in 26.2.1
- f) The maximum number of Time-stamp-buffers per DLSAP that supports GPA or GPC channels, as described in 26.6.2
- g) the number of forwarding-ports per Bridge and their directionality, as described in 26.7
- h) use and encoding of the other-status field, as defined in 26.4, 26.5 and 26.8
- i) use of the low-order-failed-octet-number field, as defined in 26.5 and 26.8
- j) the minimum accuracy, including environmental influences, of the time reference used for implementing the slot-clock, as described in 26.2.0. The maximum inaccuracy of this reference in a Host node shall be 10^{-5} . The maximum inaccuracy of this reference in a non-Host node shall be 5×10^{-5} .
- k) the maximum DLSDU size for ExSCAN, GPA and GPC channels, as described in 26.4.2.2.2.3
- l) The minimum value of max-data-length, as specified in 27.1.1.

26.9.4 Protocol Implementation Conformance Statement (PICS)

The implementor shall prepare a PICS for each distinguishable node type. This PICS shall state that, and how, the implementation conforms with the requirements of 26.9.1 and 26.9.2. It shall also state which of the requirements of 26.9.3 it complies with, and the number of supported entities in each category.

27 Type 6: MAC elements of procedure

This clause specifies Bus Startup, Bus Operation and Media Access Control.

It also specifies the DL view of the interface to the PhL. The DLL – PhL interface is detailed in IEC 61158-2.

27.1 Procedures related to bus startup and bus operation

The Conductor DLE and all non-Conductor DLEs cooperate to synchronize bus startup and operation by use of the Bus-PDU.

27.1.1 Formation, usage and encoding of the bus-PDU

The Bus-PDU is the mechanism by which the Conductor' DLE instructs non-Conductor DLEs as to the Bus-state, bus timing, Bus-time and Bus-synchronization.

The Conductor uses the Bus-PDU field to inform all nodes of the current Bus-State and Bus-Config.

By changing the Bus-PDU the Conductor can transition the bus to a new operating regime seamlessly, by directing all non-Conductor nodes through the TRANSITION state.

The Conductor distributes (a new) Bus-State, InterFrameGap, max-data-length, and Bus-sync-period to all node MACs by changing the encoded values corresponding to any or all of these item(s) in the Bus-Config field of its Bus-PDU to the desired value.

At bus start-up the Conductor automatically configures all of the above items in all MACs.

The Conductor can change any of the above items, for all nodes on the bus, on a dynamic basis by taking the bus (and thus each node) through the TRANSITION state.

The contents of Bus-PDU, as shown in Table 226, are in sequence: Bus-state, Bus-Config-field, NcycleLO and Sync-type.

Table 226 – Bus-PDU composition and encoding

Item	Bit positions	Encoding	Allowed Values
Sync-type	0	LOGICAL	see below
NcycleLO	1	USINT1	0 to 1
Bus-Config-field	13 to 2	BITSTRING	see below
Bus-state	15, 14	ENUMERATED	0 to 3
High order bit (Octet n, bit 7) of Bus-PDU ! last bit Transmitted			

a) Bus-state

the state of the Finite State Machine state shared by each DLE on the extended link

0 – START

1 – CHECK

2 – TRANSITION

3 – NORMAL

b) the Bus-Config-field is shown in Figure 139. It conveys three encoded values: IFG, maxDL and bus-sync-index.

bits 13 to 9	bits 8-4	bits 3 to 2
IFG	maxDL	bus-sync-index
0 to 31	0 to 31	0 to 3

Figure 139 – Bus-config field composition and encoding

Bus-Config, the bus timing parameters, are related to the Bus-Config-field as follows:

$$\text{InterFrameGap} = [\text{IFG} + 4] \quad (70)$$

$$\text{max-data-length} = [\text{maxDL} + 2] \quad (71)$$

$$\text{Bus-sync-period} = 2^{[\text{bus-sync-index} + 5]} \text{ slots} \quad (72)$$

c) NcycleLO

the low-order bit of the Conductor DLEs Bus-cycle-clock

d) Sync-type

TRUE – this is a Bus-end-sync slot

FALSE – this is a Bus-sync slot

This field is 1 bit long. It is transmitted as bit 0 of the Bus-PDU.

Prior to transmission of any Bus-sync or Bus-end-sync DLPDU, the Conductor sets the Sync-Type field to inform all other DLE's on this link as to whether this DLPDU is a Bus-sync or Bus-end-sync DLPDU.

27.1.2 Procedures related to the bus-cycle-counter

The master Bus-cycle-counter is an up-counter of Slots kept by the Conductor. It rolls-over to 0 at the end of the slot in which it reaches [Bus-cycle-length – 1] slots

$$\text{Bus-cycle-length} = 2^{\text{LBL}} \text{ slots} \quad (73)$$

27.1.2.1 Actions taken by the each host's DLE

At POWER-ON, each Host DLE

Initializes

Assumes the standardized bus timing parameters described in 26.5.2.2

Sets its local Bus-config = the above parameters

Assumes Bus-state = START

IF a Bus-end-sync DLPDU is received within [its local Host-number * Bus-cycle-length slot times]

NOTE The local Host-number is assigned to each Host by the system-manager. This number must be unique on the local link.

THEN

Acts as a non-Conductor

ELSE

This DLE assumes the role of Conductor

ENDIF

27.1.2.2 Actions taken by the conductor DLE

Following its election as the Conductor DLE

The Conductor's DLE maps the standardized bus timing parameters, its NcycleLO, Sync-type and Bus-state into the upcoming Bus-PDU as described in 27.1.1

The Conductor's Bus-configuration DLMS-user and DLE direct all non-conductor nodes through Bus startup and configuration, as described in 27.1, 26.5.2.3 and 26.8

27.1.2.2.1 Slot synchronization and Bus sequencing

IF the Conductor DLE's Bus-cycle-counter is incremented and then equals one less than a multiple of the Sync-Period, but does not equal $2^{L_{BL}} - 1$

NOTE 1 This indicates that a synchronization point has been reached and that a Bus-sync slot of the Bus-cycle has just begun.

The DLE transmits a Bus-sync DLPDU with Sync-type = FALSE

NOTE 2 The contents of the above Bus-PDU, as described in 27.1.1, are unchanged from the previous Bus-PDU

ENDIF

IF the Conductor DLE's Bus-cycle-counter is incremented and then equals $2^{L_{BL}} - 1$

NOTE 3 This indicates that the last (360 degree) Bus-end-sync slot of the Bus-cycle has just begun.

THEN

the DLE increments its local Bus-cycle-clock, and thus NcycleLO

IF the Bus-state = NORMAL

AND the Conductor's DLE has not been instructed by the Bus-configuration DLMS-user to change to a new Bus-Configuration

THEN

the DLE transmits a Bus-end-sync DLPDU, with Sync-type = TRUE

NOTE 4 The contents of the associated Bus-PDU, as described in 27.1.1, except for Sync-type and NcycleLO, are unchanged from the previous Bus-PDU

ENDIF

IF the Bus-state = NORMAL

AND the Conductor's DLE has been instructed by the Bus-configuration DLMS-user to change to a new Bus-Configuration

THEN

The DLE sets Bus-state = TRANSITION

the DLE transmits a Bus-end-sync DLPDU, as described in 27.1.2.2.1. The contents of the associated Bus-PDU, include:

the desired new Bus-config-field

Bus-state = TRANSITION

Sync-type = TRUE

the just incremented value of Ncycle-LO

The DLE repeats the desired new Bus-config-field, with Bus-state = CHECK or TRANSITION, as set previously, in all configured Bus-sync and the Bus-end-sync DLPDUs for $[2 + \text{max-Net-level}]$ additional Bus-cycles

NOTE 5 A total of three Bus-end-sync DLPDUs contain Bus-state = CHECK or TRANSITION. At least $[3 + \text{max-Net-level}]$ full rounds of Bus-sync DLPDUs also contain the above Bus-state.

IF the $[3 + \text{max-NET-level}]$ Bus-end-sync DLPDU is transmitted

THEN

The Conductor's Bus-configuration DLMS-user directs the DLE to move to the next Bus-state

IF the current Bus-state = CHECK

THEN

The next Bus-state = TRANSITION

ENDIF

IF the current Bus-state = TRANSITION

THEN

The next Bus-state = NORMAL

The DLE continues to repeat Bus-state = NORMAL and the desired new Bus-config-field, in all future Bus-sync and Bus-end-sync DLPDUs

ENDIF

The DLE sets Bus-state as specified above

ENDIF

ENDIF

ENDIF

27.1.2.3 Actions taken by each non-conductor DLE

At POWER-ON, each non-Conductor DLE

initializes

assumes the standardized bus timing parameters described in 26.5.2.2

assumes Bus-state = START

waits for DLM-connectionless DLPDUs sent by the Conductor on one of its offered startup-channels

NOTE These DLPDUs will sequence each non-conductor node through Bus startup and configuration, as described in 27.1, 26.5.2.3 and 26.8

27.1.2.3.1 Slot synchronization and bus sequencing

IF the node DLE receives a Bus-sync or Bus-end-sync DLPDU, as described in 27.1.1.

THEN

the DLE resets its own Slot-Clock counter at the synchronization point of the Bus-sync or Bus-end-sync DLPDU, as defined in 27.2.6.

the DLE extracts the value of NcycleLO and BUS-Config from the the Bus-PDU of the Bus-sync or Bus-end-sync DLPDU, as defined in 27.1.1.

the DLE sets its NcycleLO counter = that value

NOTE 1 NcycleLO is used, in conjunction with the current local Bus-cycle counter and Slot clock, by the DLE to establish a common sense of Bus-time, as described in 26.6.1. This common time is synchronized with all communicating nodes.

The DLE compares the just extracted value of Bus-config to its local Bus-config

IF the two values of Bus-Config differ

THEN

The DLE sets No-comm-node = TRUE

ENDIF

ELSE

IF this is a Host and it has received no Bus-end-sync DLPDU for [2 * its local Host-number * Bus-cycle-length Slot-times]

THEN

This DLE assumes the role of Conductor

The DLE assumes TRANSITION Bus-state

The Bus-configuration DLMS-user and DLE sequence the Bus back to the NORMAL Bus-state, as described in 27.1.3.3.

NOTE 2 This procedure elects a new conductor in case of current conductor fails.

ENDIF

END IF

IF the DLE receives a Bus-end-sync DLPDU, identified by Sync-type = TRUE in the Bus-PDU, as defined in 27.1.1

THEN

The DLE resets its own Bus-cycle-counter concurrent with SYN in the BSD field, as defined in IEC 61158-2 and 27.2.6.

NOTE 3 This synchronizes the node's channel scanning and trigger sequences with all other communicating nodes.

The DLE extracts NcycleLO and Bus-state from the just received Bus-PDU

IF the just extracted Bus-state = TRANSITION

the DLE copies all the Bus-config parameters and Bus-state into its local variables

the DLE assumes the mode of operation defined in Bus-state and Bus-config, as dictated by the Conductor in the Bus-PDU, as defined in 27.1.1.

ENDIF

IF the just extracted Bus-state = CHECK

the DLE copies Bus-state into its local variable

the DLE assumes the mode of operation defined in Bus-state, as dictated by the Conductor in the Bus-PDU, as defined in 27.1.1.

ENDIF

END IF

27.1.2.4 Actions taken by bridge MAC's

At the synchronization point, as described in 27.2.6, all bridge MACs

- a) assert their BUS-SYNC-OUT signals immediately after the conclusion of each Bus-sync PhpDU. This action is concurrent with SYN.
- b) assert their BUS-END-SYNC-OUT signals immediately after the conclusion in a Bus-end-sync PhpDU. This action is concurrent with SYN.

27.1.3 Procedures used to sequence the Bus

27.1.3.1 Completing START bus-state

IF the Conductor finishes configuration of all nodes and configuration-objects defined in the node list

THEN

The Conductor's Bus-configuration DLMS-user polls each of the startup-channels with a Node-DLMSDU and DLPDU.

This DLMSDU and the associated DLPDU user-data field have UUID = all 0s.

The Called-address and the associated DA are all 0s.

Any unconfigured non-Conductor nodes respond to this Node DLPDU and DLMSDU with a Node-check DLMSDU and DLPDU.

This DLMSDU and the associated DLPDU user-data field have UUID = the node's local UUID.

ENDIF

IF the Conductor's Bus-configuration DLMS-user receives such a Node-check DLMSDU from the DLE,

THEN

the Conductor's Bus-configuration DLMS-user enters this Node-check DLMSDU in the Bogus-node-list

NOTE If two or more unconfigured nodes share the same startup-channel, there will be a collision and the Conductor will not receive any Node-check DLMSDU.

ENDIF

IF the Conductor finishes the test for Bogus nodes on all startup-channels, as defined above

THEN

The conductor's Bus-configuration DLMS-user directs its DLE to enter the CHECK Bus-state

ENDIF

27.1.3.2 When in the CHECK bus-state

The Conductor sends at least 3 Bus-end-sync DLPDUs containing the configured/new Bus-Config while in CHECK state.

All nodes remain prepared to receive and transmit DLM-connectionless DLMSDUs.

NOTE This standard does not define any specific interactions during the CHECK bus-state. Future versions of this standard may define such actions.

27.1.3.3 When in the TRANSITION bus-state

The Conductor on the TOP link sends at least [3 + max-Net-level] Bus-end-sync DLPDUs containing the configured/new Bus-Config while in TRANSITION state.

NOTE 1 This procedure accounts for delays in propagation of Bus-state through the extended link.

NOTE 2 Use of the TRANSITION bus-state assists in node synchronization and reduces the probability that a node will attempt to communicate using the wrong Bus-Config or mode of operation.

NOTE 3 Non-conductor nodes in TRANSITION prepare for Connection mode communication. However, these nodes do not communicate.

Each non-Conductor DLE listens for the new Bus-Config in a Bus-end-sync DLPDU

IF a non-Conductor DLE receives at least one Bus-end-sync DLPDU with Bus-state = TRANSITION

THEN

/ * The DLE is now prepared to enter NORMAL Bus-state */

IF a non-Conductor DLE receives at least one Bus-end-sync DLPDU with Bus-state = NORMAL

THEN

The non-Conductor DLE immediately enters Bus-state = NORMAL

ENDIF

ENDIF

27.1.3.4 When in the NORMAL bus-state

The Conductor and all non-conductor nodes

- a) use the configured/new Bus-Config from the Bus-PDU
- b) use Connection mode communications, as described in 26.4
- c) activate the configured channel, trigger and half-forwarding-port communications, as described in 26.8.

27.2 Medium access control (MAC)

The MAC in each node interfaces to the PhL as described in IEC 61158-2.

The MAC:

- a) forms and decomposes unexpanded DLPDUs, which are used in Transfer DLPDUs and DLM-connectionless DLPDUs, and Bus-PDUs which are used in Bus-sync and Bus-end-sync DLPDUs. It passes these as the leading PhIDUs of the expanded DLPDU to or from the PhL.
- b) encodes/decodes and maps/extracts InterFrameGap, max-data-length and Bus-sync-period to/from the Bus-sync or Bus-end-sync DLPDUs, as described in 27.1.1. It also notifies Station Management to enforce PAD and InterFrameGap on PhPDUs formed by the PhL.
- c) forms and verifies the FCS polynomial which is passed to or from the PhL as the trailing two PhIDUs of the expanded DLPDU, as described below, leading to:

$$\text{max-DLPDU-length} = [\text{max-data-length} + 2] \quad (74)$$

- d) synchronizes the PhL in the transmission and reception of PhPDUs, as described below.
- e) enforces the synchronous actions of Data Strokes and Triggers, as described below.
- f) enforces the synchronous actions of Bridge MACs, as specified in 27.1.2.4

In performing these functions the MAC interacts with the PhL as specified below:

27.2.1 PAD

When a new Bus-config is set or received by a DLE, as described in 27.1.1, Station Management sets PAD as follows using a Ph-SETVALUE request.

$$\text{PAD} = \text{max-data-length} - 2$$

NOTE 1 PAD is used to pad out Bus-sync and Bus-end-sync PhPDUs to the configured DLPDU length when Bus-PDU length is less than max-data-length. This guarantees the synchronous arrival of the SYN intervals in Bus-sync and Bus-end-sync PhPDUs, and thus the synchronous behavior of the bus.

NOTE 2 Transfer and DLM-connectionless PhPDUs associated with DLPDUs of less than maximal length are never padded, since use of the Bus-sync channel guarantees reliable timing and synchronization of each node's Slot-Clock.

27.2.2 InterFrameGap

When a new Bus-config is set or received by a DLE, as described in 27.1.1, Station Management sets InterFrameGap to a value computed from the encoded Bus-config value by using a Ph-SETVALUE request.

27.2.3 FCS

The MAC performs the FCS generation and checking algorithms, as specified in 5.1 and Table 1.

The FCS generation and residue calculations are performed on the unexpanded DLPDU for Transfer or DLM-connectionless DLPDUs or the Bus-PDU-field for Bus-sync or Bus-end-sync DLPDUs.

If the receiver's residue does not match the expected residue, the receiving MAC discards the just received DLPDU.

NOTE The last bit of the FCS residual ends six bit-periods before the Ph-DATA indication that specifies END-OF-DATA.

27.2.4 Data strobe emission

If the last Ph-DATA indications of the PhPDU indicates END-OF-TRANSFER followed by END-OF-ACTIVITY or END-OF-TRANSFER-AND-ACTIVITY, the just received DLPDU is a Transfer DLPDU or a DLM-connectionless DLPDU.

Each MAC which has transferred a DLPDU user-data field or Bus-sync-field during the just completed slot, emits a Data-strobe immediately after receipt of this last Ph-DATA indication.

NOTE As specified in IEC 61158-2, the trigger point TR for Data-Strobes occurs immediately after reception of the last Ph-symbol of ETF delimiter.

27.2.5 Trigger synchronization

The setting of offset-bit in 26.8.2.2 is relative to rollover of the DLE's Slot-clock as described in 26.3.4.

27.2.6 Bus synchronization

If the last Ph-DATA indications of the PhPDU indicates END-OF-SYNC followed by END-OF-ACTIVITY or END-OF-SYNC-AND-ACTIVITY, the just received PhPDU is a Bus-sync DLPDU or Bus-end-sync DLPDU.

NOTE 1 As specified in IEC 61158-2, the synchronization point for all MACs, SYN, occurs immediately after reception of the last Ph-symbol of the BSD delimiter.

Actions taken at the synchronization point are specified in 27.1.2.2.1, 27.1.2.3.1 and 27.1.2.4.

NOTE 2 To meet the synchronous requirements of Type 6, all of the actions taken at the synchronization point must be performed with a jitter of ≤ 0.05 bit-periods relative to the other actions taken at the synchronization point, or relative to the same action when taken for the preceding Slot.

27.2.7 Transmission of DLPDUs

NOTE 1 This subclause discusses the coordinated actions of the DLL and PhL, in order to improve the reader's understanding. Nothing in this subclause is to imply a blurring of the DLL – PhL interface.

Each PhPDU is transmitted by the transmitting MAC and PhL as follows and in accordance with IEC 61158-2:

- a) When the MAC senses that this node will transmit in the next slot on the bus, it prepares the associated DLPDU and passes that DLPDU to the PhL as described below.

NOTE 2 The MAC may use the current values of the Bus-cycle-counter and Slot-clock in sensing its transmission opportunities.

- b) The first Ph-DATA request of this PhPDU, specifying START-OF ACTIVITY, is passed in synchronization with the roll-over of the Slot-clock.
- c) The next Ph-DATA request of this PhPDU, specifying DATA, is passed to the PhL after [InterFrameGap + 10] bit periods have elapsed.
- d) Subsequent Ph-DATA requests of this PhPDU, specifying DATA, and one concluding Ph-DATA request, specifying END-OF-DATA-AND-TRANSFER (for Transfer or DLM-connectionless DLPDUs), or END-OF-DATA-AND-SYNC (for Bus-sync or Bus-end-sync DLPDUs), are passed one for each bit period.
- e) The MAC continues to sense whether this node's MAC will transmit in the next slot on the bus, until the condition of a) is again fulfilled. On sensing this condition, it repeats the procedure of this clause, starting with step a).

NOTE 3 To meet the synchronous requirements of Type 6, all of the actions listed in this subclause must be performed with a jitter of ≤ 0.05 bit-periods relative to the other actions of this clause, or relative to the same action when taken for the preceding Slot.

27.2.8 Reception of DLPDUs

NOTE 1 This subclause discusses the coordinated actions of the DLL and PhL, in order to improve the reader's understanding. Nothing in this subclause is to imply a blurring of the DLL – PhL interface.

Each DLPDU is received by the receiving MAC and PhL as follows and in accordance with IEC 61158-2:

- a) when the PhL begins to receive a PhPDU, it emits a first Ph-DATA indication, specifying START-OF-ACTIVITY
- b) if this first Ph-DATA indication occurs within a slot that the DLE (MAC) senses it should receive, and beyond the SILENCE period specified by InterFrameGap, the MAC actively receives the DLPDU of this PhPDU as reported by the PhL.

NOTE 2 The MAC may use the current values of the Bus-cycle-counter and Slot-clock in sensing its reception opportunities.

- c) when the MAC receives a Ph-DATA indication, specifying END-OF-DATA, it terminates its reception of the DLPDU and performs the FCS check. If this check is successful, it continues to monitor the output of the PhL for one or two final Ph-DATA indications.

The final Ph-DATA indication(s) specify whether the just received PhPDU was a Transfer or DLM-connectionless DLPDU (END-OF-TRANSFER or END-OF-TRANSFER-AND-ACTIVITY) or a Bus-sync or Bus-end-sync DLPDU (END-OF-SYNC or END-OF-SYNC-AND-ACTIVITY) The end of the first Ph-DATA indication denotes the TR, see 27.2.4 and 27.3.5, or SYN, see 27.2.6, timing markers of this PhPDU.

28 Type 7: Overview of the DL-protocol

28.1 Overall description of medium allocation

An element known as the **Bus Arbitrator (BA)** controls the right of each data producer to access the medium by emitting a DLPDU containing a DLCEP-identifier. At any given instant there should be only one active Bus Arbitrator in each local link.

NOTE The term "data producer" designates the sole station connected to the local link that is recognized as having the responsibility of emitting the data associated with the identifier DLPDU circulating on the medium.

Each transaction belongs to one of the three medium allocation classes defined below:

- cyclical exchange of variables, requests, or messages,
- explicit request for variable exchange,
- explicit request for message transfer.

For cyclical variable exchanges a basic transaction is made up of the following phases. The Bus Arbitrator broadcasts a variable identifier DLPDU. The sole producer of the information required then broadcasts a variable response DLPDU. During this phase consumers take the information from the local link. Figure 140 shows the various phases of a variable exchange transaction. When one transaction has been completed the Bus Arbitrator begins the following transaction according to guidelines defined when the system is configured.

During a cyclical variable exchange the information producer can, using the response DLPDU, transmit to the BA an explicit request for the exchange of variables or messages.

For an explicit request for a variable exchange, a basic transaction is made up of the following phases: The Bus Arbitrator broadcasts a request identifier DLPDU. Then the initiator of the request broadcasts a request response DLPDU. One or more transactions identical to the cyclical variables exchange transaction then follow.

For message transfers a basic transaction consists of the following phases: The Bus Arbitrator broadcasts a message identifier DLPDU. Then the message response DLPDU is exchanged between the communicating entities. This DLPDU may or may not be followed by an acknowledgement DLPDU. The source entity then transmits an end of transaction indication DLPDU to the Bus Arbitrator.

The time interval separating the reception or emission of the end of one DLPDU and the emission or reception of the following DLPDU is known as the station's turnaround time, whether the station's function be that of a producer/consumer or Bus Arbitrator.

A more detailed definition of turnaround time is given in 3.10.27. The impact of turnaround time on Data Link timers is described in 29.6.2.

The role of the Bus Arbitrator is to "give the floor" to each data producer, taking into account the services required for the type of data (according to the three medium allocation classes defined above).

The Bus Arbitrator thus has three types of functions:

- periodic scanning of variables or periodic triggering of variable and message exchanges,
- triggered scanning of variables,
- triggered scanning of messages.

In addition, the Bus Arbitrator can provide a synchronization function in order to guarantee the constant length of scanning cycles.

Each type of scanning takes place in a specific window, that is, respectively in a periodic window, an aperiodic variables window, an aperiodic messages window, or a synchronization window, as shown in Figure 140. These four windows constitute a basic scanning cycle.

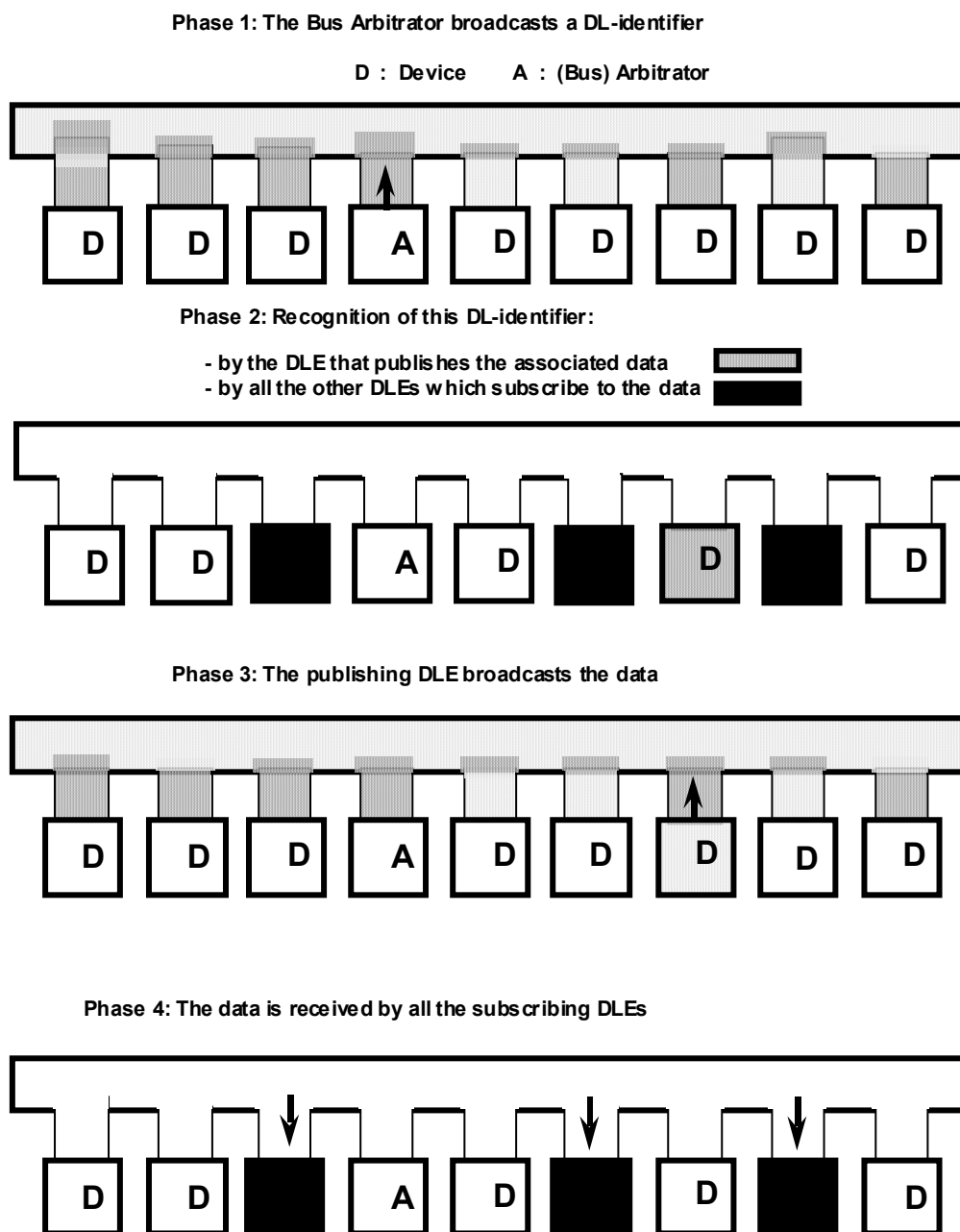


Figure 140 – General description of medium allocation

The medium access technique has the following characteristics:

- broadcasting of identified variables,
- increased efficiency in cyclical variable exchanges,
- parameters for medium sharing can be set by the user when the system is configured,
- guaranteed access time for cyclical variable exchanges, under all circumstances and regardless of the number of requests for triggered variable and/or message exchanges.
- possibility of triggering a transaction in accordance with a global clock, that is a clock that indicates the same time for all stations.

In addition, the medium access technique makes it possible to:

- give cyclical exchanges highest priority,
- respect the scanning period associated with each variable,
- give different priorities to triggered messages transfers and variable exchanges. These transactions are triggered in adjustable windows: the lengths of the "aperiodic variable" and "aperiodic message" windows are defined in terms of maximum limits set by the user when the system is configured.
- change the priority of aperiodic transactions by inserting them in the periodic window.

28.2 Types of entities

28.2.1 Producer/consumer entity

DLL services use various types of DLPDUs. Each DLPDU type is defined in the control field of the DLPDU. Interactions between a specific service and DLPDU types will be described in the detailed specifications of each service.

NOTE 29.5 describes all types of DLPDUs and also explains the use of various fields.

The functioning of an entity belonging to the highest conformance class requires:

- a mechanism for analyzing DLPDU type (use of the control field),
- a table of identifiers recognized upon emission,
- a table of identifiers recognized upon reception (both tables are defined at the interface between the DLL and System Management),

a mechanism that provides read/write access to variables and messages.

The conceptual model of a Data Link producer/consumer entity includes various buffers: queues needed to provide the services offered by the DLL, as shown in .Figure 141.

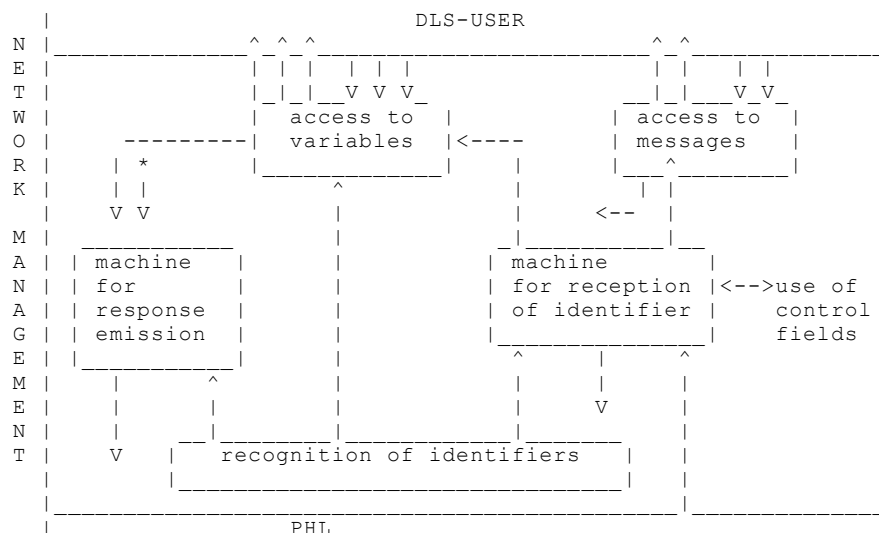


Figure 141 – Internal structure of a producer/consumer entity

The following are associated with each identifier produced:

- a buffer called B_DATprod, which contains the value of the variable produced,
- a buffer called B_REQ, containing the list of identifiers that are the object of an explicit request for a buffer transfer, if the identifier has been reserved for the explicit request for buffer transfer service,
- a queue called Q_MSGcyc, which contains messages to be emitted that are associated with cyclical message transfer, if the identifier has been reserved for cyclical message transfer,

- an indicator of the validity of the identifier produced (System Management),
- for each of the B_DATprod and B_REQ buffers, an availability indicator,
- an indicator associated with Q_MSGcyc: queue filled or not,
- indicators used to manage explicit requests for variable exchange and message transfer requests:
- explicit request for buffer transfer in progress (RQ),
- priority associated with an explicit request (PR),
- scope of the explicit request (RQ_INHIBIT)
- message transfer request in progress (MSG) if the identifier has been reserved for aperiodic message transfer,
- reference to the queue of messages to be emitted associated with aperiodic message transfer.

The following are associated with each identifier consumed:

- a buffer called B_DATcons, which contains the value of the variable consumed,
- an indicator of the validity of the identifier consumed (System Management),
- an indicator linked to the management of B_DATcons: availability of the buffer (access conflict),

NOTE The buffer availability indicator provides information concerning the integrity of data manipulated by service primitives.

Each entity that supports a message transfer request service also uses:

- a queue called Q_MSGaper, which is associated with aperiodic message transfer and contains messages to be emitted,
- a queue called Q_MSGrec that contains messages received.
- an indicator of queue status (full or not) is associated with the queue reserved for aperiodic message transfer.

In addition, each entity that supports acknowledged message service has the following characteristics:

- value of the source entity's even/odd bit
- maximum value of the restart counter
- current value of the restart counter

The following are associated with the queue Q_MSGrec:

- an indicator of queue status (full or not)
- value of the destination entity's even/odd bit
- value of the stored source address

If the entity also supports the free explicit request for buffer transfer service, there are two global queues for holding free explicit requests for buffer transfer:

- Q_REQ1 is associated with urgent requests,
- Q_REQ2 is associated with normal requests.

A status indicator (full or not) is associated with each queue.

Figure 142 shows many of these associated buffers and queues.

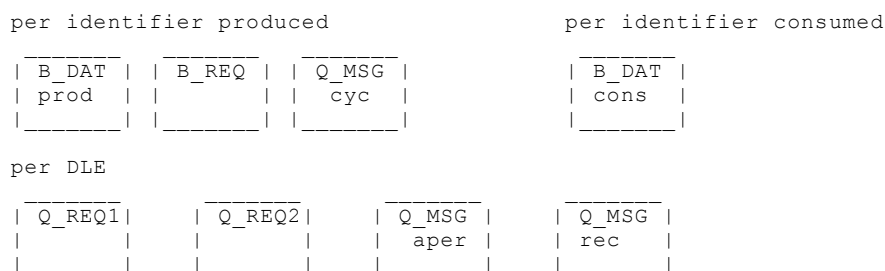


Figure 142 – Associated buffers and queues

Timers associated with the Data Link protocol are also managed by the producer/consumer entity (see 29.6).

Annex G presents an object model that defines the attributes of a DLCEP-identifier.

28.2.2 Bus arbitrator entity

The function provided by the Bus Arbitrator consists of "giving permission to speak" to each data producer. This permission is granted for three types of scanning:

- periodic or triggered periodic scanning of variables and messages,
- triggered scanning of variables,
- triggered scanning of messages.

The Bus Arbitrator also provides the following functions:

- chaining of basic transactions,
- chaining of the various types of scanning,
- analysis of DLPDU control fields (the control field carries aperiodic variable and message requests),
- filling of aperiodic windows in accordance with these requests.

NOTE A basic transaction is made up of the succession of DLPDUs related to a single service.

In addition, the Bus Arbitrator can provide a synchronization function to guarantee the constant length of a basic scanning cycle.

Each type of scanning takes place in a special window: respectively in a periodic window, an aperiodic variables window, an aperiodic messages window, or a synchronization window.

The four windows constitute a basic scanning cycle.

Figure 143 provides an overview of the Bus Arbitrator structure. A more complete description of the Bus Arbitrator is given in 31.4.

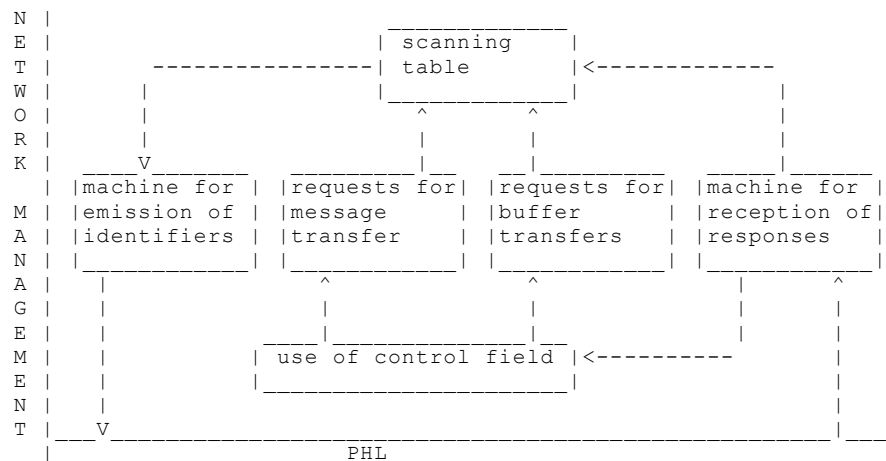


Figure 143 – Internal structure of a bus arbitrator

The conceptual model of the Bus Arbitrator details the various tables and queues needed to provide the services offered by the DLL.

The Bus Arbitrator manages according to system configuration, as shown in Figure 144:

- a scanning table with static portions and portions that are modified dynamically during scanning,
- a recovery buffer for the triggered periodic scanning of variables,
- a queue called Q_IDRQ1 for urgent explicit requests for buffer transfer,
- a queue called Q_IDRQ2 for normal explicit requests for buffer transfer,
- a queue for aperiodic transfers in progress (Q_RPRQ),
- a queue called Q_IDMSG for requests for message transfer,
- a basic padding sequence used to fill the synchronization window.

NOTE The management algorithm and/or calculation of these tables should provide equal access rights for the various entities, that is, starving certain entities should be avoided.

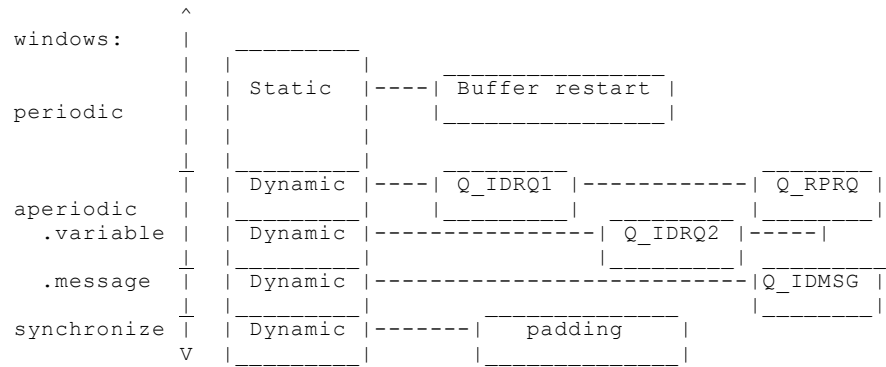


Figure 144 – Polling BA Table

28.3 Addressing

28.3.1 DLSAP-user relationship

The role of the DLL is to transmit information reliably and in accordance with a transmission protocol.

Within a single physical station several user entities have access to DLL services. Each of these entities uses a service access point (SAP) belonging to the DLL, in conformance with the ISO's static model in which each entity (N+1) is recognized by the address of the service access point (N) to which the entity is statically attached.

Several associations between user entities are possible on the level of access points to DLL services.

Thus, in accordance with the ISO model, a user entity (N+1) requests the establishment of a connection (N) in order to communicate with another user entity (N+1). The (N) entities, each of which is linked to one of the two (N+1) entities, create and manage this connection after negotiating it.

All exchanges between the entities (N+1) thus associated take place using this connection. The connection is identified locally at each SAP by a specific connection end point identifier (CEPi).

The number of DLSAPs corresponds to the number of entities in a station that are potential users of DLL services.

The number of CEPi per DLSAP corresponds to the number of potential communication links allocated to the (N+1) entity linked to the DLSAP.

28.3.2 OSI model relationship

This DLL follows the same rules as ISO/IEC 7498. Each potential user entity calls upon the services of the DLL through a DLSAP.

The DLL places two distinct addressing spaces as the disposal of the DLS-user:

- the addressing space concerning buffer transfer, each identifier being coded in 16 bits,
- the addressing space concerning message exchanges, which enables addressing messaging DLS-users, each address being coded in 24 bits.

The identifiers, encoded in 16 bits, allow addressing buffers within a local link, whereas the messaging addresses, encoded in 24 bits are meant to identify all the messaging DLS-users of the DL-subnetwork, that is those of all the DL-segments of the DL-subnetwork. Consequently, a communication system composed of n local links will have a single messaging address space but will have n identifier spaces to address the buffers.

Figure 145 illustrates this usage.

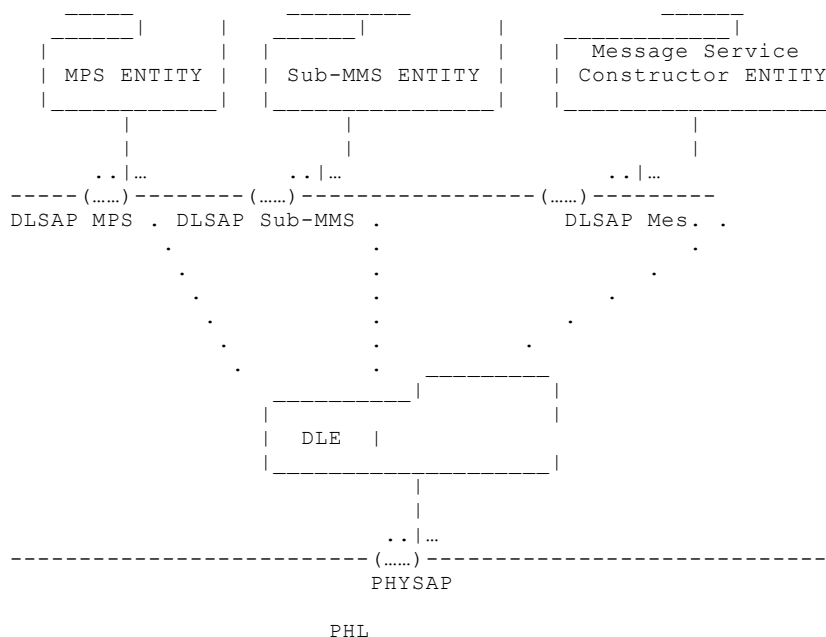


Figure 145 – Addressing scheme

28.3.2.1 Buffer transfer addressing

For reasons relating to performance and determinism, all associations between the various DLL user entities are known and defined when the system is configured.

Thus upon configuration the number of DLSAP is known, as well as the number of connections within the DLSAP.

This DLL uses broadcasting as its mode of transmission over the physical support. There is one producer and one or more consumers of each piece of information transmitted.

The connections within each DLSAP are multipoint connections providing unilateral communication (connections have more than two extremities, and over these connections data communication occurs in a single direction set in advance: from the producer to the consumers).

Connection end point identifiers (CEPi) that are known as identifiers and are encoded using 16 bits identify associations between user entities.

Thus CEPis are not recognized locally within a DLSAP, but globally throughout the DL-subnetwork, and the role of the DLSAP address is less important in the addressing model.

Thus the concept of an DLSAP is not the same as in the general ISO model. This DLL represents a group of between 1 and n identifiers employed by a user entity attached to the DLL. On the other hand, a DLSAP does not address the Application entity directly, but through the use of identifiers.

Each connection (identifier) can be used differently (according to the configuration and periodic or aperiodic services chosen) to communicate with other stations (variable transfers) or with the Bus Arbitrator (requests for aperiodic transfers).

28.3.2.2 Message exchange addressing

28.3.2.2.1 Addressing requirements

The addressing of the messaging DLS-users must allow meeting the usual communication requirement such as indicated in the ISO and which are as follows:

- point-to-point communication between two DLS-users, whether or not located in the same local link,
- multipoint communication between an entity and all the entities of a group, the latter being located in the same equipment or the local link or the extended link,
- communication by distribution between a DLS-user and all the other DLS-users of a DLE, a local link or the extended link.

NOTE This corresponds to the reservation of a particular group containing all the entities of a device, a local link or the extended link. The addressing of the messaging DLS-users must distinguish between two types of addresses:

- the physical address, in order to identify an application entity with respect to its physical location, by means of the DL-segment number and the number of the station where it is located,
- the logic address, in order to identify an application entity whose physical location will not be required.

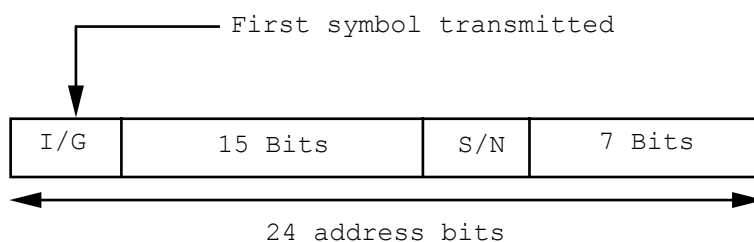
The notions of restricted groups on the DL-subnetwork with respect to groups of DLS-users in a device or in a DL-segment, have been defined with an aim to optimize the flow on the bus. Thus when addressing a group, it is thus necessary to have the possibility of a filter on the level of each bridge. The function of this filter is to prevent distribution on all the DL-segments of the extended link when this is not necessary.

The addressing of the buffers limits the number of devices on a DL-segment to 256. This must be taken into account in the messaging addressing mode.

28.3.2.2.2 Encoding of the addresses

The source and addressee DLS-users exchanging messages are identified in a DL-subnetwork with the help of addresses which are found in the link protocol data units (RP_MSG_NOACK and RP_MSG_ACK).

In order to meet the individual addressing requirements of the DLS-users in a device, in a group of devices of a local link or a group of devices of the extended link, the addressing space offered by the 24 bits is divided as shown in Figure 146.



I/G designates an individual DLS-user (I) or a group of DLS-users (G)

S/N designates a group of DLS-users on a DL-segment (S) or on the DL-subnetwork (N)

Figure 146 – Address partitioning

The encoding of the individual addresses and the groups of addresses is shown in Table ~.

Table 227 – Individual and group address encoding

I/G	S/N	Type of addressing
0	0	Individual address
1	0	Address of a group in a DL-segment
not significant	1	Address of a group in the DL-subnetwork

The link layer thus offers an addressing space which is divided into two sub-spaces:

- one containing link addresses meant to identify DLS-users individually,
- the other meant to identify DLS-user groups.

The group addresses are themselves divided into two sub-spaces:

- the sub-space defining addresses of restricted groups, thus enabling identification of the DLS-users which belong to the same local link,
- the other space defining the addresses of generalized groups, thus enabling identification of the application identities which all belong globally to the DL-subnetwork.

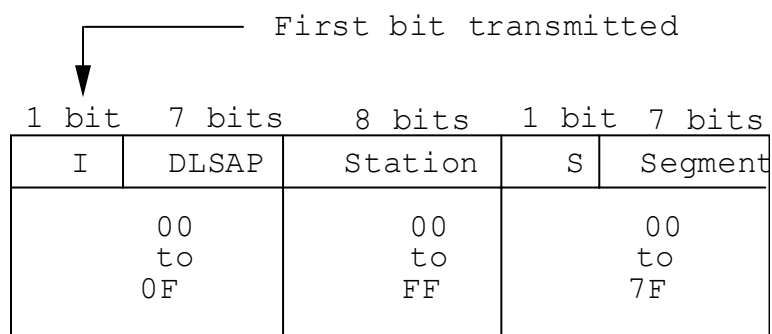
Each of these previously defined sub-spaces (individual addresses and addresses of restricted groups) are divided into:

- physical addresses,
- logical addresses.

28.3.2.2.2.1 Individual addressing

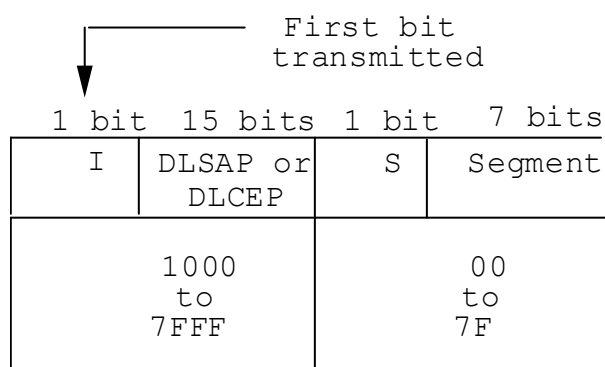
The individual addresses enabling a DLS-user to correspond with one and only one other DLS-user are structured as shown in Figure 147 and Figure 148:

a) Individual physical addresses:

**Figure 147 – Structure of an individual physical address**

This structure allows identifying 16 physical DLSAPs per device.

b) Individual logical addresses

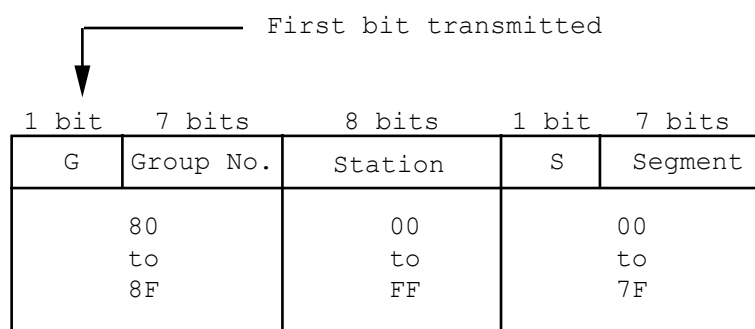
**Figure 148 – Structure of an individual logical address**

The addressing capacity is a total of 28K logical DLSAPs and DLCEPs per DL-segment.

28.3.2.2.2 Addressing of restricted groups

The restricted group addresses which enable a DLS-user to correspond with a group of DLS-users of the same device or of the same DL-segment are represented by their codes as shown in Figure 149 and Figure 150:

a) Addresses of restricted physical groups:

**Figure 149 – Structure of restricted physical group address**

The addressing capacity is then 16 numbers of physical restricted groups in a DLE. The distribution in a device is ensured by the specific group number, 8F.

b) Addresses of restricted logical groups:

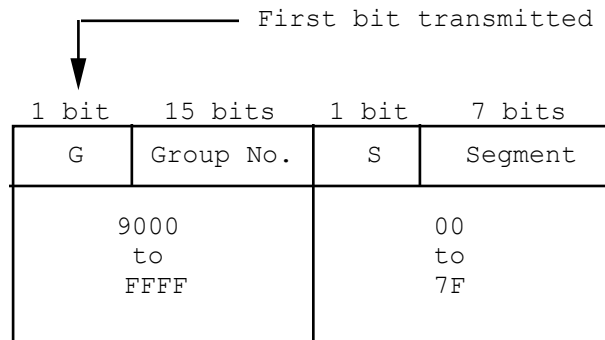


Figure 150 – Structure of restricted logical group address

This structure allows addressing 28K numbers of restricted logical groups in a local link. FFFF corresponds to the distribution group in a DL-segment.

28.3.2.2.3 Addressing of generalized groups

The addresses of generalized groups allow a DLS-user to correspond with a group of DLS-users on the extended link. These addresses are divided into three sub-areas:

- an addressing sub-area which can be used by all the equipment of the extended link,
- a reserved sub-area,
- a vendor addressing sub-area.

The encoding structure of these addresses is as shown in Figure 151:

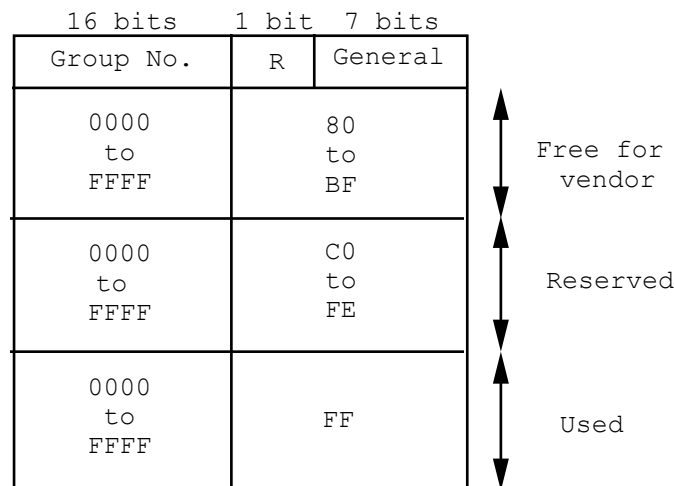


Figure 151 – Structure of a generalized group address

This addressing area allows identifying generalized 64K groups by sub-area element.

The distribution to the entire extended link is ensured by means of the particular group number FFFF.

28.3.2.3 Addressing capacity

The encoding rules offer the following possibilities for DL-subnetwork in terms of maximum capacity:

- 126 DL-segments in a DL-subnetwork;
- the "00" segment number is the DL-segment number by default;
- 256 DLEs per DL-segment in a DL-subnetwork;
- 16 physical DLSAPs in a DLE;
- 16 physical restricted group DL-addresses in a DLE;
- 28K logical DLSAPs in a local link;
- 28K logical restricted group DL-addresses in a local link;
- 64K generalized groups DL-addresses in the DL-subnetwork in the used sub-area.

The exact structure of the encoding of the beginning of an RP_MSG_XX message response DLPDU is as shown in Figure 152:

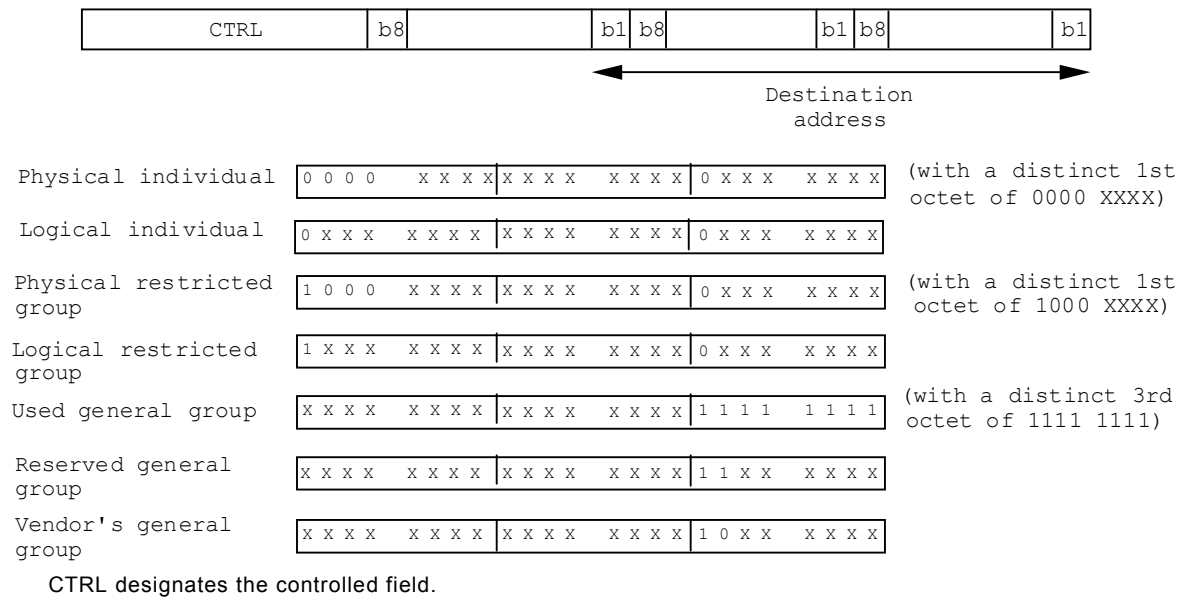


Figure 152 – Summary of address structure

The source address follows the destination address according to the same encoding.

28.4 Flow control

28.4.1 Variable exchanges

For the cyclical transfer of identified variables the principle of periodic scanning implies the replacement of an old variable value with a new variable independent of any reading access by the data consumer.

Management of flow control for these exchanges is definitively settled upon configuration:

- the Bus Arbitrator manages flow control by respecting the scanning period associated with the variable,
- stations manage flow control by associating a buffer with each identified variable.

This principle also applies to triggered variable transfers since the user is only interested in the most recent validated value, and since the user associates a buffer with each identified variable. In addition, upon configuration of a DLL system a time delimiter is defined for the aperiodic window for triggered variable transfers for each of the Bus Arbitrator's basic scanning cycles.

28.4.2 Message transfers

Flow control for message transfers is handled by the acknowledgement mechanism and by the queues for messages to be emitted and messages received.

When a DLL system is configured a time delimiter is defined for the aperiodic window for triggered message transfers for each of the Bus Arbitrator's basic scanning cycles.

For acknowledged message transfers a portion of flow control is provided upon emission since the DLL only accepts requests for acknowledged message transfers if there is available space in the queue for messages to be emitted that is specified in the request. With this type of transfer there is also flow control upon reception by the queue of messages received and through the transmission of an acknowledgement DLPDU.

28.4.3 Detection of DLPDU duplication/loss

Detection mechanisms provided apply to errors caused by communication problems (cut-off DLPDUs, FCS error) or by out-of-service stations.

These errors include:

- loss of a DLCEP-identifier DLPDU,
- lack of a response DLPDU (DLPDU lost or station absent),
- loss of a request response DLPDU,
- loss of a message response DLPDU,
- loss of a message acknowledgement DLPDU,
- loss of a DLPDU indicating the end of a message transaction.

and can cause either the loss or the duplication of data.

DLPDU losses are taken into account in the graphs of final state machines defining Data Link protocols.

Loss or duplication of DLPDUs can have the following impact on services:

- loss of a DLCEP-identifier DLPDU or a response concerning periodically exchanged variables is not usually detrimental since the value of the variable will be broadcast again during the next local link cycle.
- loss of an unacknowledged message DLPDU or of a DLPDU associated with a triggered variable exchange will only be handled by the upper layers.
- loss of an acknowledgement DLPDU results in a negative confirmation being sent to the initiating DLS-user.
- the concept of DLPDU duplication does not apply to periodically exchanged variables since the principle of cyclical variable refreshing implies that the same variable will be sent repeatedly.
- acknowledged message DLPDUs are protected from duplication by an even/odd numbering mechanism for messages.

- the duplication of a DLPDU associated with an explicit request for a buffer transfer will result in an additional overriding of the variable. This duplication is not detrimental since the concept of exchanging variables cyclically or upon explicit request is based on an asynchronous refreshing that is handled by the local link, and that the consumer cannot control. The semantics of data transmitted must be compatible with this working principle, that is, the variables exchange service is designed for the transmission of statuses, and not for the transmission of sequences of events.

28.5 Graphical representation

To be helpful for the reader some mechanisms are explained by a graphical representation. These representations are based on evaluation network built on an oriented graph with three types of nodes:

- The states, represented by a circle,
- The requests represented by a rectangle,
- The transitions, represented by a horizontal line.

The evaluation network is built according to the following principles, as illustrated in Figure 153:

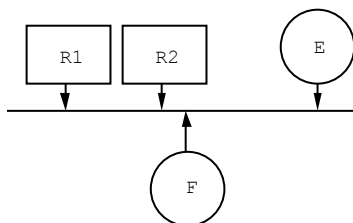


Figure 153 – Example of an evaluation net

When the described system is in state E, the arrival of request R1 or R2 results in a transition which switches it state F.

On the other hand, crossing a transition takes place, by convention, in zero time.

28.5.1 Simple actions

Simple actions can be associated with each transition, corresponding either to the transmission of requests or to the execution of local actions.

They are materially represented by a triangle, labeled with the name of the request or of the description of the executed action.

28.5.2 Conditions

The crossing of a transition can concern a condition. In this case, the oriented graph associated with the representation, shows a condition associated with the arc which precedes the transition.

A corollary of this representation allows defining choices for crossing a transition from the same initial state.

In the same way as for a simple transition, the transmission of an action can be associated with the crossing of a conditional transition.

29 Type 7: General structure and encoding of PhIDUs and DLPDUs and related elements of procedure

29.1 DLPDU formats and components

This subclause describes the structure of DLPDUs and the use of the various control fields within the DLPDU. The term "DLPDU" refers to the protocol data units exchanged by Data Link entities.

The basic structure of a DLPDU is as shown in Figure 154. The transmission and reception order is shown in Figure 155:

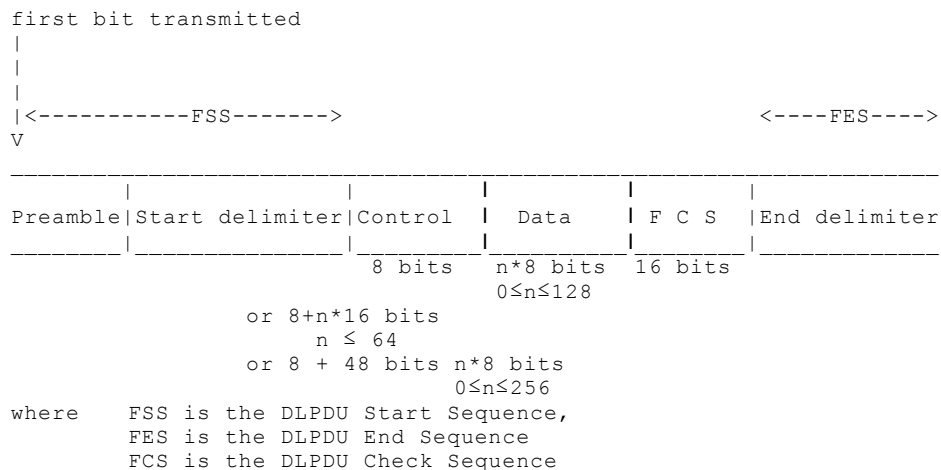


Figure 154 – Basic DLPDU structure

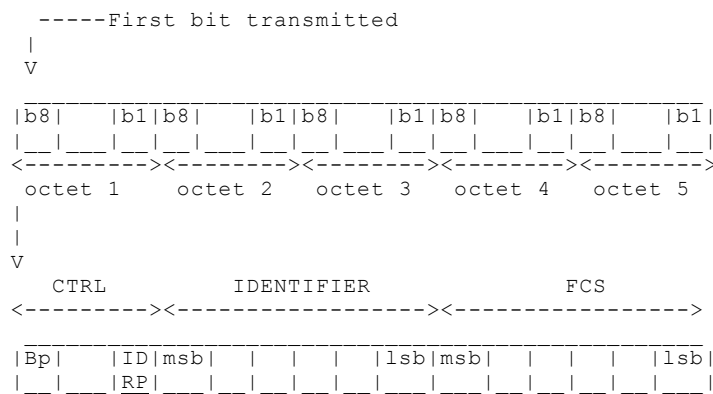


Figure 155 – DLPDU transmission / reception order

Octets are transmitted in the same order in which they are received (from the DLS-user or the PhL).

29.2 Description of each DLPDU component

29.2.1 Preamble

(See IEC 61158-2)

29.2.2 DLPDU delimiters

(See IEC 61158-2)

29.2.3 Control and addressing field

The control and addressing field specifies the type of DLPDU being exchanged:

- bit 1 of the control field indicates whether the DLPDU is a DLCEP-identifier DLPDU (bit 1 = 1) or a response DLPDU (bit 1 = 0)
- bits 2 through 7 of the control field, where each bit has a specific function, as shown in Table 228:
 - if bit 2 is set at 1 the DLPDU is related to a buffer transfer
 - if bit 3 is set at 1 the DLPDU is related to a message transfer
 - if bit 4 is set at 1 the DLPDU is related to an explicit request for a buffer transfer
 - if bit 5 is set at 1 the DLPDU is related to an acknowledgement
 - bit 6 indicates priority or the result of the acknowledgement
 - bit 7 set at 1 is used only to indicate an end of message transaction DLPDU.
- bit 8 of the control field (the first bit transmitted) is used only for a message response DLPDU or an acknowledgement response DLPDU. Bit 8 provides for even/odd numbering of messages to support duplicate message detection.

For a DLCEP-identifier DLPDU these bits specify the type of exchange in progress. For a response DLPDU these bits indicate the type of exchange in progress as well as requests for message transfer and explicit requests for variable exchanges, with their priorities.

Table 228 – DLPDU control-field coding

Bits	Identifier DLPDU
2 3 4 5 6 7	
1 0 0 0 0 x	allocation for identified variable
0 1 0 0 0 x	allocation for message
0 0 1 0 1 x	allocation for urgent explicit request
0 0 1 0 0 x	allocation for normal explicit request
	Response DLPDU
1 0 0 0 0 x	identified variable response
1 1 0 0 0 x	identified variable response + message request
1 0 1 0 1 x	identified variable response + explicit urgent request
1 0 1 0 0 x	identified variable response + explicit normal request
1 1 1 0 1 x	identified variable response + explicit urgent request + message request
1 1 1 0 0 x	identified variable response + explicit normal request + message request
0 1 0 1 0 x	acknowledged message
0 1 0 0 0 x	unacknowledged message
0 0 0 1 1 x	positive acknowledgement
0 0 0 1 0 x	negative acknowledgement
0 0 1 0 1 x	list of identifiers for an urgent request
0 0 1 0 0 x	list of identifiers for a normal request
0 0 0 0 1 x	end of message transaction

For identifier DLPDUs the control and addressing field is extended to 8 + 16 bits. In this case it includes a single DL-identifier that represents a local-link DL-address encoded in 16 bits.

For request response DLPDUs the control and addressing field is extended to 8 + × times 16 bits. In this case it includes the list of DL-identifiers associated with the variables requested.

For message response DLPDUs the control and addressing field is extended to 8 + 24 + 24 bits. In this case it includes two extended-link DL-addresses — the destination and source, respectively, of the message.

Table 229 gives the correspondence between the name and coding of the control field's bits.

Table 229 – Correspondence between name and coding of 8 bits in the control field

Bits	Associated name
1 2 3 4 5 6 7 8	
1 1 0 0 0 0 x x	ID_DAT
1 0 1 0 0 0 x x	ID_MSG
1 0 0 1 0 1 x x	ID_RQ1
1 0 0 1 0 0 x x	ID_RQ2
0 1 0 0 0 0 x x	RP_DAT
0 1 1 0 0 0 x x	RP_DAT_MSG
0 1 0 1 0 1 x x	RP_DAT_RQ1
0 1 0 1 0 0 x x	RP_DAT_RQ2
0 1 1 1 0 1 x x	RP_DAT_RQ1_MSG
0 1 1 1 0 0 x x	RP_DAT_RQ2_MSG
0 0 1 0 1 0 x N	RP_MSG_ACK
0 0 1 0 0 0 x x	RP_MSG_NOACK
0 0 0 0 1 1 x N	RP_ACK+
0 0 0 0 1 0 x N	RP_ACK-
0 0 0 1 0 1 x x	RP_RQ1
0 0 0 1 0 0 x x	RP_RQ2
0 0 0 0 0 0 1 x	RP_END
NOTE N takes the value 0 or 1; x values are ignored but should be 0	

29.2.4 DLS-user-data field

Only response DLPDUs include a DLS-user-data field. The DLS-user-data field contains:

- either the value of a variable
- or a message.

29.2.5 DLPDU check sequence (FCS)

See 5.1. The generator polynomial for this FCS, and the polynomial for the receiver's expected residue, are specified in Table 1.

29.3 PhIDU structure and encoding

Each PhIDU consists of Ph-interface-control-information (PhICI) and in some cases one octet of Ph-interface-data. When the DLE transmits a DLPDU, it computes a DLPDU check sequence for the DLPDU as specified in 29.2.5, concatenates the DLPDU and DLPDU check sequence, and transmits the concatenated pair as a sequence of PhIDUs as follows:

- a) The DLE issues a single Ph-DATA request primitive with PhICI specifying start-of-activity, and awaits the consequent Ph-DATA confirm primitive.

- b) The DLE issues a sequence of Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the DLPDU as Ph-interface-data, from first to last octet of the DLPDU, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.
- c) The DLE issues a sequence of two Ph-DATA request primitives with PhICI specifying DATA, each accompanied by one octet of the FCS as Ph-interface-data, from first to last octet of the FCS, and after each Ph-DATA request primitive awaits the consequent Ph-DATA confirm primitive.
- d) The DLE issues a single Ph-DATA request primitive with PhICI specifying END-OF-DATA-AND-ACTIVITY, and awaits the consequent Ph-DATA confirm primitive.

The DLE forms a received DLPDU by concatenating the sequence of octets received as Ph-interface-control-information of consecutive Ph-DATA indications, computing a DLPDU check sequence for those received octets as specified in 29.2.5, and checks the syndrome of the computed FCS for correctness as follows:

- e) The DLE receives a single Ph-DATA indication primitive with PhICI specifying START-OF-ACTIVITY, and initializes its computation of an FCS for the received DLPDU.
- f) The DLE receives a sequence of Ph-DATA indication primitives with PhICI specifying DATA, each accompanied by one octet of the received DLPDU as Ph-interface-data, incrementally computes an FCS on the received octet, and concatenates all but the last two of those received octets to form the received DLPDU.
- g) The DLE receives a single Ph-DATA indication primitive with PhICI specifying either END-OF-DATA, END-OF-DATA-AND-ACTIVITY or END-OF-ACTIVITY, and checks the syndrome of the computed FCS for correctness:
 - 1) If the PhICI specified END-OF-DATA or END-OF-DATA-AND-ACTIVITY, and the computed FCS syndrome was correct, then the DLE reports the reconstructed DLPDU and the two octets of received FCS as a correctly-received DLPDU suitable for further analysis.
 - 2) Otherwise the DLE increments its management statistics to reflect the erroneously received DLPDU.

29.4 Common DLPDU structure, encoding and elements of procedure

Each DLPDU consists of a DLPDU control field which specifies the type of DLPDU and conveys small size (fractional octet) parameters of the DLPDU; zero to two explicit address fields, each containing a DL-address, all of the same length; and for most DLPDUs, a user data field conveying all or part of a DLSDU. To this is appended before transmission, and removed after reception, an FCS field used to check the integrity of the received DLPDU.

29.5 Valid DLPDU types

For easier reading, DLPDUs are described without their DLPDU start and DLPDU end sequences.

For each type of DLPDU a name is given for the control field. This name is shown in parenthesis.

The correspondence between this name and the 8-bit control field code is given in the table below.

For each type of DLPDU we have indicated the separation between fields that contain Data Link protocol control information (DLPCI) and fields containing data (DLSDU).

29.5.1 Identifier DLPDU

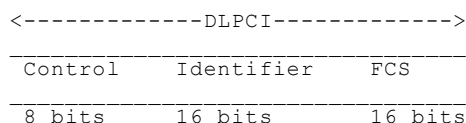


Figure 156 – Identifier DLPDU

There are four types of identifier DLPDU, whose basic structure is shown in Figure 156:

- medium allocation for identified variables (ID_DAT)
- medium allocation for message (ID_MSG)
- medium allocation for urgent explicit request (ID_RQ1)
- medium allocation for normal explicit request (ID_RQ2)

29.5.2 Variable response DLPDU

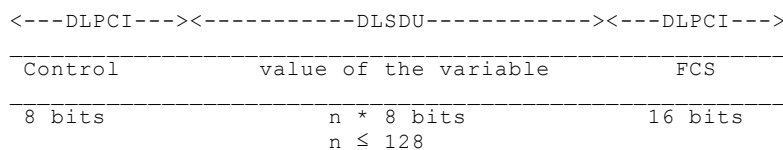


Figure 157 – Variable response DLPDU

There are six types of variable value response DLPDU, whose basic structure is shown in Figure 157:

- identified variables (RP_DAT)
- identified variables + message request (RP_DAT_MSG)
- identified variables + urgent explicit request (RP_DAT_RQ1)
- identified variables + normal explicit request (RP_DAT_RQ2)
- identified variables + urgent explicit request + message request (RP_DAT_RQ1_MSG)
- identified variables + normal explicit request + message request (RP_DAT_RQ2_MSG)

29.5.3 Request response DLPDU

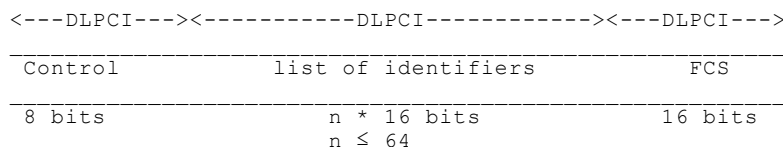


Figure 158 – Request response DLPDU

There are two types of request response DLPDU, whose basic structure is shown in Figure 158:

- list of urgent identifiers (RP_RQ1)
- list of normal identifiers (RP_RQ2)

29.5.4 Message response DLPDU

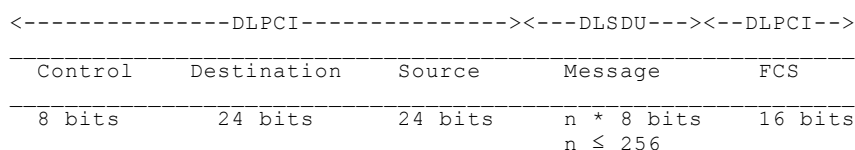


Figure 159 – Message response DLPDU

There are two types of message response DLPDU, whose basic structure is shown in Figure 159:

- acknowledged message (RP_MSG_ACK)
- unacknowledged message (RP_MSG_NOACK)

29.5.5 Acknowledgement response DLPDU

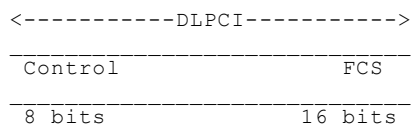


Figure 160 – Acknowledgement response DLPDU

There are two types of acknowledgement response DLPDU, whose basic structure is shown in Figure 160:

- positive acknowledgement (RP_ACK+) indicates that the message has been correctly received by the remote DLE,
- negative acknowledgement (RP_ACK-) indicates that the distant entity's queue for received message is filled.

29.5.6 End of message transaction response DLPDU

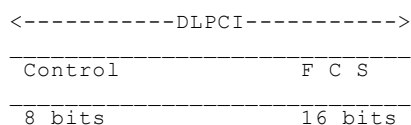


Figure 161 – End of message transaction response DLPDU

There is one type of end of message transaction response DLPDU, whose basic structure is shown in Figure 161:

NOTE The chaining of basic DLPDUs is indicated in the specifications for each service.

The RP_END DLPDU involves 2 entities: the source entity and the Bus Arbitrator. The source entity returns control of the local link to the Bus Arbitrator at the end of a message transaction.

29.6 DLL timers

This subclause briefly describes the timers involved in the description of Data Link final state devices, proposes relationships between the various timers, and describes the importance of a station's turnaround time in the definition of its timers.

29.6.1 Common criteria

The criterion common to all stations for launching timers is the absence of activity on the local link.

Two signals are used to evaluate this criterion: "Absence of activity" (SILENCE) and "Signal present" (BUSY).

A reference timer T0 is defined as corresponding to the maximum silent time permitted on a local link.

The T0 timer is a global systems parameter.

T0 is activated upon reception of an indication of the absence of activity. This indication is emitted by the Type 2 PhL (the primitive PhL-DATA indication (SILENCE)).

T0 is deactivated by any correct functioning of the local link, upon reception of the indication of signal presence. This indication is emitted by the Type 2 PhL (the primitive PhL-DATA indication (BUSY)).

The maximum silent time permitted on the local link, that is, T0 is linked to the turnaround time of stations in the system.

29.6.2 Turnaround time

The reception or emission of the last MAC symbol is signaled by a SILENCE indication from the PhL. Likewise, the reception or emission of the first MAC symbol is signaled by an BUSY indication from the PhL.

The time lapsed between these two signals within a single station defines the station turnaround time (STT), whether the station's function be that of producer/consumer or Bus Arbitrator.

Turnaround time takes into account parameters that are local to the station and parameters that are associated with the system to which the station belongs. These parameters are:

- physical reaction time (PRT): the time the PhL needs to detect the end or beginning of activity. These times depend on physical local link parameters:
 - station: time needed to detect the absence or presence of a signal
 - support: time needed to traverse the media and active copper or optical stars
- latency time (TI) resulting from processing time in the station.

TI takes on various values depending on the protocol situation in which the station finds itself:

- a) the station has emitted a DLPDU and will emit the next DLPDU,
- b) the station has received a DLPDU and will emit the next DLPDU,
- c) the station has emitted a DLPDU and will receive the next DLPDU,
- d) the station has received a DLPDU and will receive the next DLPDU.

All the parameters combine to form a minimum station turnaround time (under the most favorable conditions) and a maximum station turnaround time (under the least favorable conditions).

For the system to function correctly the following criteria must be respected:

Rule 1) a station that receives a DLPDU must always have a shorter turnaround time than the station that emitted the DLPDU, regardless of the two stations (receiving station, emitting station) involved.

This condition is expressed by:

$$\text{Max}(i) \text{ STT receiving station } i \leq \text{Min}(j) \text{ STT emitting station } j$$

Rule 2) minimum turnaround time corresponds to the maximum value of physical reaction time PRT.

Rule 3) given the impact of a station's turnaround time on transmission yield (use of the bandwidth), it is important that maximum STT take yield criteria into account.

Rules 1 and 2 are interworking criteria. Rule 3 is a performance criterion that can be verified on the level of System Management through access to the maximum turnaround time value and the resulting T0 timer.

NOTE The choice of minimum and maximum STT values can result in different interoperability and performance classes.

29.6.3 Timer functions

Five timers are defined using T0 as a basis.

The T1, T4, and T6 timers correspond to the use of T0 in the specific context of the status of a DLE (Station or Bus Arbitrator). Activation and deactivation conditions are thus identical (SILENCE or BUSY), and the action after expiration of the timers differs according to the status of the DLE.

The T3 and T5 timers also refer to T0: their lengths (which are different from T0) are calculated using T0 length as a basis. In addition, they are activated and deactivated under the same conditions (SILENCE or BUSY).

An additional T2 timer is also used. T2 is not based on T0. This timer determines the length of a basic synchronized cycle.

All of these timers are listed in Table 230

Table 230 – DL-Timers

Name of timer	Location	Function
<u>T1</u>	B.A.	monitors the absence of RPxx after IDxx
T2	B.A.	monitors the filling of the synchronization window by identifiers from the basic padding sequence
<u>T3</u>	B.A.	monitors the absence of IDxx after RPxx switching of Bus Arbitrators
<u>T4</u>	consuming station	monitors the absence of RP_DATxx after ID_DAT
<u>T5</u>	B.A.	monitors the lack of RP_END after ID_MSG
<u>T6</u>	message source station	monitors the absence of RP_ACK after RP_MSG_ACK

Timer definitions refer to names of statuses mentioned in the protocol descriptions presented in clause 31.

29.6.3.1 Timer T1

- Is located in the active Bus Arbitrator
- Its purpose is to monitor the emission of a variable response DLPDU or a request response DLPDU within a given time.
- Is active when the Bus Arbitrator is waiting for a variable response DLPDU or a request response DLPDU (BA_WAITING_DAT or BA_WAITING_RQ status), after its emission of a variable identifier or request identifier DLPDU.
- The length of timer T1 is equal to the length of timer T0.
- Expiration of the timer gives the Bus Arbitrator a new status: that of emitting the next identifier DLPDU (NEXT status).

29.6.3.2 Timer T2

- Is located in the active Bus Arbitrator
- Its purpose is to monitor the length of the synchronization window of the basic cycle. The active Bus Arbitrator emits the identifier DLPDUs of the basic padding sequence as long as the T2 timer has not expired.
- Is activated at the beginning of each basic synchronized cycle, when the Bus Arbitrator takes on the status of emitting identifier DLPDUs (NEXT status after BA_WAITING_SYNC status).
- Is not deactivated by any event.
- The length of the synchronization window is a function of the basic cycle's aperiodic exchanges or message services;
- Expiration of the timer gives the Bus Arbitrator a new status: that of emitting the next identifier DLPDU (NEXT status).

29.6.3.3 Timer T3

- Is located in potential Bus Arbitrators

- Its purpose is to monitor the active Bus Arbitrator's emission of identifier DLPDUs within a given time.
- Is active as long as the Bus Arbitrator has potentially active status.
- The length of the T3 timer should be superior to the length of the T0 timer. It should have a different value for each of the potential Bus Arbitrators. The note below suggests a value for this length.
- Expiration of the timer triggers the activation of a Bus Arbitrator.

NOTE T3 timer dimensions are set as follows:

$T3 = k \times n \times T0$ with

$n > 2$ a function of the geographic address of the Bus Arbitrator to which T3 is attached and
 k a coefficient that allows the Bus Arbitrator's turnaround time to be covered.

29.6.3.4 Timer T4

- Is located in stations
- Its purpose is to monitor the emission of a variable response DLPDU on the bus within a given time, in order for example to correctly associate a DLCEP-identifier DLPDU received with the corresponding response DLPDU.
- Is active when the station is waiting for a variable response DLPDU (WAITING_DAT status).
- The length of T4 is equal to the length of T0
- Expiration of the timer causes the station to be placed in the status of waiting for the next identifier DLPDU (WAITING_ID status).

29.6.3.5 Timer T5

- Is located in the active Bus Arbitrator
- Its purpose is to monitor a message source station's emission of a response DLPDU indicating the end of a message transaction within a given time.
- Is active when the Bus Arbitrator is waiting for an end of message response DLPDU (BA_WAITING_END status), after its emission of a message identifier DLPDU.
- The length of timer T5 should be greater than the length of timer T0. The note below suggests a value for T5.
- Expiration of the timer gives the Bus Arbitrator a new status: that of emitting the next identifier DLPDU (NEXT status).

NOTE The length of timer T5 is equal to two times the length of timer T0.

With such a choice of lengths collision between two operators is avoided. For example, when an acknowledgement DLPDU RP_ACK is lost the Bus Arbitrator cannot emit a second ID_MSG at the same time as the source station answers with the emission of RP_MSG_ACK (see T6).

29.6.3.6 Timer T6

- Is located in stations
- Its purpose is to monitor the emission of an acknowledgement DLPDU by the destination station within a given time.
- Is active when the source station is waiting for an acknowledgement DLPDU (WAITING_ACK status) after the end of its emission of an acknowledged message response DLPDU.
- The length of T6 is equal to the length of T0
- Expiration of the timer causes the station to be placed in the status of re-emitting the message if possible (RESTART_SOURCE status) or else in the status of waiting for the next identifier DLPDU (WAITING_ID status) after emitting the end of message transaction DLPDU.

The definition of these timers is necessary to guarantee the ability to interoperate of the various devices connected to a DLL local link.

All the timers within a single local link should have the same value. A default value is defined. These values can be modified by a station's System Management as long as the equivalence of values is respected. Values will be agreed upon later in accordance with implementation needs.

30 Type 7: DLPDU-specific structure, encoding and element of procedure

30.1 General

This clause describes the structure, contents of the PDU and specifies elements of procedure related to the invoked service.

Behavior of particular cases is described in detail in the relative subclause by the use of dedicated state machines.

30.2 Buffer read

Buffer reading service is local to an entity since the interactions related to this service take place only through the DLS-user/DLL interface, as shown in Figure 162. Correspondence between DLSDU service data units and interactions between the DLL and the communication medium are provided by the buffer transfer service.

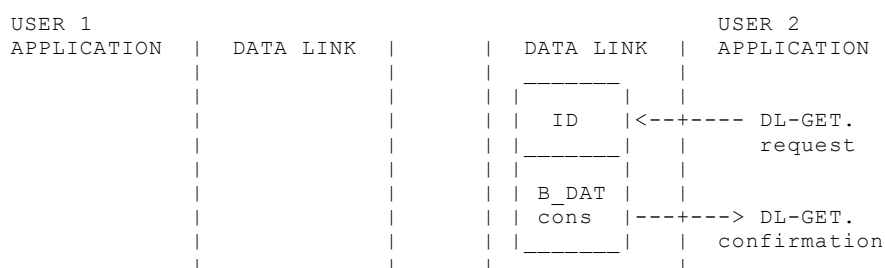


Figure 162 – Buffer reading service interactions

30.3 Buffer write

Buffer writing service is local to an entity since the interactions related to this service take place only through the DLS-user/DLL interface, as shown in Figure 163. Correspondence between DLSDU service data units and interactions between the DLL and the communication medium are provided by the buffer transfer service

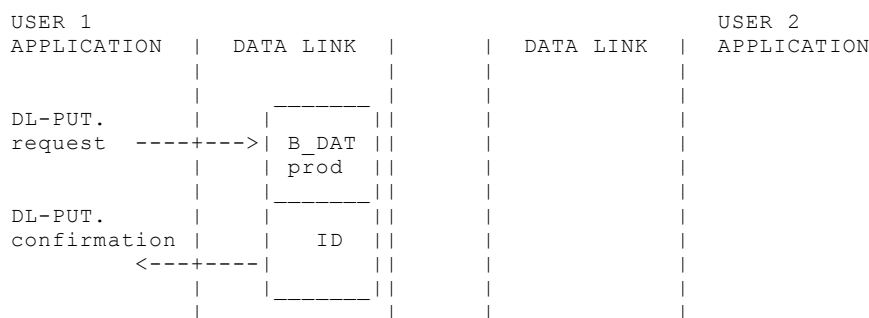


Figure 163 – Buffer writing service interactions

30.4 Buffer transfer

There are two phases to buffer transfer for all variables, as shown in Figure 164:

- the Bus Arbitrator emits the identifier DLPDU associated with the variable,
- the producing entity emits the corresponding response DLPDU and generates a DL-SENT indication primitive for the DLS-user. The value of the variable is picked up by all consuming entities and a DL-RECEIVED indication primitive is generated and passed on to each of the DLS-users concerned.

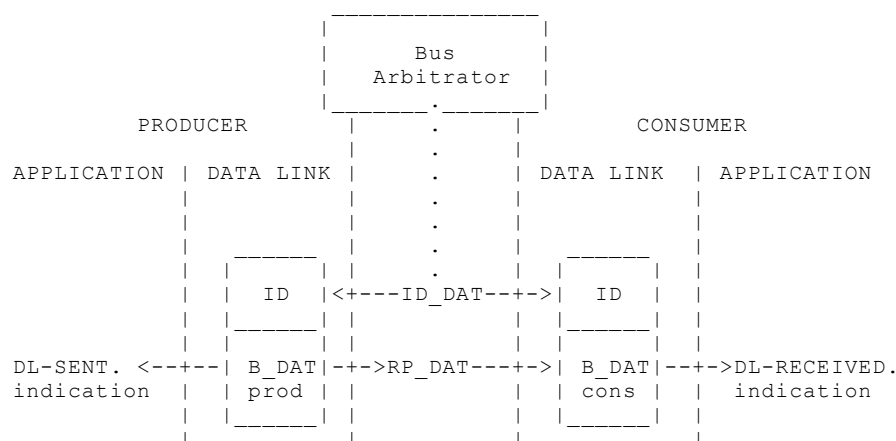


Figure 164 – Buffer transfer service interactions

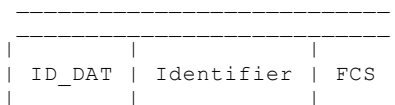
30.4.1 Associated DLPDUs

A buffer transfer involves the circulation of two DLPDUs.

The Bus Arbitrator broadcasts a DLCEP-identifier DLPDU (ID_DAT) that allocates the medium to the entity that produces the associated variable. The producing entity then broadcasts the variable in a response DLPDU (RP_DAT) that is picked up by the consuming entity or entities. The phases of this sequence are shown in Figure 165.

In the periodic window:

Medium allocation for an identified variable



Followed by a response containing the value of the variable

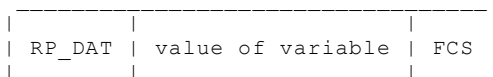


Figure 165 – Buffer transfer DLPDU sequence

NOTE A description of all DLPDUs, and the use of the various fields, can be found in chapter 28.5.

30.5 Specified explicit request

A specified explicit request for a variable exchange can be initiated by any entity. The request triggers exchanges whose form depends on the value of the indicator RQ_INHIBIT associated with SPECIFIED_IDENTIFIER. This value is set upon configuration.

If RQ_INHIBIT = FALSE, there are three phases to the exchange:

phase 1:

Following a DL-SPEC-UPDATE request primitive, the initiating entity uses the control field of the variable response DLPDU (corresponding to the identifier specified when the service was requested) to ask the Bus Arbitrator to circulate an urgent request identifier.

phase 2:

The Bus Arbitrator emits the corresponding request identifier in the window allocated to triggered variable exchanges (aperiodic window),

phase 3:

The initiating entity emits a request response DLPDU whose "data" field contains the list of identifiers that the entity would like broadcast. The initiating entity also transmits a DL-SPEC-UPDATE confirm primitive to the DLS-user. One or more buffer transfers then take place within the aperiodic traffic cycle.

- the Bus Arbitrator emits the identifiers of the variables requested,
- the producer of each variable requested emits a variable response DLPDU.

If RQ_INHIBIT = TRUE, there are two phases to the exchange:

phase 1:

Regardless of requests, for all identifiers set aside for specified service whose RQ_INHIBIT indicator is "true" the Bus Arbitrator emits a request identifier in the periodic window.

phase 2:

The initiating entity emits a request response DLPDU whose "data" field contains the list of variable identifiers that the entity would like broadcast. The initiating entity also transmits a DL-SPEC-UPDATE confirmation primitive to the DLS-user. One or more buffer transfers then take place within the periodic traffic cycle immediately following the request.

- the Bus Arbitrator emits the identifiers of the variables requested,
- the producer of each variable requested emits a variable response DLPDU.

The interaction sequences for the two classes of non-erroneous operation are shown in

Figure 166 and

Figure 167.

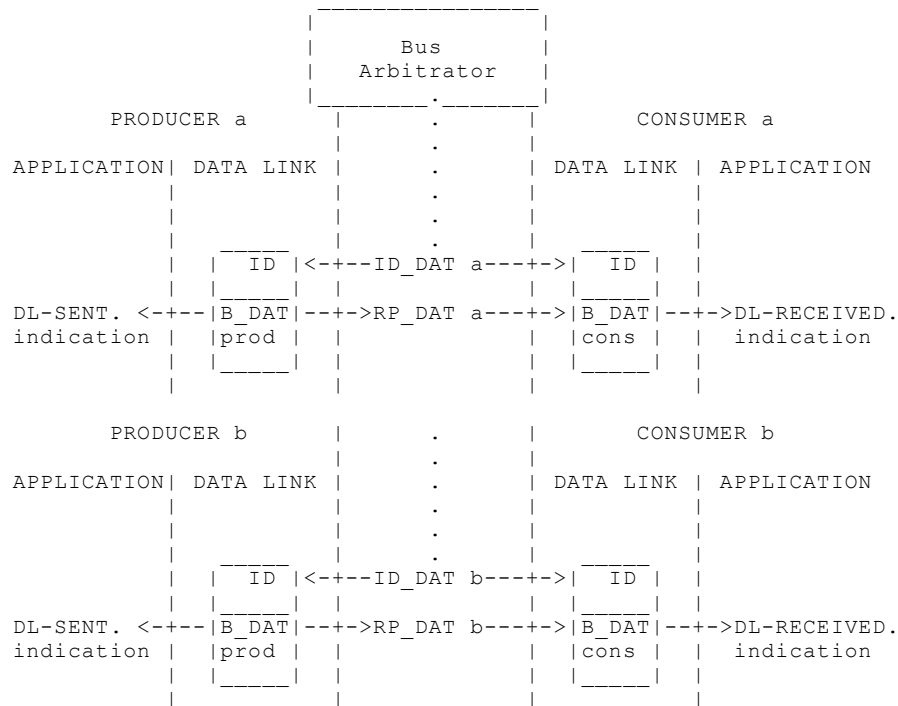
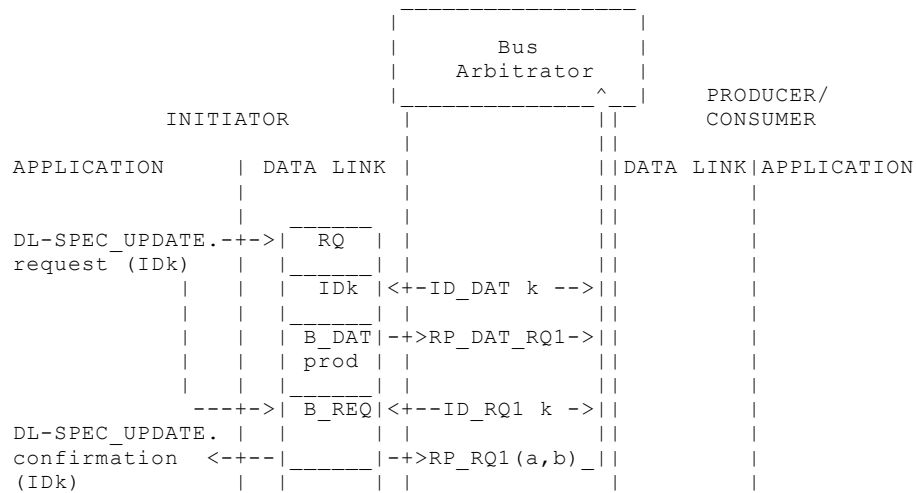


Figure 166 – Interactions within the specified explicit request for buffer transfer service in the aperiodic window

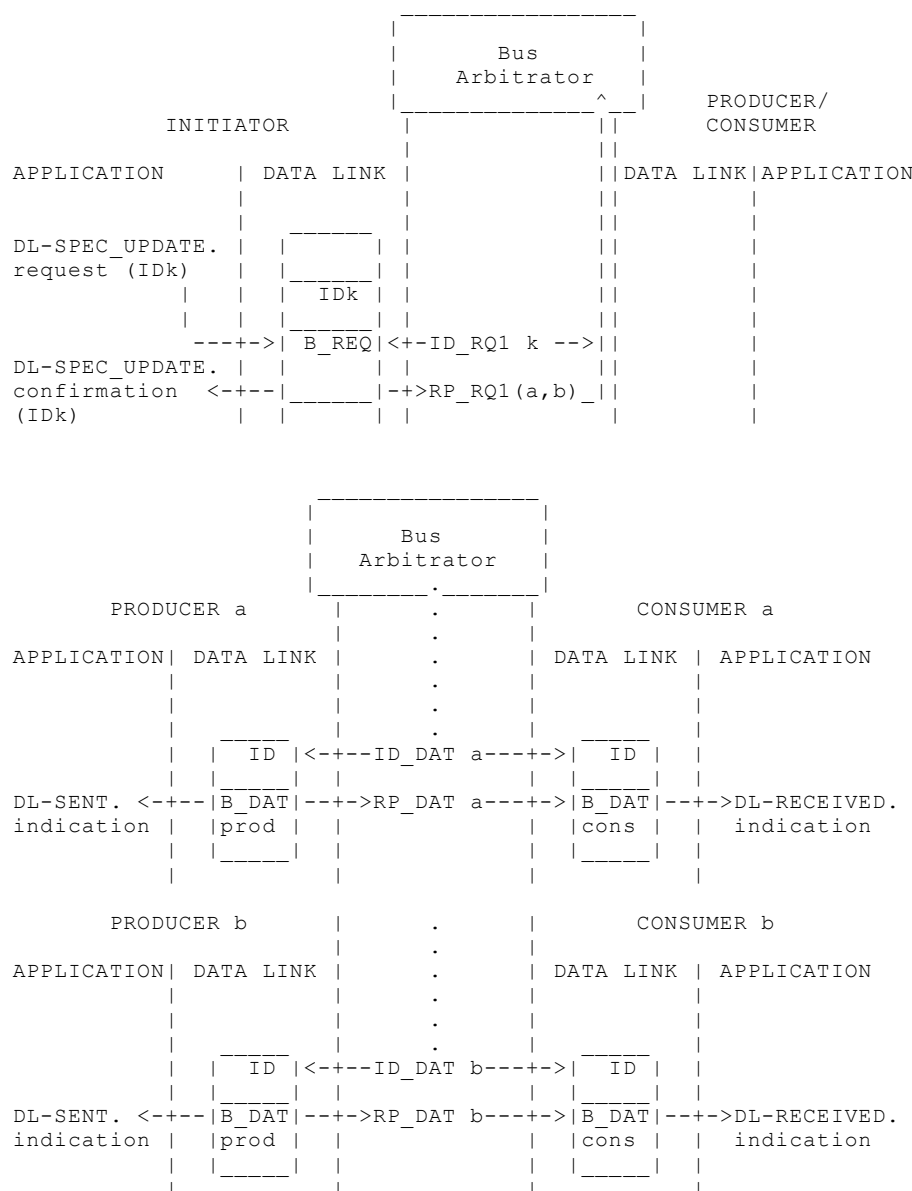


Figure 167 – Interactions within the specified explicit request for buffer transfer service in the periodic window

30.5.1 Associated DLPDUs

An explicit request for a buffer transfer corresponds to the circulation of three DLPDUs. The first of these DLPDUs is part of normal periodic scanning.

The initiating entity broadcasts a variable exchange response DLPDU whose control field includes the request to the Bus Arbitrator to circulate a request identifier. The initiating entity states the priority level (RP_DAT_RQi, or RP_DAT_RQi_MSG if an aperiodic message service is also requested with this identifier).

The Bus Arbitrator then broadcasts a request identifier DLPDU (ID_RQi) during an aperiodic window. The Bus Arbitrator uses the identifier which carried the request.

The list of variable identifiers to be broadcast is sent to the Bus Arbitrator (RP_RQi).

The Bus Arbitrator responds with one or more buffer transfers: (ID_DAT, RP_DAT) transaction.

If the specified explicit request is part of the Bus Arbitrator's periodic scanning cycle, the circulation of the first DLPDU is null and the other DLPDUs are exchanged during aperiodic scanning.

Figure 168 shows the DLPDU sequence for an explicit request for a transfer:

In the periodic window:

Medium allocation for an identified variable

ID_DAT	Identifier	FCS

followed by a response with an explicit request for a transfer, and the transfer's priority:

RP_DAT_RQi	value of the variable	FCS

or

RP_DAT_RQi_MSG	value of the variable	FCS

and then a number of identifiers later, in the aperiodic window:

**Medium allocation for an identified variable
and a response list of variable identifiers**

ID_RQi	Identifier	FCS

followed by a list of the identifiers required

RP_RQi	list of identifiers	FCS

followed by the transactions at a later time during the aperiodic window:

- medium allocation for identified variables (ID_DAT)
- associated responses (RP_DAT).

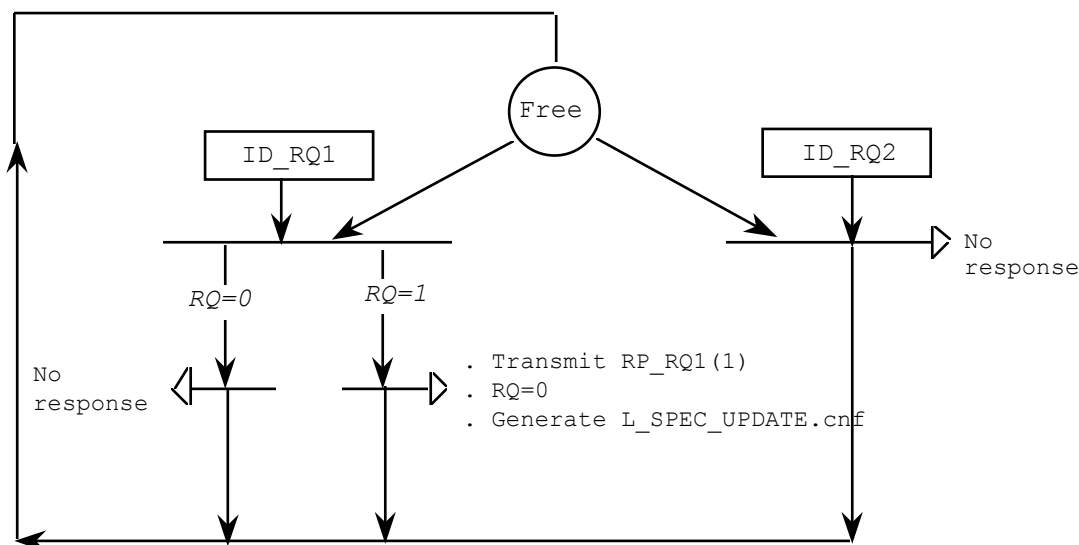
Figure 168 – DLPDU sequence for an explicit request for a transfer

30.5.2 Behavior with mismatched parameters

30.5.2.1 Buffer transfer specified explicit request (RQ_INHIBITED = False)

The evaluation network of Figure 169 allows specifying the behavior of a producer/consumer entity in the following cases:

- reception of an ID_RQ2 DLPDU containing a DLCEP-identifier bearing a buffer transfer exchange specified explicit request (RQ=1),
- reception of an ID_RQ1 DLPDU containing a DLCEP-identifier not bearing a buffer transfer specified explicit request (RQ=0).



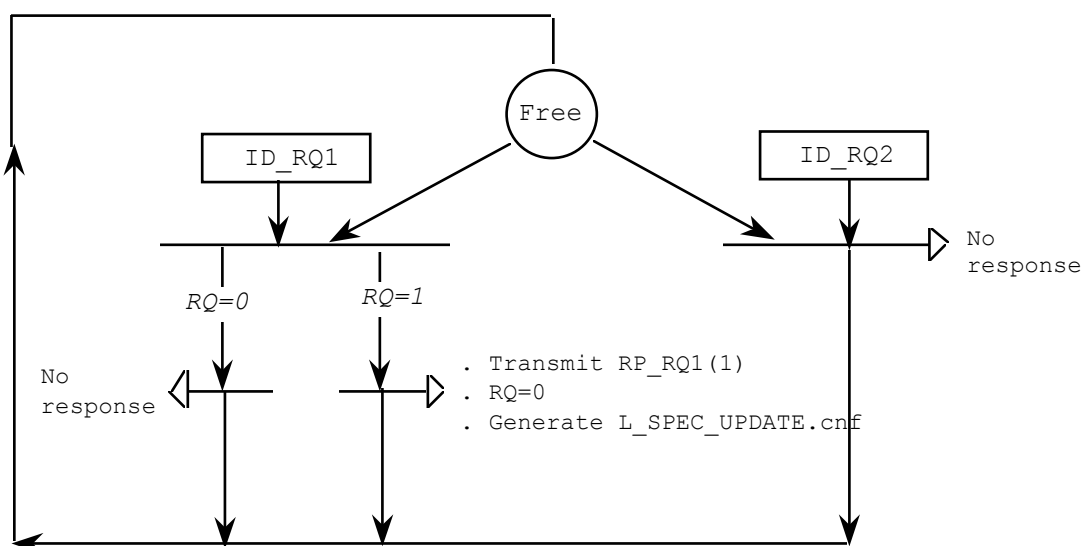
NOTE 1 When RQ=1 and B_REQ contains an empty list, RP_RQ1 is transmitted with an empty list.

Figure 169 – Evaluation network for a buffer transfer specified explicit request with (RQ_INHIBITED = False)

30.5.2.2 Buffer transfer specified explicit request (RQ_INHIBITED = True)

The evaluation network of Figure 170 allows specifying the behavior of a producer/consumer entity in the following cases:

- reception of an ID_RQ2 DLPDU containing a DLCEP-identifier configured for the service of the buffer transfer specified explicit request such as RQ_INHIBITED=TRUE,
- reception of an ID_RQ1 DLPDU containing a DLCEP-identifier configured for the service of the buffer transfer specified explicit request such as RQ_INHIBITED=TRUE and whose content has already been transmitted on the bus.



NOTE 1 When RQ=1 and B_REQ contains an empty list, RP_RQ1 is transmitted with an empty list.

Figure 170 – Evaluation network for a buffer transfer specified explicit request with (RQ_INHIBITED = True)

30.6 Free explicit request

A free explicit request for the exchange of an identified variable can be initiated by any entity and takes place in three phases.

phase 1:

Following a DL-FREE-UPDATE request primitive, the initiating entity uses the control field of the variable response DLPDU to ask the Bus Arbitrator to circulate a request identifier. The entity indicates priority.

phase 2:

The Bus Arbitrator emits the corresponding request identifier in the window allocated to triggered variable exchanges (aperiodic window),

phase 3:

The initiating entity emits a request response DLPDU whose "data" field contains the list of variable identifiers that the entity would like broadcast. The initiating entity also transmits a DL-FREE-UPDATE confirmation primitive to the DLS-user.

One or more buffer transfers then take place within the aperiodic traffic cycle.

The interaction sequences for non-erroneous operation is shown in Figure 171.

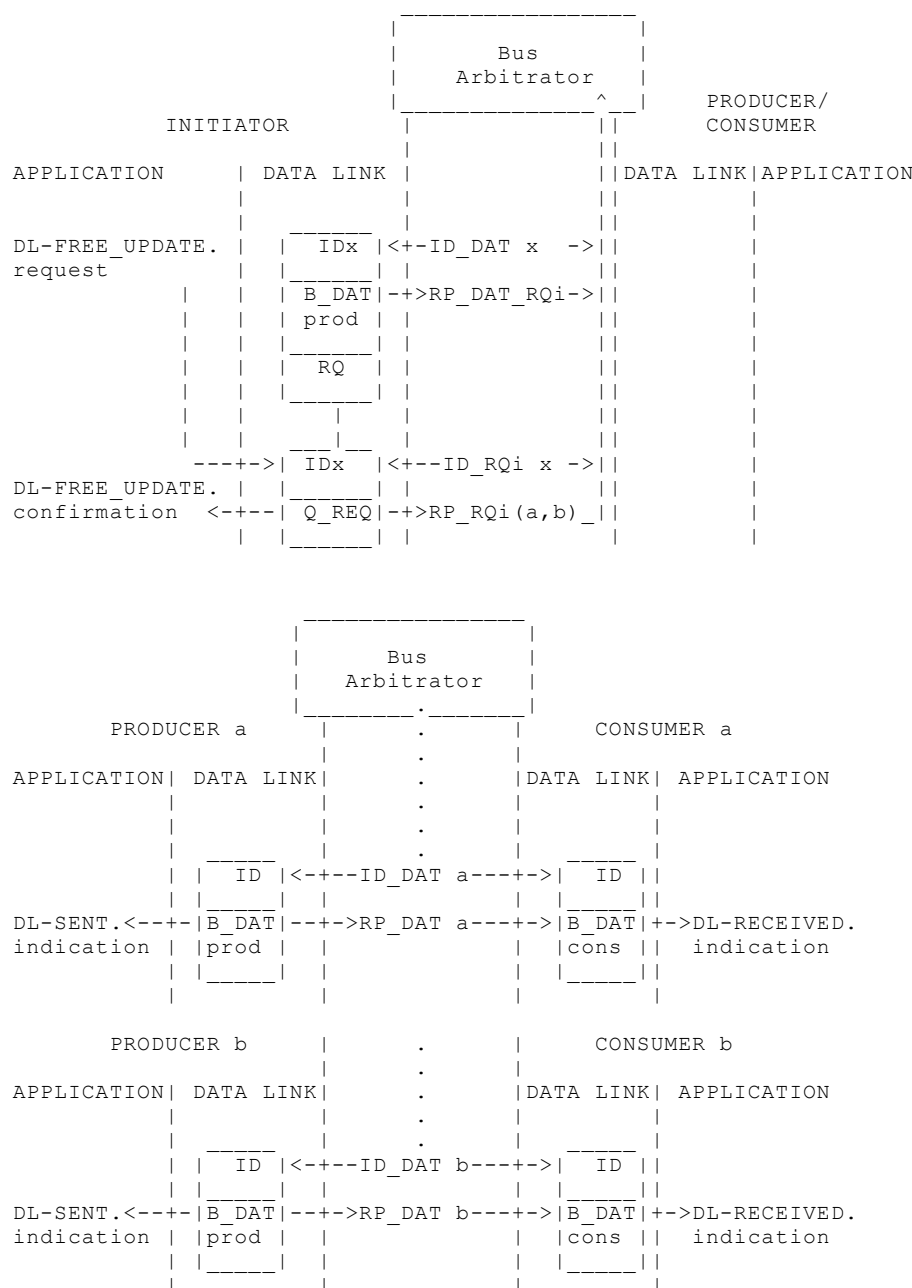


Figure 171 – Diagram of interactions within the free explicit request for buffer transfer service

30.6.1 Associated DLPDUs

A free explicit request for a buffer transfer corresponds to the circulation of three DLPDUs, as shown in 30.5.1. The first of these DLPDUs is part of normal periodic scanning.

The initiating entity broadcasts a variable exchange response DLPDU whose control field includes the request to the Bus Arbitrator to circulate a request identifier. The initiating entity states the priority level (RP_DAT_RQi, or RP_DAT_RQi_MSG if an aperiodic message service is also requested with this identifier).

The Bus Arbitrator then broadcasts a request identifier DLPDU (ID_RQi) during an aperiodic window. The Bus Arbitrator uses the identifier, which carried the request.

The list of variable identifiers to be broadcast is sent to the Bus Arbitrator (RP_RQi).

The Bus Arbitrator responds with one or more buffer transfers: (ID_DAT, RP_DAT) transaction.

If the specified explicit request is part of the Bus Arbitrator's periodic scanning cycle, the circulation of the first DLPDU is null and the other DLPDUs are exchanged during period scanning.

30.6.2 Behavior with mismatched parameters

The evaluation network of Figure 172 allows specifying the management of a producer/consumer entity in the following cases:

- Reception of an ID_RQ1 or ID_RQ2 DLPDU containing a DLCEP-identifier configured for service of the buffer transfer free explicit request and not associated with Q_REQ1 or Q_REQ2 (RQ of the identifier has a value of 0).
- Reception of an ID_RQ1 or ID_RQ2 DLPDU containing a DLCEP-identifier associated with an empty Q_REQ1 or Q_REQ2 queue.

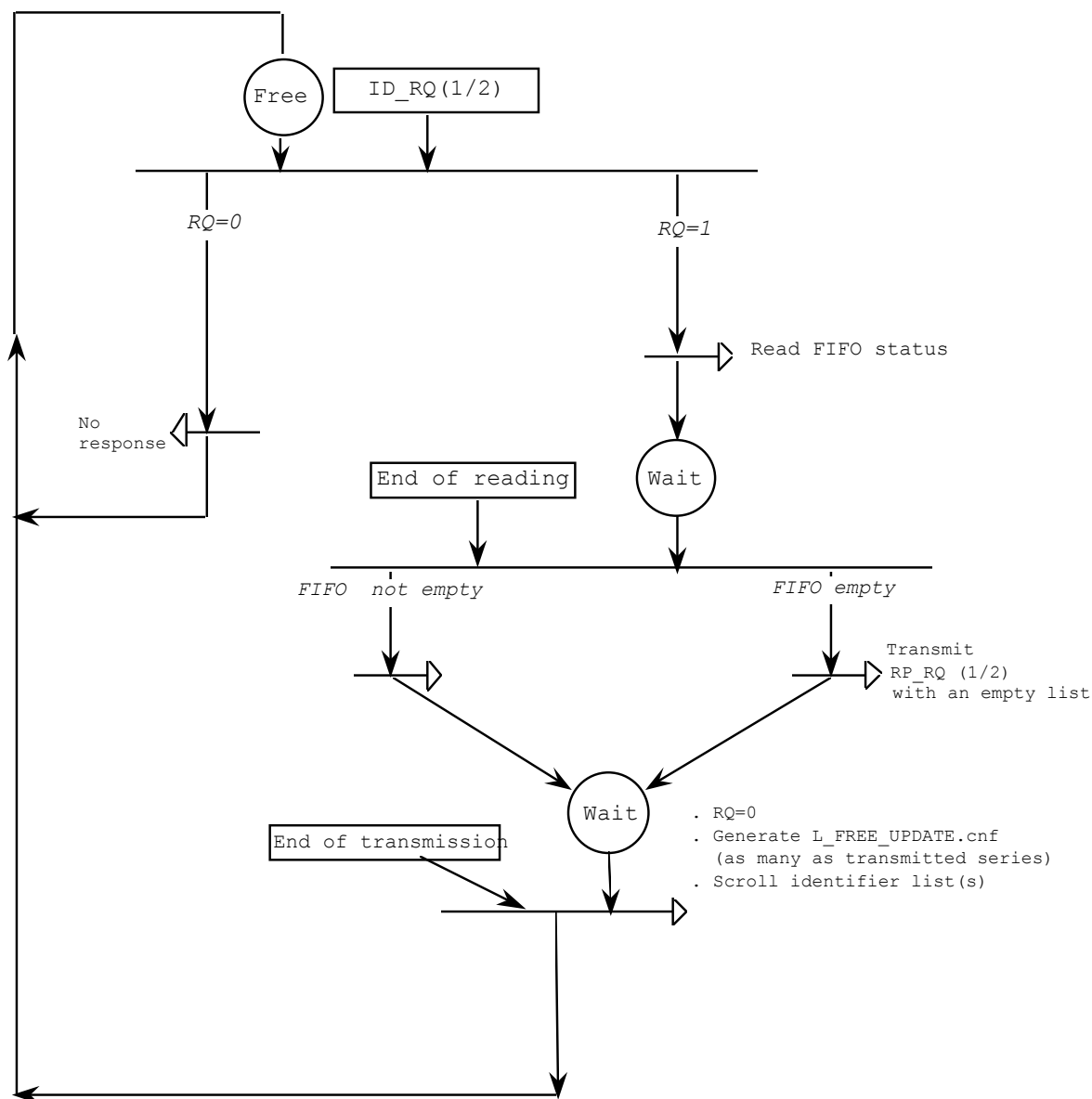


Figure 172 – Evaluation network for a free explicit request

After transmission of the RP_RQ, the request is detached from the separator concerned (RQ set to 0); If the FIFO is not empty, the request will be attached during the next reception of an ID_DAT concerning a DLCEP-identifier configured for this service. The RQ bit of this identifier will be set to 1.

30.7 Messaging

30.7.1 General introduction to message services

Message services make it possible to exchange acknowledged or unacknowledged messages.

The initiator of the message exchange request is the DLS-user that is the source of the message. The request is filled either during aperiodic scanning or during the Bus Arbitrator's periodic scanning cycle, depending on configuration.

If the request is filled during aperiodic scanning the message is deemed aperiodic. A single resource is set aside for this type of message: a queue that is dynamically linked to one or more identifiers configured for this mode of message transfer. The variable response DLPDUs associated with these identifiers carry the message transfer requests. Requests are filled in the aperiodic window.

If the request is filled during periodic scanning it is deemed cyclical, and the message transfer request takes place in the periodic window by configuration. Resources are allocated upon configuration and do not change. The resource set aside is a queue attached to a DLCEP-identifier configured for this mode of transfer. The message identifier DLPDU associated with this identifier is transmitted in the periodic window.

A single identifier cannot be configured for both message transfer modes.

Service primitives for message transfer requests are identical for both aperiodic and cyclical modes. Only one parameter of the request primitive indicates the message transfer mode requested and makes it possible to call upon the proper mechanism within the source DLE.

To put the communicating entities into contact two addresses are given during the message transaction:

- a destination address (access point for a message service) made up of 24 bits that encode the local link number of the entity and the entity's sub-address within that DL-segment.
- a source address (access point for a message service) made up of 24 bits that encode the local link number of the entity and the entity's sub-address within that DL-segment.

An access point for a message service addresses an Application message service entity for emission (source) and for reception (destination). Each address is unique within the system.

The format of the destination address and the source address is described in detail in an appendix.

Each entity detects the message response DLPDU (RP_MSG_NOACK or RP_MSG_ACK) and verifies its destination address to determine whether or not it is the destination of the message.

A recovery mechanism intervenes upon detection of the loss of an acknowledgement DLPDU linked to an acknowledged message transfer.

This mechanism for recovery upon loss of acknowledgement, and the restart counters, are found within the source entity. The number of authorized restarts is the same for all source entities within the system, and this number can be set upon configuration. The recovery mechanism can lead to the duplication of messages. An algorithm for numbering messages between the source entity and the destination entity is used to detect message duplication caused by the recovery mechanism (see 28.4.3).

30.7.2 Diagram of interactions

Calling upon the unacknowledged message transfer request service causes the addition of a message to the queue for messages to be emitted. This queue is designated by the parameter SPECIFIED_IDENTIFIER. How the exchange is carried out depends on this queue's type.

30.7.2.1 Aperiodic transfers

The queue is set aside for aperiodic transfer. The transfer includes four phases:

phase 1:

Upon reception of a DLCEP-identifier that can support an aperiodic request, and that is not already in use, the source entity indicates in the control field of the variable response DLPDU its request that the Bus Arbitrator circulate a message transfer DLPDU, provided that the number of occupied places in the queue is inferior to the number of identifiers associated with this resource.

In addition, for all identifiers already in use the source entity maintains the indication of a message transfer request in the variable response DLPDU.

phase 2:

The Bus Arbitrator gives control of the local link to the source entity by emitting a message DLSAP-address in the window allocated to messages.

phase 3:

The source entity emits the first message stored in the queue of messages to be emitted. The entity emits the message in a message response DLPDU whose extended control DLPDU contains the destination address and source address of the message.

phase 4:

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE confirm primitive is sent to the DLS-user (This primitive's SPECIFIED_IDENTIFIER parameter has the value NIL).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns control of the local link to the Bus Arbitrator immediately, and phase 3 is eliminated.

These phases are shown diagrammatically in Figure 173.

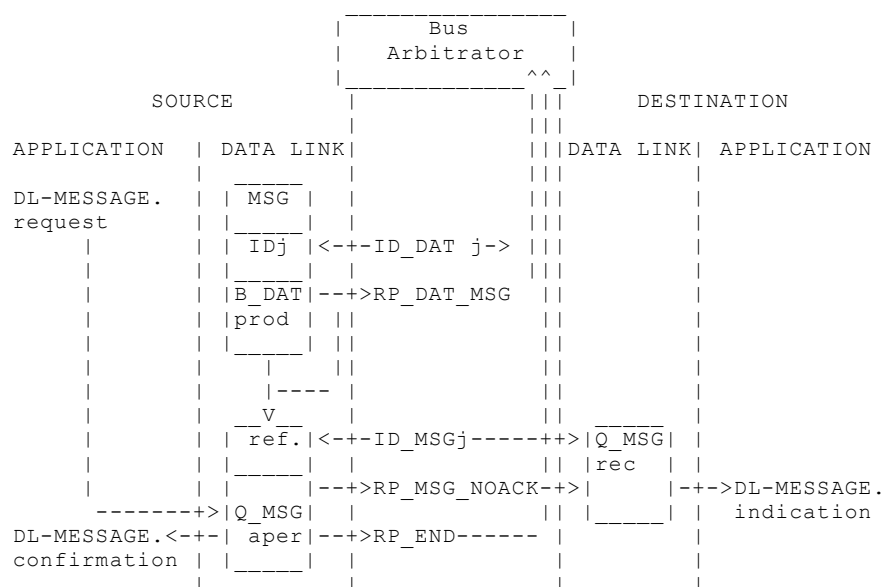


Figure 173 – Diagram of interactions within the unacknowledged message transfer request service for an aperiodic transfer

30.7.2.2 Cyclical transfers

The resource available for reception is a queue known as Q_MSGrec. The queue is set aside for cyclical transfers.

There are three phases to transfers:

phase 1:

The Bus Arbitrator gives the source entity control of the local link by emitting a message identifier DLPDU in the periodic window,

phase 2:

The source entity emits the message response DLPDU whose extended control field contains the message's destination and source addresses.

phase 3

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE confirm primitive is sent to the DLS-user (This primitive's SPECIFIED_IDENTIFIER parameter corresponds to the identifier in the message identifier DLPDU).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns local link control to the Bus Arbitrator immediately, and phase 2 is eliminated.

The resource available for reception is a queue known as Q_MSGrec.

These phases are shown diagrammatically in Figure 174.

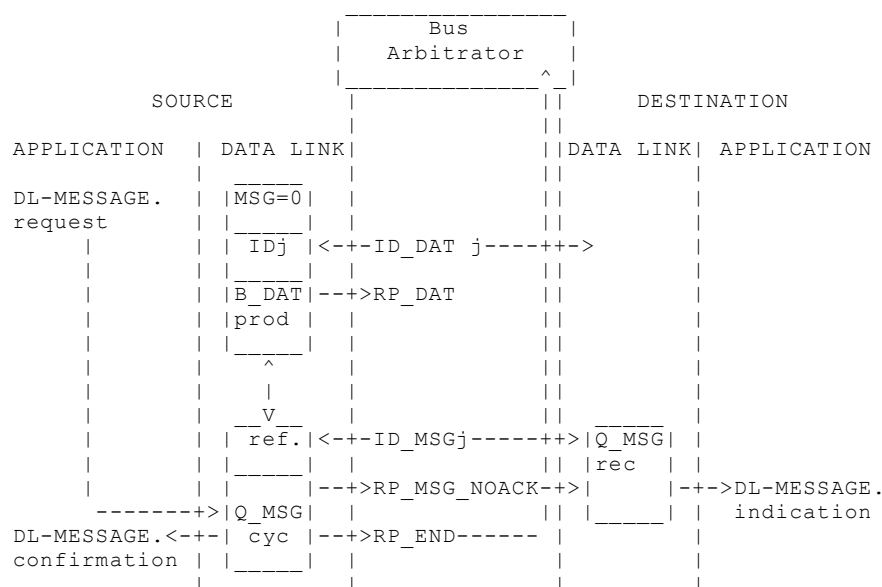


Figure 174 – Diagram of interactions within the unacknowledged message transfer request service for a cyclical transfer

30.7.3 Associated DLPDUs

An unacknowledged message transfer request involves the circulation of four DLPDUs in the case of an aperiodic message transfer. In the case of a cyclical message transfer, the first DLPDU is not broadcast.

30.7.3.1 Aperiodic message transfer

In the case of an aperiodic message transfer the source entity broadcasts a variable response DLPDU whose control DLPDU contains the request that the Bus Arbitrator circulate a message identifier DLPDU (RP_DAT_MSG or RP_DAT_RQi_MSG). The first of these DLPDUs is part of normal cyclical traffic.

The identifier refers to the queue for messages to be emitted that is set aside for aperiodic message transfer.

The Bus Arbitrator gives local link control to the source entity during an aperiodic message window by broadcasting a message identifier DLPDU (ID_MSG) with a DLCEP-identifier that has carried a request.

The message is emitted by the source entity (RP_MSG_NOACK) which states in the extended control field the message destination and source addresses.

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

On reception of a PDU containing a destination address which is homogeneous with a group address, layer 2 consults the local directory (by specific functions) in order to have the cross-reference between this group address and one or more local individual addresses to which is sent the DL-MESSAGE indication(Ad,As,m) indication primitive with the group address contained in the PDU as destination address.

Figure 175 shows the DLPDU sequence as it occurs in the periodic and aperiodic windows:

In the periodic window:

Medium allocation for identified variables

ID_DAT	Identifier	FCS

followed by a response plus message request

RP_DAT_MSG	value of the variable	FCS

or

RP_DAT_RQi_MSG	value of the variable	FCS

and then a number of identifiers later, in the aperiodic window:

Medium allocation for a message

ID_MSG	Identifier	FCS

followed by an unacknowledged message + destination address + source address

RP_MSG_NOACK	destination	source	message	FCS

after which the source entity returns control to the Bus Arbitrator

RP_END	FCS

Figure 175 – DLPDU sequence for an aperiodic message transfer

30.7.3.2 Cyclical message transfer

In the case of a cyclical message transfer, the Bus Arbitrator gives control to the source entity by broadcasting a message identifier DLPDU (ID_MSG) in the periodic window.

The message is emitted by the source entity (RP_MSG_NOACK) which notes in the extended control field the destination and source addresses.

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

On reception of a PDU containing a destination address which is homogeneous with a group address, layer 2 consults the local directory (by specific functions) in order to have the cross-reference between this group address and one or more local individual addresses to which is sent the DL-MESSAGE indication(Ad,As,m) indication primitive with the group address contained in the PDU as destination address.

Medium allocation for an identified variable is shown in Figure 176.

In the periodic window:

Medium allocation for a message

ID_MSG	Identifier	FCS

followed by an unacknowledged message + destination address + source address

RP_MSG_NOACK	destination	source	message	FCS

the source entity returns control to the Bus Arbitrator

RP_END	FCS

Figure 176 – DLPDU sequence for a cyclical message transfer

30.8 Acknowledged messaging

30.8.1 Diagram of interactions

Calling upon the acknowledged message transfer request service (DL-MESSAGE-ACK request) causes the addition of a message to the queue of messages to be emitted. This queue is designated by the parameter SPECIFIED_IDENTIFIER. The manner in which exchanges are carried out depends on the type of queue.

30.8.1.1 Queue dedicated to aperiodic transfer

If the queue is set aside for aperiodic transfer the exchange includes five phases:

phase 1:

Upon reception of a DLCEP-identifier that can support an aperiodic request, and that is not already in use, the source entity indicates in the control field of the variable response DLPDU its request that the Bus Arbitrator circulate a message transfer DLPDU, provided that the number of occupied places in the queue is inferior to the number of identifiers associated with this resource.

In addition, for all identifiers already in use the source entity maintains the indication of a message transfer request in the variable response DLPDU.

phase 2:

The Bus Arbitrator gives control to the source entity by emitting a message DLSAP-address in the window allocated to messages.

phase 3:

The source entity emits the first message stored in the queue of messages to be emitted. The entity emits the message in a message response DLPDU whose extended control DLPDU contains the destination address and source address of the message.

phase 4:

The destination entity emits the acknowledgement DLPDU immediately, except if the address destination is a group address. In this case, the entity ignores the DLPDU.

phase 5:

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE-ACK confirm primitive is sent to the DLS-user (This primitive's SPECIFIED_IDENTIFIER parameter has the value NIL).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns local link control to the Bus Arbitrator immediately, and phases 3 and 4 are eliminated.

The resource available for reception is a queue known as Q MSGrec.

These phases are shown diagrammatically in Figure 177.

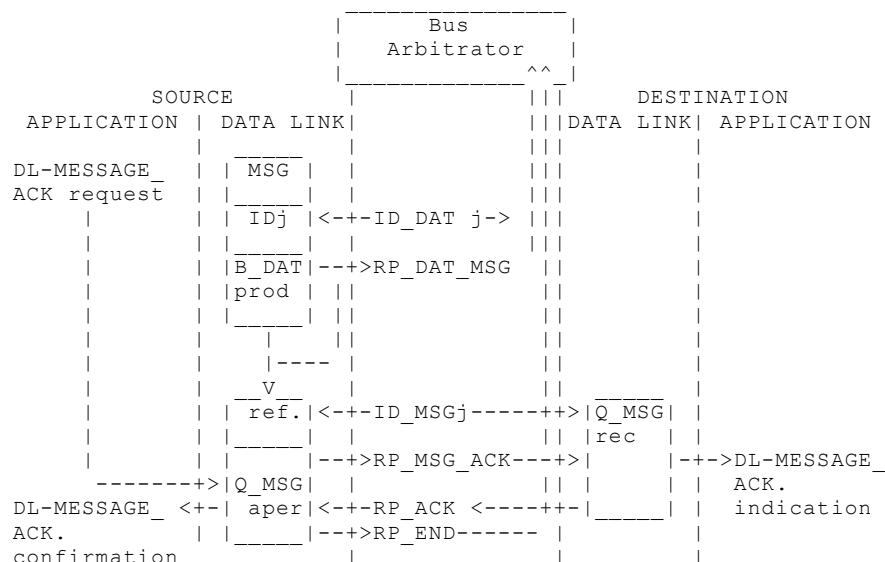


Figure 177 – Diagram of interactions within the acknowledged message transfer request service for an aperiodic transfer

30.8.1.2 Queue dedicated to cyclical transfers

The queue is set aside for cyclical transfers. There are four phases to transfers:

phase 1:

The Bus Arbitrator gives the source entity control of the local link by emitting a message identifier DLPDU in the periodic window.

phase 2:

The source entity emits the message response DLPDU whose extended control field contains the message's destination and source addresses.

phase 3:

The destination entity emits the acknowledgement DLPDU immediately, except if the address destination is a group address. In this case, the entity ignores the DLPDU.

phase 4:

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signifies the end of the message transaction. A DL-MESSAGE-ACK confirm primitive is sent to the DLS-user. (This primitive's SPECIFIED_IDENTIFIER parameter corresponds to the identifier in the message identifier DLPDU).

NOTE If the source entity has no message associated with the message identifier DLPDU being circulated (empty queue) it returns local link control to the Bus Arbitrator immediately, and phases 2 and 3 are eliminated.

The resource available for reception is a queue known as Q MSGrec.

These phases are shown diagrammatically in Figure 178.

In the periodic window:

Medium allocation for identified variables

ID_DAT	Identifier	FCS

followed by a response + message request

RP_DAT_MSG	value of the variable	FCS

or

RP_DAT_RQi_MSG	value of the variable	FCS

and then a number of identifiers later, in the aperiodic messages window:

Medium allocation for a message

ID_MSG	Identifier	FCS

followed by a message with request for acknowledgement + destination address + source address

RP_MSG_ACK	destination	source	message	FCS

followed by an acknowledgement DLPDU

RP_ACK	FCS

followed by the end of transaction signal

RP_END	FCS

Figure 179 – DLPDU sequence for an aperiodic message transfer

30.8.2.2 Cyclical message transfer

In the case of a cyclical message transfer, the Bus Arbitrator gives control to the source entity by broadcasting a message identifier DLPDU (ID_MSG) in the periodic window.

The message is emitted by the source entity (RP_MSG_ACK) which notes in the extended control field the destination and source addresses.

The destination entity emits the acknowledgement DLPDU (RP_ACK) immediately.

The source entity returns control to the Bus Arbitrator by emitting a DLPDU that signals the end of the message transaction (RP_END).

Figure 180 shows the DLPDU sequence as it occurs in the periodic window:

In the periodic window:

Medium allocation for a message

ID_MSG	Identifier	FCS

followed by a message with request for acknowledgement + destination address + source address

RP_MSG_ACK	destination	source	message	FCS

followed by the acknowledgement DLPDU

RP_ACK	FCS

followed by the end of transaction signal

RP_END	FCS

Figure 180 – DLPDU sequence for a cyclical message transfer

The RP_MSG_ACK and RP_ACK control fields containing the numbering bit Bp given by the algorithm defined in 30.9.

NOTE This DL-protocol provides SDA and SDN services as specified in IEC 60955 (PROWAY C). The DLL primitives also directly correspond to primitives specified in the documents ISO/IEC 8802-2 and ISO TC 97 SC6 WG1 N169. The DL-MESSAGE primitive corresponds to the DLSDU primitive. The DL-MESSAGE-ACK primitive corresponds to the DLSDU_ACK primitive.

30.9 Numbering of acknowledged messages

The algorithm describing the message numbering protocol uses the following data:

- each destination entity records the source address (ADRsource_stoc) of the last message response DLPDU received, and the associated even/odd bit (Bc),
- upon reception of any DLPDU other than RP_MSG_ACK, entities give the stored source address the value "empty",
- recovery is immediate (no other intermediate transaction). Recovery is provided by the source entity, which manages a waiting period for the acknowledgement DLPDU and a restart counter,
- the source entity sends the message with the indication Bp, and changes the value of this bit upon transmission of RP_END.

When the system is initialized each entity has the following information:

- ADRsource_stoc = "empty"
- Bp has no significance
- Bc has no significance

The algorithm has three parts. One part is implemented by the message source entity, another by the destination entity, and the third by the Bus Arbitrator.

30.9.1 Destination algorithm:

```

Upon reception of RP_MSG_ACK,
compare ADRsource received with ADR source_stoc
if equal compare Bp received with Bc
  if equal
    if there is room to store
      then send RP_ACK+Bp
      store message
      Bc=1-Bp
    else send RP_ACK-Bp
  else send RP_ACK+Bp
    ignore message because duplicate
else
  if there is room to store
    send RP_ACK+Bp
    Bc=1-Bp
    store the message
    ADRsource_stoc=ADR source received
  else send RP_ACK-Bp

if reception of DLPDU other than RP_MSG_ACK
  ADRsource_stoc=empty

```

30.9.2 Source algorithm:

```

Upon reception of ID_MSG(IDk) and IDk recognized by the source entity
send message with Bp indication
wait for RP_ACK

if reception RP_ACK
  set MSG of IDk at 0
  send confirmation with status (ack ±)
  reset restart counter to zero
  send RP_END and Bp=1-Bp

if expiration of waiting time for RP_ACK,
  test restart counter

  if counter > maximum number of restarts authorized
    set MSG of IDk at 0
    reset restart counter to zero
    send confirmation with negative status
    send RP_END and Bp=1-Bp

  if counter ≤ maximum number of restarts authorized
    retransmit same RP_MSG_ACK
    increment restart counter
    wait for RP_ACK

```

30.9.3 Bus Arbitrator algorithm:

After sending ID_MSG

The Bus Arbitrator is waiting for RP_END:

if the timer for waiting for RP_END has expired

go on to the next identifier

if reception of RP_END

go on to next identifier

if reception of DLPDU other than RP_END, or of an invalid DLPDU

continue to wait for RP_END

30.10 Behavior with mismatched parameters

30.10.1 Message aperiodic transfer (acknowledged or not)

The evaluation network of Figure 181 allows specifying the behavior of a producer/consumer entity in the following cases:

- Reception of an ID_MSG DLPDU containing a DLCEP-identifier configured for aperiodic transfer of messages and not associated with the Q_MSG.aper queue (the identifier MSG has a value of 0),
- Reception of an ID_MSG DLPDU containing a DLCEP-identifier configured for aperiodic transfer of messages and associated with the Q_MSG.aper queue (the identifier MSG has a value of 1), the latter being empty.

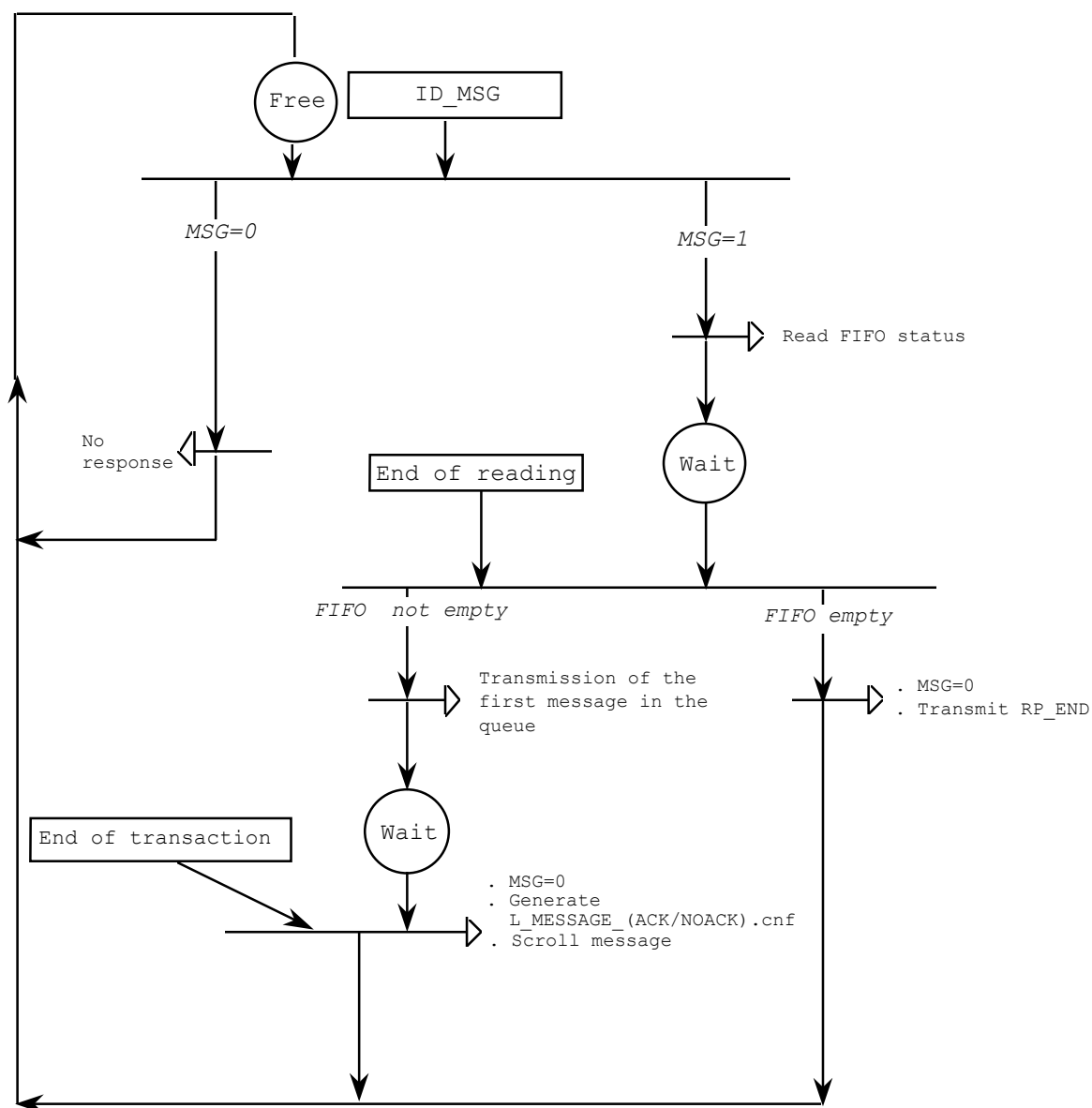


Figure 181 – Evaluation network for message aperiodic transfer

30.10.2 Message cyclic transfer (acknowledged or not)

The evaluation network of Figure 182 allows specifying the behavior of a producer/consumer entity, on reception of an ID_MSG DLPDU containing a DLCEP-identifier configured for the cyclic transfer of messages and whose Q_MSG.cyc queue is empty.

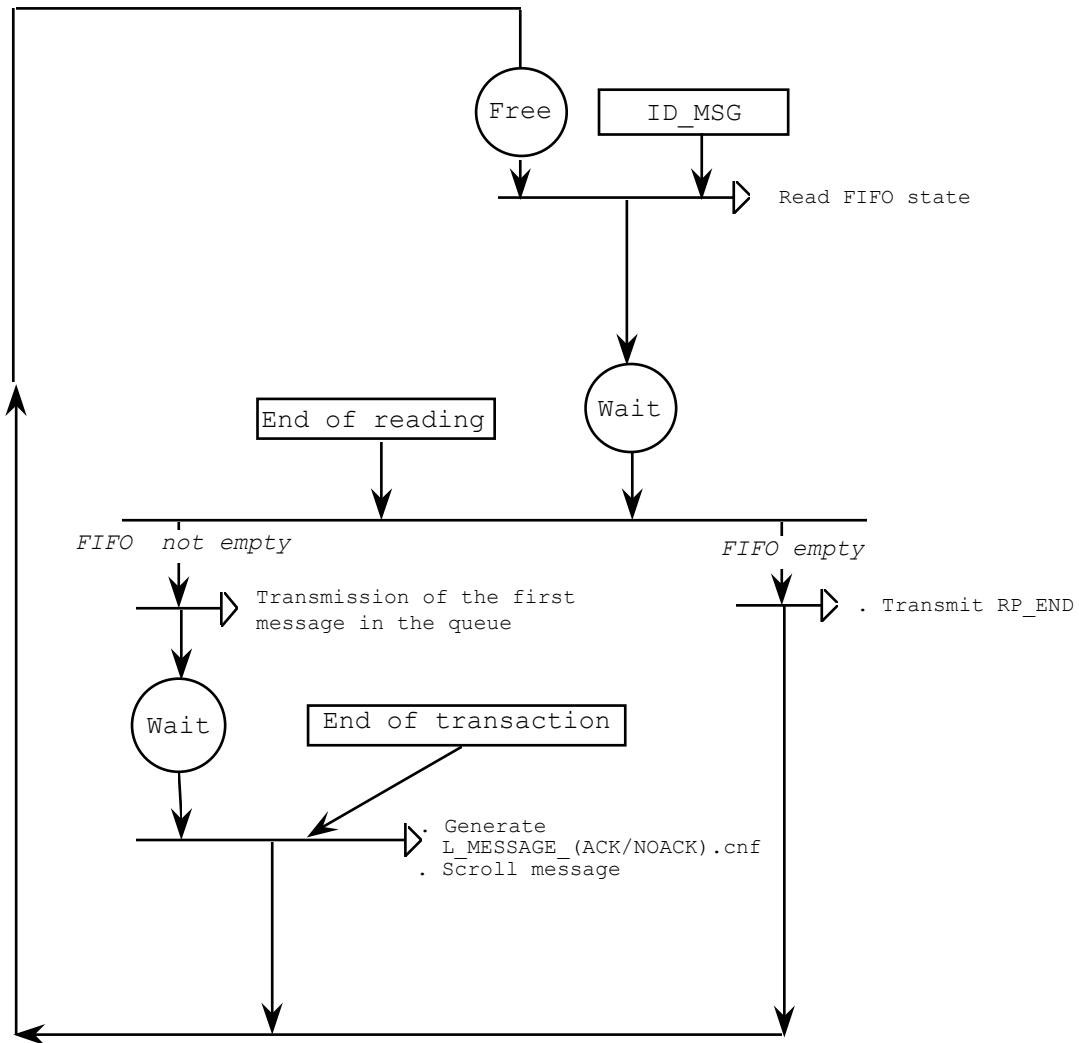


Figure 182 – Evaluation network for message cyclic transfer

31 Type 7: DL-service elements of procedure, interfaces and conformance

31.1 General

These finished status machines describe the overall functioning of a DLE as a function of the DLPDUs circulating on the medium, and they use the time-fillers defined above.

Transition tables complete the description of the functioning of a DLE by stating actions carried out upon reception of a DLPDU.

The protocols associated with each service provided by the DLL are described for a producer/consumer entity and for the Bus Arbitrator.

The interfaces to the DLS-user, DL- Management and the PhL are summarized and the Type 7 conformance classes are shown in Figure 183.

31.2 Producer/consumer entity

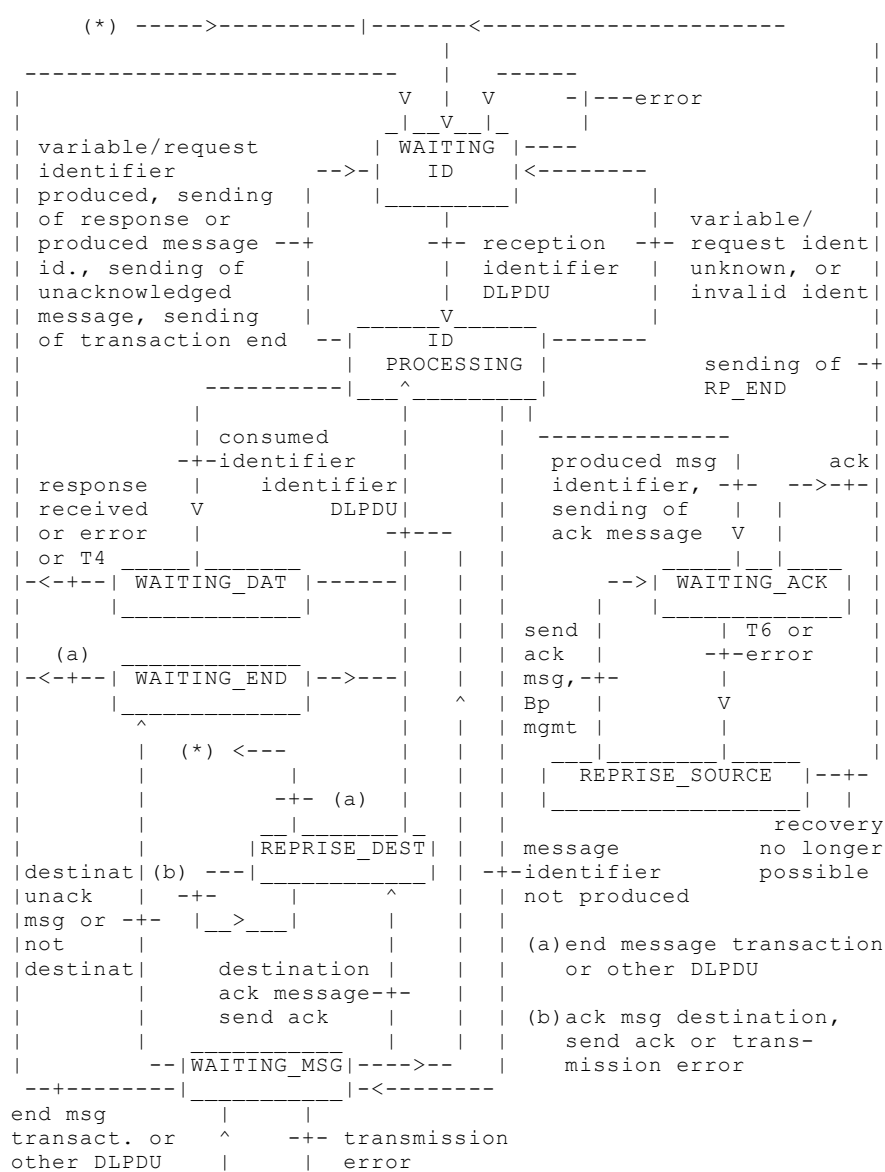


Figure 183 – Simplified states machine for a producer/consumer entity

The status machine indicates the behavior of a station during reception and transmission. This machine takes into account in its various statuses:

- reception of various types of link DLPDU,
- time-outs specific to the station (they are not those of the BA).

It is important to note that these various link DLPDUs taken into account in the various statuses of this machine can only be taken into account in so far as the latter at least observe the temporal characteristics of the protocol.

a) Temporal characteristics of the protocol without interruption.

In the case of normal sequencing of the protocol, the temporal characteristics of the latter are fixed by the turn-around time value during production of the various stations. Each DLPDU received by a device which must provide a response is separated from the latter on the bus by a time interval equal to its own turn-around time during production.

NOTE Knowing that the device which has the maximum turn-around time during production conditions the choice of the time-out T0.

b) Temporal characteristics of the protocol in case of interruption

If the DLPDU sequencing is degraded for whatever reason (disturbance on the bus, absence of production) the temporal characteristics of the protocol are fixed by the bus arbitrator.

Thus for example, an ID_XX DLPDU shall be taken into account in the various reception statuses only if the BA observes the following temporal constraints:

- In the WAIT_DAT status, the BA must wait at least $T1=T0$ before proceeding in the transmission of an ID_XX. Knowing that time-out T1 on the BA level is set by the line silence following transmission of a n ID_DAT DLPDU and that it can elapse only if silence is maintained all through its life.
- In the MSG_WAIT or END_WAIT or DEST_RESTART, the BA must attain at least $T5=2T0$, in case of non-reception of RP_END before proceeding with transmission of an ID_XX DLPDU. Knowing that time-out T5 at the BA level is set by each line silence following transmission of an ID_MSG DLPDU and that it can only elapse if the silence is maintained all through its life.

NOTE In the case of an arbitrator which has identified the cause of this interruption (for example, producer absent) and which would like to turn around immediately to pass to transmission of the following identifier, operation of the protocol is not guaranteed.

31.2.1 Transitions table for a producer/consumer entity

```

WAITING_ID (initial status)
  ID_xx:                                --> PROCESSING_ID

  other DLPDU:                          --> WAITING_ID
  transmission error:                   --> WAITING_ID

PROCESSING_ID
  ID_DAT,ID produced:
    if the identifier is invalid, no response                                --> WAITING_ID

    if indicator of urgent explicit request emit RP_DAT_RQ1
    if indicator of normal explicit request emit RP_DAT_RQ2
    if indicator of message request emit RP_DAT_MSG
    if 2 indicators emit RP_DAT_RQi_MSG
    else emit RP_DAT

    generate a DL-SENT indication (ID produced)                            --> WAITING_ID

  ID_DAT,ID consumed: activate T4                                           --> WAITING_DAT
  ID_DAT, new ID:                                                            --> WAITING_ID

```

```

ID_MSG, ID produced:
    if msg exists and is unack, emit RP_MSG_NOACK
                                emit RP_END                --> WAITING_ID

    if msg exists and is ack, emit RP_MSG_ACK
                                activate T6                --> WAITING_ACK

    if msg does not exist, emit RP_END                    --> WAITING_ID

ID_MSG, new ID:                                          --> WAITING_MSG

ID_RQ1, ID produced:
    emit RP_RQ1
    generate DL-SPEC_UPDATE confirm (ID produced, OK)
    or DL-FREE_UPDATE confirm (OK)                      --> WAITING_ID

ID_RQ1, new ID:                                          --> WAITING_ID

ID_RQ2, ID produced:
    emit RP_RQ2
    generate DL-FREE_UPDATE confirm (OK)                  --> WAITING_ID

ID_RQ2, new ID:                                          --> WAITING_ID

WAITING_DAT
    RP_DAT or RP_DAT_MSG or RP_DAT_RQi or RP_DAT_RQi_MSG:
        store value of the variable
        generate a DL-RECEIVED indication (ID consumed)
                                                                --> WAITING_ID

    ID_xx:                                                --> PROCESSING_ID

    other DLPDU:                                          --> WAITING_ID
    transmission error:                                  --> WAITING_ID
    T4 timer: indicate transaction aborted                --> WAITING_ID

WAITING_MSG
    RP_MSG_NOACK, my address is destination:
        receive message
        generate a DL-MESSAGE indication (ADRdestination, ADRsource,
        message)                                          --> WAITING_END

    RP_MSG_NOACK, other destination address:              --> WAITING_END

    RP_MSG_ACK, my address is destination:
        generate message numbering bit
        receive msg, send RP_ACK
        generate a DL-MESSAGE_ACK indication (ADRdestination, ADRsource,
        message)                                          --> RECOVERY_DEST

    RP_MSG_ACK, other destination address                --> WAITING_END

    RP_END: end of message transaction                    --> WAITING_ID

    ID_xx:                                                --> PROCESSING_ID

    other DLPDU:                                          --> WAITING_ID
    transmission error:                                  --> WAITING_MSG

WAITING_ACK
    RP_ACK:
        generate the message numbering bit (see 30.9)
        generate a DL-MESSAGE_ACK confirm
        (ADRdestination, ADRsource, status)
        emit RP_END                                        --> WAITING_ID

    T6 timer:                                             --> RECOVERY_SOURCE

    other DLPDU:                                          --> RECOVERY_SOURCE
    transmission error:                                  --> RECOVERY_SOURCE

```

```

WAITING_END
  RP_END:                                --> WAITING_ID

  ID_xx                                  --> PROCESSING_ID

  other DLPDU:                            --> WAITING_ID
  transmission error:                     --> WAITING_ID

RECOVERY_DESTINATION
  RP_END:                                --> WAITING_ID

  RP_MSG_ACK: management of Bp (see 30.9)
    send ack                              --> RECOVERY_DEST

  ID_xx                                  --> PROCESSING_ID

  other DLPDU:                            --> WAITING_ID
  transmission error:                     --> RECOVERY_DEST

RECOVERY_SOURCE
  if recovery possible
    emit same RP_MSG_ACK
    activate T6                            --> WAITING_ACK
  else send RP_END
    indicate transaction aborted           --> WAITING_ID

```

NOTE Types of errors are defined in 4.6.22 (DLL-counter class description) of EN 50170-7.

31.3 Protocol elements by service

NOTE The protocols associated with the services provided by a producer/consumer entity are specified by describing the entity's behavior for each primitive exchanged with the DLS-user.

31.3.1 Reception of a DL-PUT request

Upon reception of a DL-PUT request primitive (IDk, DLSDU) a producer triggers the following actions:

- * semantic testing (unknown identifier or incorrect length)
 - semantics incorrect
 - immediate transmission of DL-PUT confirm (IDk, unknown identifier or length of DLSDU incorrect)
 - semantics correct
- * test validity of identifier
 - identifier invalid
 - immediate transmission of DL-PUT confirm (IDk, invalid identifier)
 - identifier valid
- * test availability of buffer
 - buffer unavailable,
 - immediate transmission of DL-PUT confirm (IDk, buffer unavailable)
 - buffer available,
 - write the new value
 - immediate transmission of DL-PUT confirm (IDk, OK)

The various tests carried out are based on data concerning:

- recognition of identifier parameter
- the identifier is known if it has been declared as a produced identifier by the producer/consumer entity
- compatibility of length of DLSDU with the buffer B_DATprod (length less than 128 octets).

- validity of the identifier
- a DLCEP-identifier is valid if the Data Link configuration of the associated variable is valid.
- availability of buffer
- buffer unavailable if the same B_DATprod buffer is already being accessed by the local link. To manage this conflict the standard recommends giving the local link highest priority in all cases.

31.3.2 Reception of a DL-GET request

Upon reception of a DL-GET request primitive (IDk) a producer triggers the following actions:

- * test identifier IDk
 - identifier unknown
 immediate transmission of DL-GET confirm (IDk, identifier unknown)
 - identifier known
- * test availability of buffer
 - buffer unavailable,
 immediate transmission of DL-GET confirm (IDk, buffer unavailable)
 - buffer available
- * test validity of identifier
 - identifier invalid
 immediate transmission of DL-GET confirm (IDk, invalid identifier)
 - identifier valid
 immediate transmission of DL-GET confirm (IDk, DLSDU, reading OK).

The various tests carried out are based on data concerning:

- recognition of identifier parameter

the identifier is known if it has been declared as a consumed identifier by the producer/consumer entity
- availability of buffer

buffer unavailable if the same B_DATcons buffer is already being accessed by the local link. To manage this conflict the standard recommends giving the local link highest priority in all cases.
- validity of the identifier
- a DLCEP-identifier is valid if the Data Link configuration of the associated variable is valid.

31.3.3 Reception of a DL-SPEC-UPDATE request

Upon reception of a DL-SPEC-UPDATE request primitive (IDk, LIST) a producer triggers the following actions:

- * test IDk
 - IDk unknown or not configured for this service
 immediate transmission of DL-SPEC-UPDATE confirm (IDk, identifier unknown)
 - IDk known and configured for this service

- * test availability of B_REQ (IDk) buffer
 - buffer unavailable
 - immediate transmission of DL-SPEC-UPDATE confirm (IDk, buffer unavailable)
 - buffer available
- * test transmission of preceding request
 - request not transmitted (RQ of IDk=1)
 - . immediate transmission of DL-SPEC-UPDATE confirm (IDk, override) of the preceding request
 - . store list of identifiers (this list can be empty)
 - request transmitted (RQ of IDk=0)
 - . set request indicator (RQ of IDk=1)
 - . store list of identifiers
- * reception of ID_DAT (IDk) produced
 - * test RQ of IDk
 - RQ of IDk inhibited by configuration
 - emit only RP_DAT
 - RQ of IDk not inhibited by configuration
 - emit RP_DAT_RQ
- * reception of ID_RQ (IDk)
 - . emit RP_RQ with list of identifiers requested
 - . generate DL-SPEC-UPDATE confirm (IDk, OK)
 - . set RQ of IDk to 0

The various tests carried out are based on data concerning:

- availability of buffer
 - buffer unavailable if B_REQ is already being accessed by the local link
- waiting for the request to be taken into account by the Bus Arbitrator
 - RQ of IDk is set at 1 by DL-SPEC-UPDATE request
 - RQ of IDk is set at 0 upon transmission of RP_RQ

A specific transfer request that contains an empty variables list makes it possible to clear a buffer of specific transfer requests.

31.3.4 Reception of a DL-FREE-UPDATE

Upon reception of a DL-FREE-UPDATE request primitive (LIST, PRIORITY) a producer triggers the following actions:

- * test queue
 - sufficient space not available,
immediate transmission of DL-FREE-UPDATE confirm (priority, saturation)
 - space available and queue not empty
store list of identifiers
 - queue empty
. store list of identifiers
. wait for next produced ID_DAT
- * reception of produced ID_DAT (IDp)
 - * test IDp
 - IDp not authorized
wait for next produced ID_DAT
 - IDp authorized
. associate IDp with the queue concerned
. set RQ of IDp at 1
. send RP_DAT_RQ
- * reception of ID_RQ (IDp)
 - * test status of queue
 - queue empty no answer
 - queue not empty
. emit RP_RQ with all stored lists of identifiers (up to a limit of 64 identifiers)
. generate as many DL-FREE-UPDATE confirm (OK) primitives as there are lists stored
. run through the lists of identifiers
. set RQ of IDp at 0
- * test queue status
 - queue not empty
wait for next produced ID_DAT

The various tests carried out are based on data concerning:

- status of the queue Q_REQi: queue filled or not, queue empty or not
wait for an authorized produced ID_DAT for DL-FREE-UPDATE. A mechanism makes it possible to prohibit the use of certain identifiers for DL-FREE-UPDATE.

31.3.5 Reception of a DL-MESSAGE request

Upon reception of a DL-MESSAGE request primitive (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message) a source entity triggers the following actions:

- * test SPECIFIED_IDENTIFIER
 - incorrect parameter (identifier not configured for cyclical message transfer, or value of parameter different from NIL)
immediate transmission of DL-MESSAGE confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, incorrect parameter)
 - parameter correct

- * test status of queue
 - queue full
 - . immediate transmission of DL-MESSAGE confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, queue full)
 - queue not full
 - . store message
 - . wait for ID_MSG (IDp) or ID_DAT (IDp)
- * reception of ID_DAT (IDp)
 - * test IDp
 - IDp configured for cyclical message transfer
 - . emit RP_DAT
 - . wait for next produced ID_DAT
 - IDp configured for aperiodic message transfer
 - * test MSG of IDp
 - MSG of IDp = 1
 - . emit RP_DAT_MSG
 - . wait for next ID_DAT
 - MSG of IDp = 0
 - * test status of the resource Q_MSGaper set aside for aperiodic message transfer
 - queue empty
 - . emit RP_DAT
 - queue not empty and number of identifiers associated with the resource equal to the number of places occupied in said resource
 - . emit RP_DAT
 - queue not empty and number of identifiers associated with the resource less than number of places occupied in the resource
 - . establish reference between IDp and the queue for messages waiting to be emitted set aside for aperiodic message transfer
 - . set MSG of IDp at 1
 - . emit RP_DAT_MSG
 - . wait for next ID_DAT
 - * reception of ID_MSG (IDp)
 - * test IDp
 - IDp shows no reference to a resource
 - . no response
 - IDp shows reference to a resource

- * test status of resource given by IDp
- if resource is empty, send RP_END
- else
 - . emit RP_MSG_NOACK with the first message of the queue given by IDp
 - . send RP_END
 - . send DL-MESSAGE confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message sent)
 - . free up place in queue
 - . if the resource is the queue set aside for aperiodic message transfer:
 - . set MSG of IDp at 0
 - . cancel reference between IDp and the queue.

The various tests carried out are based on data concerning:

- status of the queue used: empty, full, or space available
- indicator showing whether the Bus Arbitrator has taken the message request into account in the case of aperiodic message transfer

MSG is set at 1 through attribution of IDp to the queue set aside for aperiodic message transfer.

MSG is set at 0 upon emission of RP_END.

number of identifiers associated with the queue set aside for aperiodic message transfer, and number of places occupied in this queue.

31.3.6 Receipt of a DL-MESSAGE-Ack request

Upon reception of a DL-MESSAGE-Ack request primitive (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message) a source entity triggers the following actions:

- * test SPECIFIED_IDENTIFIER
 - incorrect parameter (identifier not configured for cyclical message transfer, or value of parameter different from NIL) immediate transmission of DL-MESSAGE-Ack confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, incorrect parameter)
 - parameter correct
- *test ADR Destination

ADR destination is a group address

 - . ignore the message
 - . immediate transmission of DL-MESSAGE-Ack confirm(Spec Id, ADR Dest, ADR Source, Group Address)
- * test status of queue
 - queue full
 - . immediate transmission of DL-MESSAGE-Ack confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, queue full)
 - queue not full
 - . store message
 - . wait for ID_MSG (IDp) or ID_DAT (IDp)

- * reception of ID_DAT (IDp)
- * test IDp
 - IDp configured for cyclical message transfer
 - . emit RP_DAT
 - . wait for next produced ID_DAT
 - IDp configured for aperiodic message transfer
- * test MSG of IDp
 - MSG of IDp = 1
 - . emit RP_DAT_MSG
 - . wait for next ID_DAT
 - MSG of IDp = 0
- * test status of the resource Q_MSGaper set aside for aperiodic message transfer
 - queue empty
 - . emit RP_DAT
 - queue not empty and number of identifiers associated with the resource equal to the number of places occupied in said resource
 - . emit RP_DAT
 - queue not empty and number of identifiers associated with the resource less than number of places occupied in the resource
 - . establish reference between IDp and the queue for messages waiting to be emitted set aside for aperiodic message transfer
 - . set MSG of IDp at 1
 - . emit RP_DAT_MSG
 - . wait for next ID_DAT
- * reception of ID_MSG (IDp)
- * test IDp
 - IDp shows no reference to a resource
 - . no response
 - IDp shows reference to a resource
- * test status of resource given by IDp
 - if resource is empty, send RP_END
 - else
 - . emit RP_MSG_ACK with the first message of the queue referenced by IDp
 - . launch timer for waiting for RP_ACK
 - . wait for acknowledgement DLPDU
- * reception of RP_ACK
- * test acknowledgement
 - acknowledgement positive
 - . send RP_END
 - . send DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, message sent)
 - . free up place in queue
 - . if the resource is the queue set aside for aperiodic message transfer:

- . set MSG of IDp at 0
- . cancel reference between IDp and the queue.
- acknowledgement negative
 - . send RP_END
 - . send DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, acknowledgement negative)
 - . free up place in queue
 - . if the resource is the queue set aside for aperiodic message transfer:
 - . set MSG of IDp at 0
 - . cancel reference between IDp and the queue.
- * expiration of timer for acknowledgement reception
- * test restart counter
 - the counter has not reached its maximum value
 - emit RP_MSG_ACK with the first message in the queue referenced by IDp
 - launch the timer for waiting for RP_ACK
 - wait for acknowledgement DLPDU
 - the counter has reached its maximum value
 - send RP_END
 - send DL-MESSAGE-ACK confirm (SPECIFIED_IDENTIFIER, ADRdestination, ADRsource, maximum number of restarts attained)
 - free up place in queue
 - if the resource is the queue set aside for aperiodic message transfer:
 - set MSG of IDp at 0
 - cancel reference between IDp and the queue.

The various tests carried out are based on data concerning:

- status of the queue used: empty, full, or space available
- indicator showing whether the Bus Arbitrator has taken the message request into account in the case of aperiodic message transfer MSG is set at 1 through attribution of IDp to the queue set aside for aperiodic message transfer MSG is set at 0 upon emission of RP_END.
- current value of the restart counter
 - number of identifiers associated with the queue set aside for aperiodic message transfer, and number of places occupied in this queue.

31.3.7 Invalid identifier

This part of standard define the actions to be performed at reception of an ID_XX DLPDU concerning an invalid identifier.

The invalidation of a DLCEP-identifier concerns at least one of the four services: cyclic messaging, specified explicit periodic request for transfer of identified objects, consumption, production. In all cases, a producer/consumer entity does not answer the demands from the medium which concern the invalidated service(s) of a DLCEP-identifier.

Besides, at invalidation of a DLCEP-identifier in production (Product_ID_Veracity attribute set to FALSE), the possible associations with Q_REQ1 or Q_REQ2 as well as with Q_MSG.aper are interrupted. The attachment attribute relative to Q_REQ1 or Q_REQ2 is set to FALSE if the identifier was associated with one of these files before being invalidated; the Attachment_number attribute relative to Q_MSG.aper is decreased by one if the identifier was associated with this file before being invalidated.

31.4 Bus arbitrator operation

31.4.1 General description of the bus arbitrator

The role of the Bus Arbitrator is to "give permission to speak" to each information producer, while taking into account the services needed by the user application.

The Bus Arbitrator thus has three functions:

- 1) periodic scanning of variables or triggered periodic scanning of variables and messages
- 2) triggered scanning of variables
- 3) triggered scanning of messages.

In addition, the Bus Arbitrator can provide a synchronization function to guarantee the constant length of a scanning cycle.

Each type of scanning takes place in a periodic window, an aperiodic variables window, an aperiodic messages window, or a synchronization window respectively.

The four windows together make up a basic scanning cycle.

The functions of the Bus Arbitrator are broken down into three layers, as are other DL-services. For each of these layers there is a set of arbitrating services and arbitrating data. The complete specification of the Bus Arbitrator will be found in 5.1 (Bus Arbitrator) of EN 50170-3.

The remainder of this chapter describes the Data Link functions provided by the Bus Arbitrator. The Data Link level of a Bus Arbitrator entity offers the following functions:

- a) emission of identifier DLPDU sequences,
- b) reception of DLPDUs containing identifiers,
- c) detection of requests for aperiodic variable and message exchanges
- d) emission of identifier DLPDUs used for synchronization.

The resources used to provide these functions are also described: they include the identifier sequences, queues for requests for aperiodic variable and message exchanges, the restart buffer.

The Bus Arbitrator's Data Link level oversees the proper outcome of the various basic transactions, and executes requests from the DLS-user for services, in order to properly carry out Basic Cycles and Macro Cycles.

31.4.2 Management of basic transactions

31.4.2.1 General

A basic transaction is made up of the successive DLPDUs related to a single service.

Managing basic transactions means controlling the succession of DLPDUs that make up a transaction. Valid basic transactions are:

ID_DAT + RP_DAT
 ID_DAT + RP_DAT_MSG
 ID_DAT + RP_DAT_RQi
 ID_DAT + RP_DAT_RQi_MSG
 ID_MSG + RP_MSG_NOACK + RP_END
 ID_MSG + RP_MSG_ACK + RP_ACK + (repetitions) + RP_END
 ID_RQi + RP_RQi

Subclause 31.4.6 specifies the Bus Arbitrator's simplified states machine as well as the associated transitions table.

31.4.2.2 Emitting a variable identifier

After emitting a variable identifier DLPDU (ID_DAT) the Bus Arbitrator activates the timer T1 and then waits for a variable response DLPDU:

- a) if the DLPDU received is a simple response (RP_DAT) the Bus Arbitrator goes on to the next identifier to be scanned
- b) if the DLPDU received is a variable response accompanied by a message transfer request (RP_DAT_MSG), and if the Bus Arbitrator is in the cyclical scanning phase, it records the identifier that carried the request in the queue for message transfer requests (Q_IDMSG) and goes on to the next identifier to be scanned
- c) if the DLPDU received is a variable response accompanied by an explicit request for a buffer transfer (RP_DAT_RQi), and if the Bus Arbitrator is in the cyclical scanning phase, it records the identifier that carried the request in the queue for explicit buffer transfer requests that has the proper priority (Q_IDRQi) and goes on to the next identifier to be scanned
- d) if the DLPDU received is a variable response accompanied by a request for a message transfer and an explicit request for a buffer transfer (RP_DAT_RQi_MSG) and if the Bus Arbitrator is in the cyclical scanning phase, it records the identifier that carried the request in the queue for message requests (Q_IDMSG) and in the queue for explicit buffer transfer requests that has the proper priority (Q_IDRQi) and goes on to the next identifier to be scanned
- e) if the DLPDU received is of a different type the Bus Arbitrator detects an error and goes on to the next identifier to be scanned
- f) if the DLPDU received has a flawed FCS the Bus Arbitrator detects a transmission error and goes on to the next identifier to be scanned
- g) if the T1 timer expires the Bus Arbitrator detects the loss of a response DLPDU and goes on to the next identifier to be scanned.

31.4.2.3 Emitting a message identifier

After emitting a message identifier DLPDU (ID_MSG), the Bus Arbitrator activates the T5 timer and waits for the response DLPDU indicating the end of the message transaction:

- a) if the DLPDU received is an end of message transaction the Bus Arbitrator goes on to the next identifier to be scanned.
- b) if the DLPDU received is an unacknowledged message (RP_MSG_NOACK), an acknowledged message (RP_MSG_ACK) or an acknowledgement response DLPDU (RP_ACK), the Bus Arbitrator activates T5 and continues to wait for a response DLPDU indicating the end of the message transaction.
- c) if the DLPDU received is of a different type the Bus Arbitrator detects an error and goes on to the next identifier to be scanned.
- d) if the DLPDU received has an incorrect residual FCS (see 5.1) the Bus Arbitrator detects a transmission error and continues to wait for the end of message transaction DLPDU.

- e) if the timer expires the Bus Arbitrator detects the loss of an end of message transaction DLPDU and goes on to the next identifier to be scanned.

31.4.2.4 Emitting a request identifier

After emitting a request identifier DLPDU (ID_RQi), the Bus Arbitrator activates the T1 timer and waits for a request response DLPDU:

- a) if the DLPDU received is an urgent request response DLPDU (RP_RQ1) and if the Bus Arbitrator is in its periodical scanning phase, it records the list of identifiers in the recovery buffer and immediately scans all the variable identifiers stored
- b) if the DLPDU received is an urgent request response DLPDU and if the Bus Arbitrator is in its aperiodic scanning phase, it records the list of identifiers in the aperiodic queue being processed and goes on to the next identifier to be scanned
- c) if the DLPDU received is a normal request response DLPDU (RP_RQ2) the Bus Arbitrator records the list of identifiers in the aperiodic queue being processed and goes on to the next identifier to be scanned
- d) if the DLPDU received is of a different type the Bus Arbitrator detects an error and goes on to the next identifier to be scanned.
- e) if the DLPDU received has a flawed FCS the Bus Arbitrator detects a transmission error and continues to wait for the end of message transaction DLPDU.
- f) if the timer T1 expires the Bus Arbitrator detects the loss of a response DLPDU and goes on to the next identifier to be scanned.

31.4.2.5 Synchronization window

During the synchronization window the Bus Arbitrator goes through a loop composed of the following actions:

- a) emission of a synchronization identifier DLPDU (this DLPDU corresponds, for example, to a DLCEP-identifier that is neither produced nor consumed) and activation of the T1 timer (used for waiting for a response DLPDU),
- b) if the Bus Arbitrator receives a response DLPDU or detects the expiration of the T1 timer, it restarts loop operations,
- c) if the Bus Arbitrator detects the expiration of the T2 timer signifying the end of the Basic Cycle, it abandons the loop and goes on to the next Basic Cycle.

This function guarantees the constant length of a Basic Cycle. The precise timing of the beginning of a cycle is thus at best equal to the length of the emission of a DLCEP-identifier DLPDU followed by a timer used to wait for a response DLPDU.

31.4.3 Resources used

The DLL uses two categories of resources to implement the Data Link level functions of the Bus Arbitrator:

- the basic sequences of identifier numbers that constitute the cyclical portion of the scanning table and make it possible to generate the bus's periodic flow,
- the queues used to store the numbers of identifiers that have carried a normal (Q_IDRQ2) or urgent (Q_IDRQ1) variable request.
- the queues used to store the numbers of identifiers that are the object of aperiodic variable exchange requests (Q_RPRQ).

(These four queues make it possible to generate the bus's aperiodic flow.)

- a recovery buffer for the immediate emission of identifiers involved in a triggered periodic request for a variable exchange,
- a basic padding sequence used to generate synchronization window flow.

For further details see 28.2.2.

31.4.4 Chaining of basic transactions

31.4.4.1 Bus arbitrator cycles

The basic function of the Bus Arbitrator is to grant access to the bus by emitting identifier DLPDUs.

Several types of identifier DLPDUs have been defined, and a hierarchy in the order of emission of these identifiers can be set up to accommodate specific user needs.

The Bus Arbitrator's highest priority function is to allocate the medium so that the cyclical transfer of variables, requests, and messages is guaranteed.

Allocation of the medium for explicit requests for variable or message transfers has a lower priority.

Screening time requirements are not identical for all the variables scanned cyclically, but are always multiples of a basic period.

A Basic Cycle includes the Bus Arbitrator's scanning of:

- a set of variable, request, and cyclical message identifiers based on the application. This set of identifiers characterizes the individual Basic Cycle,
- plus the aperiodic exchanges window,
- plus the message services window,
- plus the synchronization window.

A Macro Cycle is the group of Basic Cycles that are juxtaposed to obtain the scanning of all the application's cyclical identifiers.

The Macro Cycle covers all the application's cyclical communications needs, and provides an available bandwidth for triggered transactions.

A Basic Cycle thus has a maximum of 4 phases:

- P1: periodic scanning of variables (cyclical ID_DAT), triggered periodic scanning of variables (cyclical ID_RQi) or triggered periodic scanning of messages (cyclical ID_MSG)
- P2: triggered message scanning
- P3: triggered scanning of variables with management of the two priority levels
- P4: wait for synchronization signal if a synchronized scanning cycle is requested.

Restart is possible in P1 and P2.

P3 can precede P2.

A Macro Cycle includes at least one Basic Cycle.

All Basic Cycles include a P1 (periodic scanning) phase.

Chaining of cycles

A DLL system incorporates several variable-scanning periods, but Bus Arbitrator operations include only two levels of chaining:

- chaining of sequences to make up a Basic Cycle
- chaining of Basic Cycles to make up a Macro Cycle

In addition, chaining of basic cycles can be:

- with synchronization
- without synchronization.

Consequently, the DLS-user sends the Bus Arbitrator a set of service request primitives that describe the execution of the Basic Cycles (the flow unit on the bus) and the Macro Cycles (the repetitious unit on the bus, made up of the chaining of one or several Basic Cycles).

These primitives are used to express user needs both in terms of the information to be exchanged and in terms of the screening time constraints to be respected.

If the synchronized scanning function has not been requested, Phase P4 of the Basic Cycle is eliminated. The maximum length of the Basic Cycle can still be guaranteed, however. In this operating mode if the Basic Cycle continues beyond its maximum length the next Basic Cycle is automatically begun.

31.4.4.2 Cycle configuration parameters

Certain configuration parameters are connected with each phase of the Basic Cycle.

Parameters linked to phase P1 of a Basic Cycle are:

- the list of basic sequences that characterize the basic cycle. These identifiers will be scanned using a variable, request, or cyclical message identifier (ID_DAT, ID_RQ1 or ID_MSG).

In the case of a cyclical request transaction (ID_RQ1, RP_RQ1) the number of transactions (ID_DAT, RP_DAT) that immediately follow is limited by the entity making the request and is known in advance. Thus a maximum length for the transaction is guaranteed.

The parameter linked to phase P2 of a Basic Cycle is:

- the time limit on the window allocated to triggered message transfers. This limit is defined with respect to the beginning of the Basic Cycle. When this window exists, it corresponds at minimum to the transmission of an ID_MSG DLPDU, followed by an RP_MSG_NOACK DLPDU and a RP_END DLPDU. If acknowledged message transfers and a restart mechanism are supported the window corresponds to the transmission of an ID_MSG DLPDU, followed by a RP_MSG_ACK DLPDU, followed by a RP_ACK DLPDU and enough time for as many repetitions of RP_MSG_ACK, RP_ACK as the number of authorized restarts allows. All these DLPDUs are followed, of course, by an RP_END DLPDU.

The parameter linked to phase P3 of a Basic Cycle is:

- the time limit on the window allocated to triggered aperiodic variable exchanges. This limit is defined with respect to the beginning of the Basic Cycle. When this window exists, it corresponds at minimum to the transmission of an ID_RQi DLPDU, followed by an RP_RQi DLPDU.

The parameter linked to phase P4 of a Basic Cycle is:

- the total length of a Basic Cycle and the basic padding sequence emitted while waiting for the indication signaling the end of the Basic Cycle.

31.4.5 Multiple bus arbitrators

The information presented in this subclause is further developed in the document C46 605 Network Management.

The overall results given below are based on the hypothesis that the multiple Bus Arbitrators are connected to a single physical support.

- at a given instant ONLY ONE BA is active,
- a procedure for choosing the active BA is defined,
- the other BA listen to what is occurring on the physical support and manage their own scanning tables, but do not emit any identifier DLPDUs,
- a timer value is specified in the final states machine of each producer/consumer entity in order to detect any prolonged silence on the part of the active BA (timer T3),
- a protocol for the exchange of synchronization data can be used to synchronize Macro Cycles.

31.4.6 Bus arbitrator

A simplified state machine for the active Bus Arbitrator is shown in Figure 184 and Table 231.

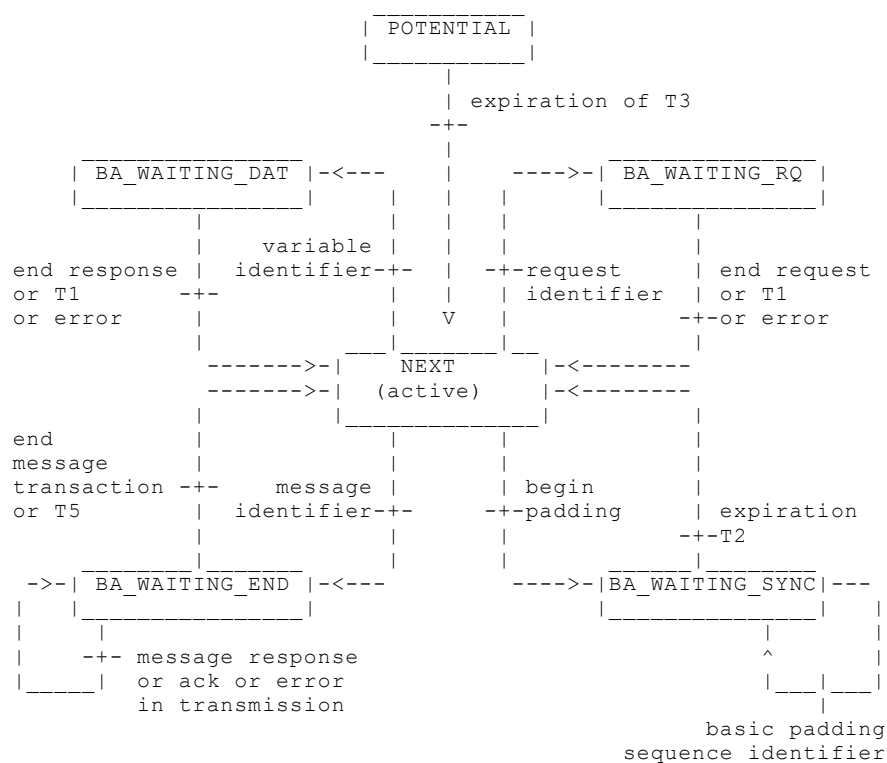


Figure 184 – Active bus arbitrator's simplified state machine

Table 231 – Bus arbitrator state transition table

Current state	Event	Action	Next state
(initial)			
	expiration T3		--> NEXT
NEXT			
	emission ID_DAT	activate T1	--> BA_WAITING_DAT
	emission ID_RQ1	activate T1	--> BA_WAITING_RQ
	emission ID_RQ2	activate T1	--> BA_WAITING_RQ
	emission ID_MSG	activate T5	--> BA_WAITING_END
	emission basic padding sequence		--> BA_WAITING_SYNC
BA_WAITING_DAT			
	RP_DAT		--> NEXT
	RP_DAT_MSG	record ID message emitter	--> NEXT
	RP_DAT_RQ1	record ID request emitter	--> NEXT
	RP_DAT_RQ2	record ID request emitter	--> NEXT
	RP_DAT_RQ1_MSG	record ID request emitter and ID message emitter	--> NEXT
	P_DAT_RQ2_MSG	record ID request emitter and ID message emitter	--> NEXT
	other DLPDU		--> NEXT
	transmission error		--> NEXT
	T1 timer	transaction aborted	--> NEXT
BA_WAITING_END			
	RP_END		--> NEXT
	RP_MSG_NOACK, RP_MSG_ACK, RP_ACK	activate T5	--> BA_WAITING_END
	other DLPDU		--> NEXT
	transmission error		--> BA_WAITING_END
	T5 timer	transaction aborted	--> NEXT
BA_WAITING_RQ			
	RP_RQ1	record list of identifiers in urgent queue	--> NEXT
	RP_RQ2	record list of identifiers in normal queue	--> NEXT
	other DLPDU		--> NEXT
	transmission error		--> NEXT
	T1 timer	transaction aborted	--> NEXT
BA_WAITING_SYNC			
	T2 not expired	emit identifier for the basic padding sequence	-->BA_WAITING_SYNC
	other DLPDU		--> NEXT
	T2 timer		--> NEXT

31.5 Bridges

31.5.1 Introduction

This chapter describes additional functions to allow the organization of a DL-subnetwork into several local links interconnected by bridges, as typified in Figure 185.

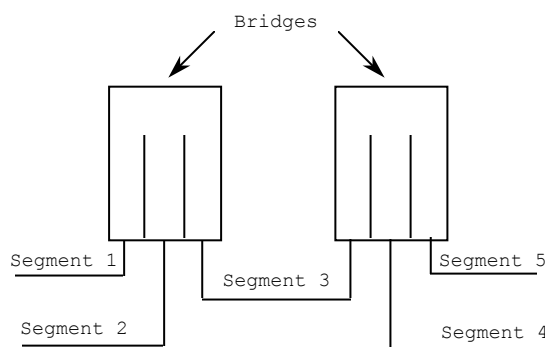


Figure 185 – Typical bridge usage

The specification of a bridge thus requires definition of the principle and the various possible modes for switching the information from one DL-segment to another.

31.5.2 Bridges

This subclause allows the reader to locate the bridge with respect to the DLL communication architecture functions.

31.5.2.1 Situation of the bridge in the DLL communication architecture

The bridge has a certain number of functions specific to the DLL (DLL) and particularly to the Logic Control (LC) level in order to allow a DL-subnetwork organization into several local links.

Each of the logical DL-segments of such a DL-subnetwork has a medium access control (MAC) independent from that of the other DL-segments. Consequently, there will be an active Bus Arbitrator on each of the local links.

Figure 186 shows the placement of bridges in the OSI Basic Reference Model (ISO/IEC 7498).

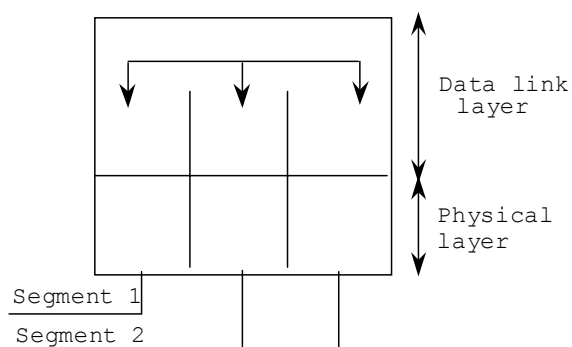


Figure 186 – Architectural placement of bridges in OSI Basic Reference Model (ISO/IEC 7498)

Because a bridge can store a received DLPDU for retransmission, the PhLs of the various local links can be of different speeds. They also can have differing media and modulation.

NOTE 1 Change of media or coding/modulation can also be achieved through Ph-relay devices, commonly called repeaters.

It is quite possible to envisage linking devices which are generalizations of bridges, and which could interconnect different types of DL-protocol. Usually there would be some loss of DL-service when transitioning the different protocol types.

All these possibilities of diversity pose specific problems, which are not dealt with in this addendum. However, a MAC address of the same size is required in all the local links, otherwise it is necessary to include address cross-reference tables.

NOTE 2 Having different access control on two interconnected DL-segments corresponds to considering, for example, a DL-subnetwork with a predominantly centralized administration such as the Type 7 DL-protocol, and another with a predominantly decentralized administration such as found in ISO/IEC 8802-4 or the Type 2 or Type 3 DL-protocol.

31.5.2.2 Communication through a bridge

Bridges are DL-relay devices which allow users of various local links to communicate without the user applications having to know or take into consideration the existence of these bridges, as in Figure 187.

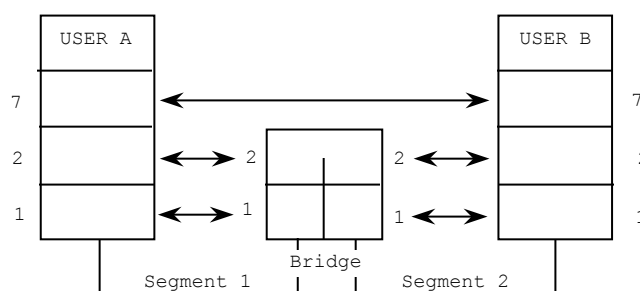


Figure 187 – Representation of an extended link communication

There are thus direct exchanges between the final devices belonging to various DL-segments at the level of the DLS-user and the users, whereas exchanges at the medium access control (MAC) level and the physical (Ph) level are performed from DL-segment to DL-segment.

The functions of such a bridge exclusively concern the messaging services.

NOTE The buffer transfer services can only be executed in a single-DL-segment environment.

31.5.3 Bridge function

31.5.3.1 General

A bridge is formed of several channels each of which has a PhL and a link layer allowing connection between various DL-segments.

The DLL functions relative to the cyclic messaging (outside the service primitives) are minimum functions, which must be supported to accept the bridges.

31.5.3.2 Link functions required

The link functions supported by each bridge channel can contain all the link layer services in case the bridge is also a station. However the minimum link layer resources and functions necessary for operation of the bridge are those of the cyclic messaging, with or without acknowledgement. These include:

- a queue, containing the messages to be transmitted, meant for cyclic transfer of messages called Q_MSGcyc,
- a queue full/not full indicator, associated with Q_MSG.cyc,

- a DLCEP-identifier configured for messaging,
- the attributes associated with a link entity supporting the messaging service with acknowledgement:
- . value of the even/odd bit, as source entity,
- . maximum value of the restart counter,
- . current value of the restart counter.
- a mechanism supporting the cyclic messaging service, whether acknowledged or not.

31.5.3.3 Properties

The extended link must have the properties of connectivity and non-meshing.

Connectivity consists in ensuring that there is always a path from any given DL-segment to any other DL-segment of the extended link .

Non-meshing consists in ensuring that the transmission of a message between the transmitter DL-segment and the destination DL-segment can only be executed along a single path (to prevent messages being received more than once).

These two rules allow calculating the local data base specific to each bridge in order to ensure its operation.

31.5.4 Object model

31.5.4.1 Bridge object description

A bridge is modeled on the link level with the help of the object described in Table 232:

Table 232 – Bridge object description

Class:	BRIDGE
Key Attribute:	Bridge identification
Attribute:	Operating status [ON; OFF]
Attribute:	<i>List of</i>
Attribute:	Channel <i>reference</i>

31.5.4.1.1 Identification of the bridge

This attribute corresponds to a number allowing unique identification of a bridge in the DL-subnetwork.

31.5.4.1.2 Operating mode

A bridge connected to a DL-subnetwork has two operating modes:

OFF: in this state a bridge does not ensure any of its functions. The passage from this state to another can only be obtained by a local action at the device level.

ON: in this state a bridge ensures its functions. The services offered by each channel are thus usable; the channels, considered individually, are not necessarily in operation.

31.5.4.1.3 List of channel references

This attribute allows designating all the bridge channels. Each of these channels has minimum functions allowing acceptance of the bridge retransmission mechanism.

31.5.4.2 Channel object description

Each of the bridge channels is modeled with the help of the object described in Table 233:

Table 233 – Channel object description

Class:	CHANNEL
Key Attribute:	Channel number
Attribute:	Segment number
Attribute:	Channel validity [Valid; Invalid]
Attribute:	Segment directory <i>reference</i>
Attribute:	Network directory <i>reference</i>

31.5.4.2.1 Channel number

This attribute is a number identifying a channel relative to a bridge.

31.5.4.2.2 Segment number

This attribute designates the segment connected to the bridge channel identified by the "Channel number" attribute.

31.5.4.2.3 Channel validity

A bridge channel has two states:

- VALID: This state corresponds to normal operation of a bridge channel.
It is able to:
 - retransmit messages from other channels,
 - redirect received messages towards other channels.
- INVALID: In this state a bridge channel does not ensure any of its functions.

The passage from one state to another is ensured by the System Management. These two states are necessary for the configuration and addition/deletion of bridges in the DL-subnetwork.

When the bridge is in the OFF state, each of its bridge channels is in the INVALID state.

31.5.4.2.4 Segment directory reference

The segment directory has a bridge channel giving the rules for retransmission of a message received by this channel on another channel of the same bridge, when the message destination address has a segment number (S/R=1).

The set of segment directories must be built to respect the properties of connectivity and non-meshing.

31.5.4.2.5 Network directory reference

The network directory which has a bridge channel gives the rules for retransmission of a message received by this channel on one or more channels of the same bridge when the message destination address does not have a segment number (S/R=0).

The set of network directories must be built in a manner observing the properties of connectivity and non-meshing.

31.5.4.3 Segment directory object description

The segment directory is modeled with the help of the object described in Table 234:

Table 234 – Segment directory object description

Class:	SEGMENT DIRECTORY
Key Attribute:	Directory number
Attribute:	<i>List of</i>
Attribute:	Recognized segment number
Attribute:	Retransmission channel number

31.5.4.3.1 Directory number

This attribute corresponds to a number allowing identification of the individual directory associated with a bridge channel.

31.5.4.3.2 Recognized segment number and retransmission channel number

The received messages with a destination address containing this segment number will be retransmitted on the associated retransmission channel.

31.5.4.4 Network directory object description

The network directory is modeled with the help of the object described in Table 235:

Table 235 – Network directory object description

Class:	NETWORK DIRECTORY
Key Attribute:	Directory number
Attribute:	<i>List of</i>
Attribute:	Retransmission channel number

31.5.4.4.1 Directory number

This attribute is a number allowing identification of the network directory associated with a bridge channel.

31.5.4.4.2 Retransmission channel number list

The messages received with a destination address not containing a segment number are retransmitted on each of the retransmission channels.

31.5.5 Mechanisms

The inter-DL-segment messaging can be acknowledged or not. In the first case, the acknowledgement is partial; each of the transactions performed on each of the DL-segments is acknowledged, but there is no acknowledgement from end to end.

The restart mechanism defined in this document is used in the case of transmission of acknowledged messages.

31.5.5.1 Retransmission rule

On reception of a message on one channel:

- Either the destination address contains a DL-segment number belonging to the channel segment directory: the bridge then retransmits the message on the corresponding retransmission channel.
- Or the destination address does not contain a DL-segment number: the bridge then retransmits the message on all the retransmission channels belonging to the receive channel network directory.

31.5.5.2 Retransmission triggering

On reception of an RP_MSG_ACK or RP_MSG_NOACK DLPDU containing a destination address observing the retransmission rule, a bridge channel stores the message:

- in the Q_MSG cycle of the retransmission channel if the destination address contains a DL-segment number,
- in the Q_MSG.cyc file of each of the retransmission channels if the destination address does not contain a DL-segment number.

On reception of a messaging DLPDU, a bridge channel does not call the indication primitives. The acknowledgement status (positive or negative) is determined by the state (full or not) of the retransmission channel Q_MSG.cyc queue.

31.5.5.3 Retransmission mechanism

The retransmission mechanism is triggered on reception of an RP_MSG_NOACK or RP_MSG_ACK DLPDU. The evaluation networks of

Figure 188 and

Figure 189 describe this mechanism.

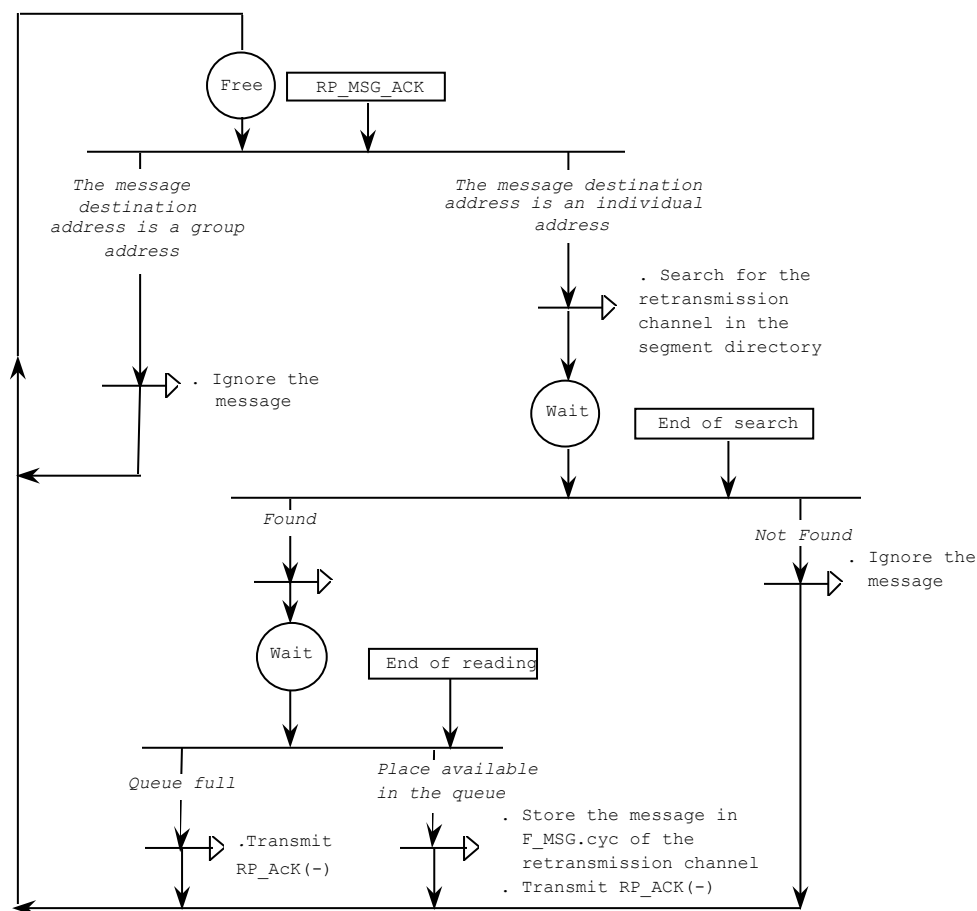


Figure 188 – Evaluation network for reception of an RP_MSG_ACK DLPDU

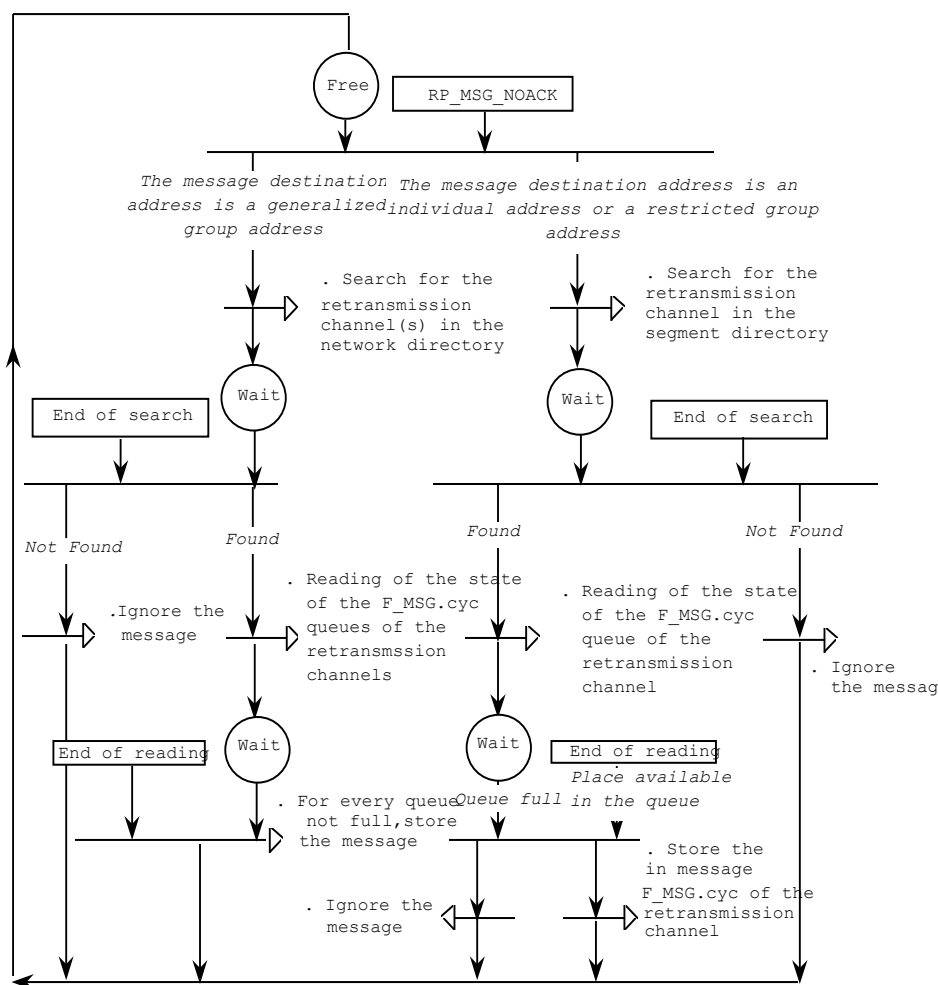


Figure 189 – Evaluation network for reception of an RP_MSG_NOACK DLPDU

31.6 Interfaces

31.6.1 DLS-user interface

The DLL provides the DLS-user with the following set of services:

- buffer writing
- buffer reading
- buffer transfer
- specified explicit request for a buffer transfer
- free explicit request for a buffer transfer
- unacknowledged message transfer request
- acknowledged message transfer request

The primitives associated with each service completely describe the interface between the Data Link and DLS-users. Detailed specification of primitives can be found in Part 3 of this standard.

The primitives associated with each service are given in Table 236.

Table 236 – Service primitives by type

Type of service	Primitive
Buffer writing:	DL-PUT request
	DL-PUT confirm
Buffer reading:	DL-GET request
	DL-GET confirm
Buffer transfer:	DL-SENT indication
	DL-RECEIVED indication
Specified explicit request for a buffer transfer:	DL-SPEC-UPDATE request
	DL-SPEC-UPDATE confirm
Free explicit request for a buffer transfer:	DL-FREE-UPDATE request
	DL-FREE-UPDATE confirm
Unacknowledged message transfer request:	DL-MESSAGE request
	DL-MESSAGE confirm
	DL-MESSAGE indication
Acknowledged message transfer request	DL-MESSAGE-ACK request
	DL-MESSAGE-ACK indication
	DL-MESSAGE-ACK confirm

31.6.2 DL- Management interface

DL-Management offers a number of services that will be specified in the Management document. These services make possible:

Control of DLE functioning modes such as:

- initialization of the DLE operating function,
- control of bus arbitrating,
- changing the Bus Arbitrator functioning mode.

Reports on events concerning changes within the DLE such as:

- expiration of a timer
- queue full
- Bus Arbitrator function change from PASSIVE to ACTIVE

Parameter setting for DLE functions such as:

- installation of the various Data Link objects, that is, the resources associated with identifiers for cyclical and aperiodic services, queues for messages, the Bus Arbitrator's scanning table
- turnaround time,
- the basic timer T0,
- the maximum value of the restart counter,
- the assigning of a physical address to the device.

Access to DLE error and performance counters such as:

- DLPDU fragment error
- FCS error
- DLPDU length error
- pierced DLPDU error

- coding error
- overrun error
- DLPDU sequencing error
- DLPDU type not identified
- message reception file saturated
- message acknowledgement lost
- number of cyclical transactions
- number of aperiodic transactions
- number of message service transactions
- number of messages refused
- total number of restarts

This set of device-level locally available services is used locally by a specific application dedicated to system management.

31.6.3 PhL interface

See IEC 61158-2.

31.7 Conformance

The DLL provides the DLS-user with the following set of services:

- a) buffer writing,
- b) buffer reading,
- c) buffer transfer,
- d) specified explicit request for buffer transfer,
- e) free explicit request for buffer transfer,
- f) unacknowledged message transfer request,
- g) acknowledged message transfer request.

A DLE provides the buffer transfer service and the buffer writing service in all cases, and, according to conformance classes, at least one of the following services:

- 1) buffer reading,
- 2) specified or free explicit request for a buffer transfer,
- 3) unacknowledged message transfer request,
- 4) acknowledged message transfer request.

The reading service makes it possible to extract data from the DLL and route it to the DLS-user. The service is initiated by the DLS-user for the purpose of reading the value of an identified variable.

The writing service makes it possible to deposit data from the DLS-user in the DLL. The service is initiated by the DLS-user for the purpose of writing the value of an identified variable.

The buffer transfer service participates in variable exchanges. In this service the producer of a variable emits a variable response DLPDU for the variable concerned. This DLPDU is then received by the consumer or consumers of the variable. The service generates a report (indication) for each variable emitted or received via the medium and sends this report to the DLS-user.

The specified explicit request for buffer transfer makes it possible to give the Bus Arbitrator a request for the circulation of one or more identifier DLPDUs. The transfer of the associated identified variable or variables is thus triggered. This service is initiated by the DLS-user for the purpose of placing one or more identifiers in circulation.

When the service is requested it is linked to a specific identifier, since the request is transmitted to the Bus Arbitrator during the cyclical variable exchange concerning this identifier.

The request is fulfilled during the aperiodic variable window (triggered scanning of variables) or during the Bus Arbitrator's periodic window (periodic triggered scanning of variables) depending on configuration.

The free explicit request for buffer transfer makes it possible to give the Bus Arbitrator a request for the circulation of one or more identifier DLPDUs. Transfer of the associated identified variable or variables is thus triggered. This service is initiated by the DLS-user for the purpose of placing one or more identifiers in circulation.

No identifier is specified when this service is requested. The request is transmitted to the Bus Arbitrator during the first exchange of a variable produced by the initiating entity.

The request is fulfilled only during the Bus Arbitrator's aperiodic variable window (triggered scanning of variables).

When the Bus Arbitrator receives an explicit (specified or free) request for a buffer transfer it begins a transaction that conforms to the buffer transfer service previously described.

For each free request for a buffer transfer, two priority levels are provided: the Bus Arbitrator processes the request in the urgent service or normal service mode.

Only urgent service is provided for a specified explicit request for a buffer transfer.

These two types of service make it possible to manage the priority levels of triggered variable exchanges without negative impact on cyclical variable exchanges or message transfers.

The unacknowledged message transfer request service causes the Bus Arbitrator to put a message identifier DLPDU in circulation. Circulation of this DLPDU triggers the transfer of an unacknowledged message.

The request is fulfilled either during aperiodic scanning or during the Bus Arbitrator's periodical scanning, depending on configuration.

If the request is filled during aperiodic scanning the unacknowledged message transfer is deemed aperiodic. The message transfer request is transmitted to the Bus Arbitrator during the first exchange of a variable produced by the source entity. The message identifier DLPDU is then put in circulation during the Bus Arbitrator's aperiodic messages window (triggered message scanning).

If the request is filled during the Bus Arbitrator's periodic scanning, the unacknowledged message transfer is deemed cyclical. In this case the message transfer request is in the periodical window because of configuration. Thus the message identifier DLPDU is put in circulation without prior request.

In both cases the message exchange between entities then makes use of an unacknowledged message response DLPDU.

When the transaction has been completed the source entity sends the Bus Arbitrator a DLPDU indicating the end of the message transaction.

The acknowledged message transfer request service causes the Bus Arbitrator to put a message identifier DLPDU in circulation. Circulation of this DLPDU triggers the transfer of an acknowledged message.

The request is fulfilled either during aperiodic scanning or during the Bus Arbitrator's periodical scanning, depending on configuration.

If the request is filled during aperiodic scanning the acknowledged message transfer is deemed aperiodic. The message transfer request is transmitted to the Bus Arbitrator during the first exchange of a variable produced by the source entity. The message identifier DLPDU is then put in circulation during the Bus Arbitrator's aperiodic messages window (triggered message scanning).

If the request is filled during the Bus Arbitrator's periodic scanning, the acknowledged message transfer is deemed cyclical. In this case the message transfer request is in the periodical window because of configuration. Thus the message identifier DLPDU is put in circulation without prior request.

In both cases the message exchange between entities then makes use of an acknowledged message response DLPDU.

Upon reception of the message, the receiving entity immediately sends the source entity an acknowledgement DLPDU.

When the transaction has been completed the source entity sends the Bus Arbitrator a DLPDU indicating the end of the message transaction.

NOTE Specifications for the DLL support several Application message service entities of various types.

A DLE offers the conformance classes enumerated in Table 237:

Table 237 – Conformance classes

Class	1	2	3	4	5	6	7	8	9
Service									
Buffer transfer	X	X	X	X	X	X	X	X	X
Buffer write	X	X	X	X	X	X	X	X	X
Buffer read		X	X	X	X	X	X	X	X
Free explicit request				X	X			X	X
Specified explicit request			X		X				X
Unacknowledged message						X	X	X	X
Unacknowledged message							X	X	X

32 Type 8: DL-protocol

32.1 Overview

The DLL is modelled as a Four-Level model (see Figure 190).

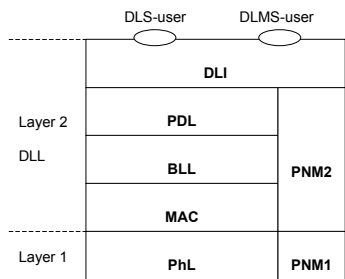


Figure 190 – Data Link Layer Entity

32.2 DL-service Interface (DLI)

32.2.1 General

The Data Link service Interface (DLI) provides service primitives to the DLS-user and DLMS-user (see Figure 191).

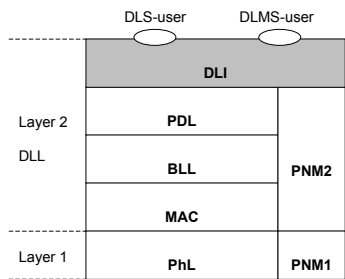


Figure 191 – Location of the DLI in the DLL

The DLI translates and issues the primitives received from the DLS-/DLMS-user to the local PDL and PNM2 interface. It also translates and issues the primitives received from the local PDL or PNM2 interface and delivers it to the DLS-/DLMS-user.

The DLI protocol has only a single state called “ACTIVE”

32.2.2 Primitive definitions

32.2.2.1 General

Table 238 and Table 239 show the primitives exchanged between DLS-/DLMS-user and DLI.

32.2.2.2 Primitives exchanged between DLS-/DLMS-user and DLI

Table 238 – Primitives issued by DLS-/DLMS-user to DLI

Primitive name	Source	Associated parameters	Functions
DL-PUT request	DLS-user	Buffer DL-identifier, DLS-user-data	Requests the DLE to write a DLSDU into the transmit buffer
DL-GET request	DLS-user	Buffer DL-identifier	Requests the DLE to read a DLSDU from the receive buffer
DL-DATA request	DLS-user	DLCEP DL-identifier, DLS-user-data	Requests the DLE to write a DLSDU into the send queue
DLM-RESET request	DLS-user	(<none>)	Requests the DLE to execute a reset.
DLM-SET-VALUE request	DLMS-user	Variable-name, Desired-value	Requests the DLE to overwrite a local variable
DLM-GET-VALUE request	DLMS-user	Variable-name	This Primitive is issued to request the DLL to read the content of a local variable
DLM-GET-CURRENT-CONFIGURATION request	DLMS-user	Desired-configuration	Requests the DLE to read out the current configuration of the DL-subnetwork.
DLM-GET-ACTIVE-CONFIGURATION request	DLMS-user	(<none>)	Requests the DLE to read out the active configuration of the DL-subnetwork
DLM-SET_ACTIVE_CONFIGURATION request	DLMS-user	Active-configuration	Requests the DLE to execute a certain active configuration of the DL-subnetwork

Table 239 – Primitives issued by DLI to DLS-/DLMS-user

Primitive name	Source	Associated parameters
DL-PUT confirm	DLI	Status
DL-GET confirm	DLI	Status, DLS-user-data
DL-BUFFER-RECEIVED indication	DLI	Status
DL-DATA confirm	DLI	Status
DL-DATA indication	DLI	DLCEP DL-identifier, DLS-user-data
DLM-RESET confirm	DLI	Status
DLM-EVENT indication	DLI	Event-identifier, Additional-information
DLM-SET-VALUE confirm	DLI	Status
DLM-GET-VALUE confirm	DLI	Status, Additional-information
DLM-GET-CURRENT-CONFIGURATION confirm	DLI	Status, Additional-information
DLM-GET-ACTIVE-CONFIGURATION confirm	DLI	Status, Additional-information
DLM-SET-ACTIVE-CONFIGURATION confirm	DLI	Status, Additional-information

32.2.2.3 Parameters of DLS-/DLMS-User and DLI primitives

All parameters used in the primitives exchanged between the DLS-/DLMS-user and the DLI are specified in IEC 61158-3, Type 8.

32.2.3 DLI State Tables

32.2.3.1 General

Figure 192 show a state transition diagram of the DLI.

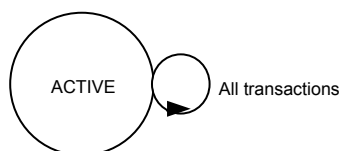


Figure 192 – State transition diagram of DLI

The transitions of the DLI protocol are specified in Table 240 and Table 241. Service primitive names are mixed-case with underscores (“_”) replacing dashes (“-”), and with a dot-separated suffix indicating the underlying type of primitive: request, confirm or indication.

Table 240 – DLI state table – sender transactions

#	Current state	Event Action	Next state
S1	ACTIVE	DL_Put.request PSM.request{ offset := GetOffset(Buffer DL-identifier) length := “length of DLS-user-data” data := DLS-user-data }	ACTIVE
S2	ACTIVE	DL_Get.request GSM.request{ offset := GetOffset(Buffer DL-identifier) length := GetLength(Buffer DL-identifier) }	ACTIVE
S3	ACTIVE	DL_Data.request{ PDL_Data_Ack.request{ rem_add := GetRemAdd (DLCEP DL-identifier) DLSDU := DLS-user-data }	ACTIVE
S4	ACTIVE	DLM_Reset.request PNM2_Reset.request{ }	ACTIVE
S5	ACTIVE	DLM_Set_Value.request PNM2_Set_Value.request { variable_name := Variable-name, desired_value := Desired-value }	ACTIVE
S6	ACTIVE	DLM_Get_Value.request PNM2_Get_Value.request{ variable_name := Variable-name }	ACTIVE
S7	ACTIVE	DLM_Get_Current_Configuration.request PNM2_Get_Current_Configuration.request{ network_configuration := Desired Configuration }	ACTIVE
S8	ACTIVE	DLM_Get_Active_Configuration.request PNM2_Get_Active_Configuration.request{ }	ACTIVE
S9	ACTIVE	DLM_Set_Active_Configuration.request PNM2_Set_Active_Configuration.request{ active_configuration := Active-configuration }	ACTIVE

Table 241 – DLI state table – receiver transactions

#	Current state	Event Action	Next state
R1	ACTIVE	PSM.confirm DL_Put.confirm{ Status := status }	ACTIVE
R2	ACTIVE	GSM.confirm DL_Get.confirm{ Status := status, DLS-user-data := data }	ACTIVE
R3	ACTIVE	Buffer_Received.indication DL_Buffer_Received.indication{ Status := status }	ACTIVE
R4	ACTIVE	PDL_Data_Ack.confirm DL_Data.confirm{ Status := L_status }	ACTIVE
R5	ACTIVE	PDL_Data_Ack.indication DL_Data.indication{ DLCEP DL-identifier := GetDlsUserId(local_add), DLS-user-data := DLSDU }	ACTIVE
R6	ACTIVE	PNM2_Reset.confirm DLM_Reset.confirm{ Status := M_status }	ACTIVE
R7	ACTIVE	PNM2_Event.indication DLM_Event.indication { Event-identifier := event, Additional-information := add_info }	ACTIVE
R8	ACTIVE	PNM2_Set_Value.confirm DLM_Set_Value.confirm { Status := M_status }	ACTIVE
R9	ACTIVE	PNM2_Get_Value.confirm DLM_Get_Value.confirm{ Status := M_status Current-Value := current_value }	ACTIVE
R10	ACTIVE	PNM2_Get_Current_Configuration.confirm DLM_Get_Current_Configuration.confirm{ Status := status, Current-configuration := current_configuration }	ACTIVE
R11	ACTIVE	PNM2_Get_Active_Configuration.confirm DLM_Get_Active_Configuration.confirm{ Status := status, Active-configuration := active_configuration }	ACTIVE
R12	ACTIVE	PNM2_Set_Active_Configuration.confirm DLM_Set_Active_Configuration.confirm{ Status := status, Additional-information := add_info }	ACTIVE

32.2.3.2 Functions used by DLI

The functions used by DLI are given in Table 242 to Table 245. The details of these functions is not specified by of this standard. These functions use information which was stored by the local DL-management when establishing the DLCs.

Table 242 – Function GetOffset

Name	GetOffset	Used in	DLI
Input	Buffer DL-identifier	Output	Offset address
Function	Returns a value that can unambiguously identify the offset address from the transmit buffer		

Table 243 – Function GetLength

Name	GetLength	Used in	DLI
Input	Buffer DL-identifier	Output	Length of data
Function	Returns the size of the DLSDU which can be held by the buffer named by Buffer DL-identifier.		

Table 244 – Function GetRemAdd

Name	GetRemAdd	Used in	DLI
Input	DLCEP DL-identifier	Output	Remote address
Function	Returns a value that can unambiguously identify the remote address from the remote device		

Table 245 – Function GetDisUserId

Name	GetDisUserId	Used in	DLI
Input	Local address	Output	DLCEP DL-identifier
Function	Returns a value that can unambiguously identify the DLCEP DL-identifier from the DLS user		

32.3 Peripherals data link (PDL)

32.3.1 Location of the PDL in the DLL

The Peripherals Data Link (PDL) is part of the Data Link Layer and uses the Basic Link Layer. Figure 193 shows its location.

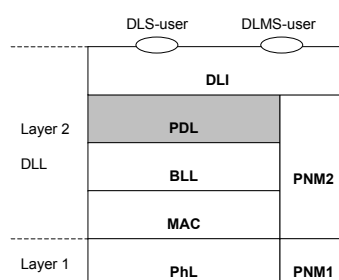


Figure 193 – Location of the PDL in the DLL

By means of the PDL layer each slave can establish a communication link with the master (see Figure 194).

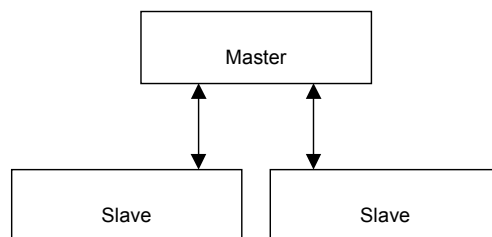


Figure 194 – PDL connection between slave and master

32.3.2 Functionality of the PDL

The PDL performs the following tasks:

- Processing of PDL_Data_Ack service
- Conversion of the non-cyclic PDL_Data_Ack service to cyclic BLL_Data services and vice versa
- Conversion of several DLSDUs of the PDL_Data_Ack.request primitives into a PDLSDU of the BLL_Data.request primitive
- Implementation of two trigger_modes within the PDL (*bus master only*)
- Control of the local PDL protocol machine(s)
- Update of the receive update memory and starting of the PDL protocol machines after a PDLSDU which was received from the BLL has been accepted,
- Generation of a PDLSDU from the transmit update memory as well as by means of the PDL protocol machines and transfer of this PDLSDU to be sent to the BLL
- Implementation of a direct access for PDL-user to the PDL receive and transmit update memory

NOTE A PDLSDU of the master contains all cyclic data via PSM service to be transmitted in a data cycle and PDL message segments. The PDLSDU of a slave is a subset of the PDLSDU of the master and contains only the cyclic data to be transmitted in one data cycle and the PDL message segment of this slave

The PDL translates these functions by means of the four following protocol machines:

- PDL base protocol machine
- PDL protocol machine
- TRANSMIT protocol machine
- RECEIVE protocol machine

32.3.3 DLI-PDL interface

32.3.3.1 General

The PDL provides service primitives for the PDL-user (see Figure 195).

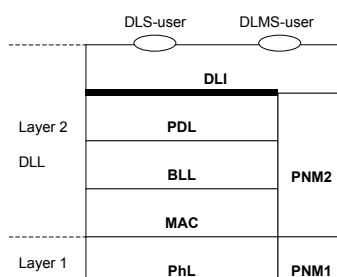


Figure 195 – Interface between PDL-user (DLI) and PDL in the layer model

32.3.3 describes the data transmission services which are available to the PDL-user, together with their service primitives and their associated parameters. These PDL services are mandatory.

32.3.3.2 Overview of the services

32.3.3.2.1 Available services

The following service for data transfer shall be available to the PDL-user:

- Send Parameter with Acknowledge (PDL_Data_Ack)

Furthermore, the PDL-user can use the following services to directly access the update memory.

- **Put Shared Memory (PSM)**
- **Get Shared Memory (GSM)**

Figure 196 shows an overview of the services of the PDL.

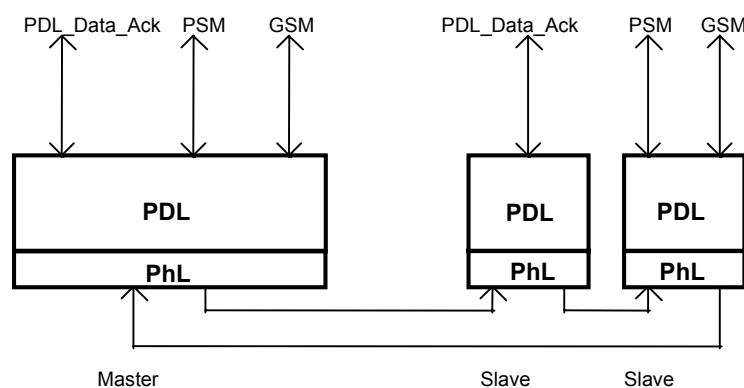


Figure 196 – Overview of the PDL services

32.3.3.2.2 Send parameter with acknowledge (PDL_Data_Ack)

This service allows a local PDL-user to send user data (DLSDU) to a single remote PDL-user. The remote PDL transfers the DLSDU to its PDL-user, provided that the DLSDU was received without errors. The local PDL-user receives a confirmation on the receipt or non-receipt of the DLSDU of the remote PDL.

The PDL_Data_Ack service shall only be used to transfer data from a queue.

Service primitives:

- PDL_Data_Ack.request
- PDL_Data_Ack.indication
- PDL_Data_Ack.confirm

32.3.3.2.3 Put shared memory (PSM)

This service allows a PDL-user to write data of a certain length into the transmit update memory. The BLL shall transmit this data in the next bus cycle.

Service primitives:

- PSM.request
- PSM.confirm

32.3.3.2.4 Get shared memory (GSM)

This service allows a PDL-user to read data of a certain length from the receive update memory.

Service primitives:

- GSM.request
- GSM.confirm

32.3.3.2.5 Buffer received (Buffer_Received)

The PDL uses this service to indicate the local PDL-user, that the contents of

Transmit Update Memory is transmitted, and the contents of
Receive Update Memory is updated with new received data.

Service primitive:

Buffer_Received.indication

32.3.3.3 Overview of the interactions

The services are provided by several service primitives (beginning with PDL_...). In order to request a service, the PDL-user uses a request primitive. A confirmation primitive is returned to the PDL-user after the service has been completed. The arrival of a service request is indicated to the remote PDL-user by means of an indication primitive.

Figure 197, Figure 198, Figure 199 and Figure 200 show the sequences of service primitives to handle the data transfer between master and slave:

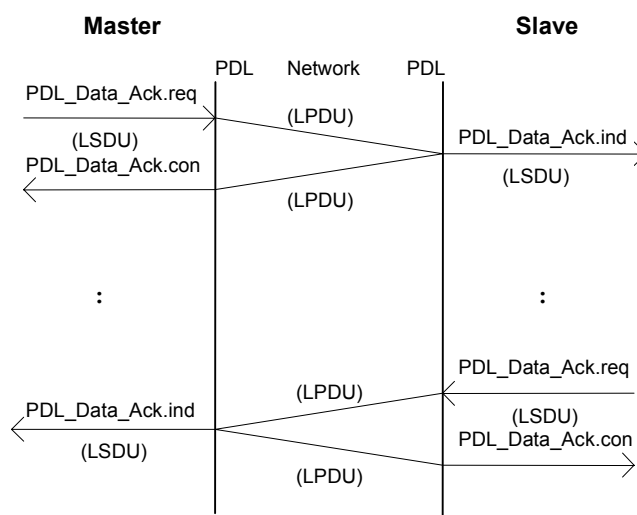


Figure 197 – PDL_Data_Ack service between master and only one slave

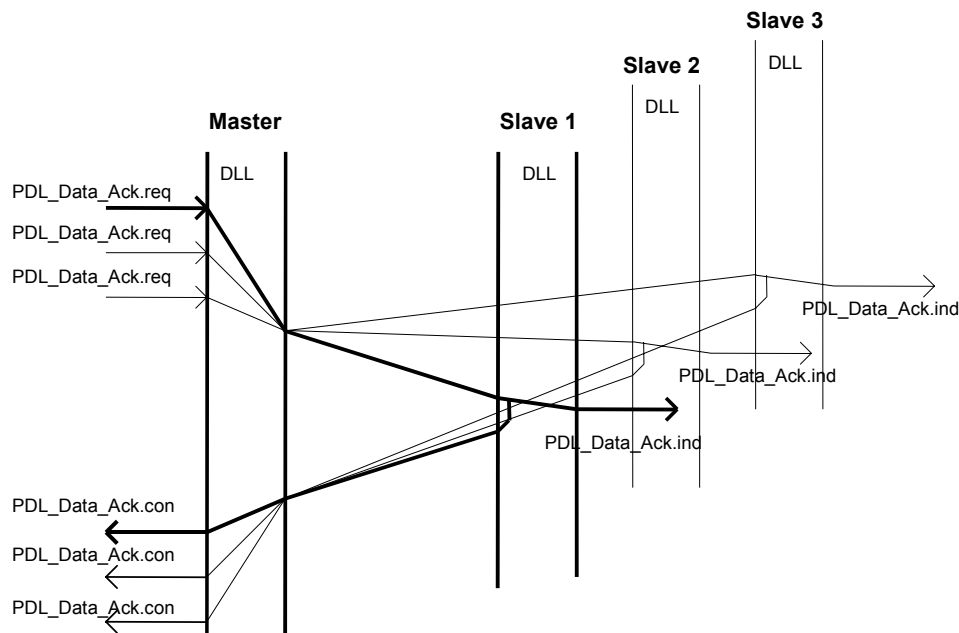


Figure 198 – Parallel processing of PDL_Data_Ack services

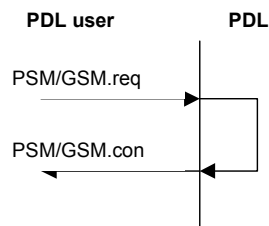


Figure 199 – PSM and GSM service for buffer access

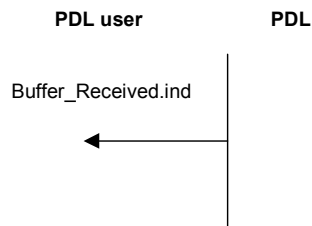


Figure 200 – Buffer_Received service to indicate successful data transfer

32.3.3.4 Formal description of the services and parameters

32.3.3.4.1 PDL_Data_Ack Service

Table 246 shows the parameters of the PDL_Data_Ack service.

Table 246 – PDL_Data_Ack

Parameter name	Request	Indication	Confirm
Argument	M	M	
rem_add	M		
local_add		M	
DLSDU	M	M(=)	
Result			M
rem_add			M
L_status			M

rem_add:

The rem_add parameter defines the PDL address of the remote device. The rem_add corresponds to the physical position of the device in the ring.

local_add:

The local_add parameter conveys the PDL address of the device where the PDL_Data_Ack service was invoked.

DLSDU:

The DLSDU parameter contains the PDL-user data to be transmitted.

L_status:

The L_status parameter indicates the success or failure of the preceding PDL_Data_Ack.request. The following values are defined for this parameter in Table 247:

Table 247 – PDL_Data_Ack L_status values

Value	Meaning
OK	Positive acknowledgement, service executed successfully
RR	Negative acknowledgement, resources of the remote PDL not available or insufficient
LR	Resources of the local PDL not available or insufficient
NA	No or not a plausible response (acknowledge response) from the remote device
DS	PDL layer not synchronized at the moment
IV	Invalid parameter in the request call

32.3.3.4.2 PSM service

Table 248 shows the parameters of the PSM service.

Table 248 – PSM

Parameter name	Request	Confirm
Argument	M	
offset	M	
length	M	
data	M	
Result(+)		S
Result(-)		S
error_type		M

offset:

This parameter specifies the offset address, beginning from the start address of the PDL transmit update memory, where the data should be written.

length:

This parameter specifies the amount of the data, which should be written into the PDL transmit update memory of layer 2.

data:

This parameter conveys the data, which should be written into the PDL transmit update memory of the layer 2.

error_type:

This parameter indicates the reason, why the service could not be executed successfully.

Possible errors are:

- IV Invalid parameters in the request call
Data to write into the transmit update memory are not allowed, because the given parameter(s) of offset and/or length is/are invalid.

32.3.3.4.3 GSM service

Table 249 shows the parameters of the GSM service.

Table 249 – GSM

Parameter name	Request	Confirm
Argument	M	
offset	M	
length	M	
Result(+)		S
data		M
Result(-)		S
error_type		M

offset:

This parameter specifies the offset address, beginning from the start address of the PDL receive update memory, from where the data should be read.

length:

This parameter specifies the amount of the data, which should be read from the PDL receive update memory.

data:

This parameter conveys the data, which was read from the PDL receive update memory.

error_type:

This parameter indicates the reason, why the service could not be executed successfully.
Possible error sources:

- IV Invalid parameters in the request call
Data to be read from the receive update memory are not allowed, because the given parameter(s) of offset and/or length is/are invalid.

32.3.3.5 Detailed description of the interactions

32.3.3.5.1 Send parameter with acknowledge (PDL_Data_Ack)

The local PDL-user prepares a DLSDU which is transmitted by a PDL_Data_Ack.request primitive to the local PDL. The PDL accepts this service request and tries to send the DLSDU to the requested remote PDL. The local PDL sends a confirmation to its PDL-user with the PDL_Data_Ack.confirm primitive, which indicates a correct or incorrect data transfer.

Before the local PDL sends a confirmation to its user, a confirmation from the remote PDL is mandatory. If this confirmation is not received within the timeout period $T_{TO\_SPA\_ACK}$, the local PDL retries to send the DLSDU to the remote PDL. If the confirmation does not come after the Nth repetition (max_retry_count), then the local PDL sends a negative confirmation to its user.

If the data message was received without errors, the remote PDL transfers the DLSDU with a PDL_Data_Ack.indication primitive through the PDL-user interface.

The coding of the DLSDU is described in 32.3.5.3. Figure 201 shows the data flow between PDL-user, PDL and BLL for a PDL_Data_Ack service:

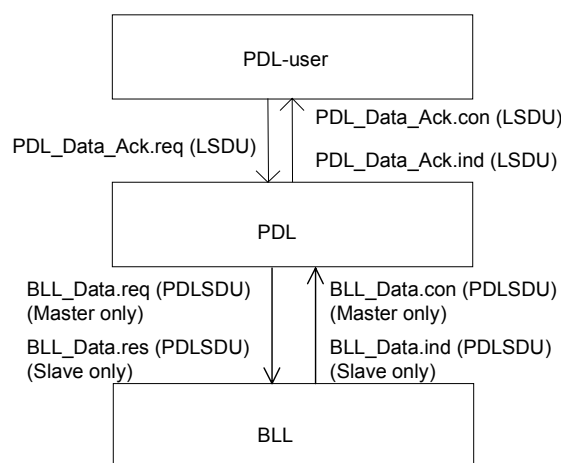


Figure 201 – Data flow between PDL-user, PDL and BLL of a PDL_Data_Ack service

32.3.3.5.2 Put shared memory (PSM)

The PDL-user uses this service to write user data directly to the transmit update memory. The service is locally processed after the PSM.request primitive has arrived. The PDL communicates the successful processing of the service to its PDL-user by means of a PSM.confirm primitive (immediate confirmation).

32.3.3.5.3 Get shared memory (GSM)

The PDL-user uses this service to read user data directly from the PDL receive update memory. The service is locally processed after the GSM.request primitive has arrived. The PDL communicates the successful processing of the service to the PDL-user by means of a GSM.confirm primitive (immediate confirmation).

32.3.4 PDL-PNM2 interface

32.3.4.1 General

This subclause defines the administrative PDL management services which are available to the PNM2, together with their service primitives and the associated parameters.

The PDL management is a part of the PDL that provides the management functions of the PDL requested by the PNM2. The PDL management handles the initialization, monitoring and error recovery in the PDL. Figure 202 shows the interface between PDL and PNM2 in the layer model.

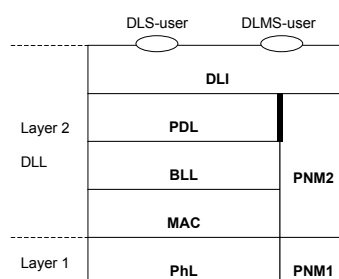


Figure 202 – Interface between PDL and PNM2 in the layer model

The service interface between PDL and PNM2 provides the following functions:

- Reset of the PDL protocol machine
- Request and change of the current operating parameters of the PDL protocol machine
- Indication of unexpected events, errors and status changes which occurred or are detected in the PDL

32.3.4.2 Overview of the services

32.3.4.2.1 Available services

The PDL makes the following services available to the PNM2:

- Reset PDL
- Set Value PDL or Get Value PDL
- Event PDL

The PDL services are described with service primitives (beginning with PDL_...).

32.3.4.2.2 Reset PDL

The PNM2 uses this required service to reset the PDL. Upon execution, the PNM2 receives a confirmation.

Service primitives:

- PDL_Reset.request
- PDL_Reset.confirm

32.3.4.2.3 Set Value PDL

The PNM2 uses this optional service to set new values to the PDL variables. Upon completion, the PNM2 receives a confirmation from the PDL whether the defined variables are assumed with the new value.

Service primitives:

- PDL_Set_Value.request
- PDL_Set_Value.confirm

32.3.4.2.4 Get Value PDL

The PNM2 uses this optional service to read the actual value of the PDL variables. The current value of the defined variable is transmitted with the confirmation from the PDL.

Service primitives:

- PDL_Get_Value.request
- PDL_Get_Value.confirm

32.3.4.2.5 Event PDL

The PDL uses this required service to inform the PNM2 about certain detected events or errors in the PDL.

Service primitive:

- PDL_Event.indication

32.3.4.3 Overview of the interactions

Figure 203 and Figure 204 show the time relations of the service primitives:

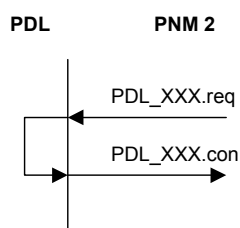


Figure 203 – Reset, Set Value and Get Value PDL services

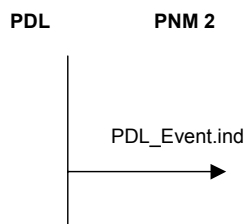


Figure 204 – Event PDL service

32.3.4.4 Detailed definition of the services and interactions

32.3.4.4.1 PDL_Reset

The PDL_Reset service is mandatory. The PNM2 transmits a PDL_Reset.request primitive to reset the PDL protocol machine (see Table 250).

Table 250 – PDL_Reset

Parameter name	Request	Confirm
Argument	M	
Result(+)		M

32.3.4.4.2 PDL_Set_Value

The PDL_Set_Value service is optional. The PNM2 transfers a PDL_Set_Value.request primitive to the PDL to set a defined PDL variable with a desired value. After receipt of this primitive, the PDL tries to select the variable and to set the new value. Upon execution, the PDL transfers a PDL_Set_Value.confirm primitive to the PNM2 (see Table 251).

Table 251 – PDL_Set_Value

Parameter name	Request	Confirm
Argument	M	
variable_name	M	
desired_value	M	
Result(+)		M

variable_name:

This parameter defines the PDL variable which is set to a new value.

desired_value:

This parameter declares the new value for the PDL variable.

Table 252 provides information on which PDL variable may be set to which new value.

Table 252 – PDL variables

Name of PDL variable	Value range	Default
max_spa_retry	0,2,4,6, ... 14	14
max_swa_count	0 ... 255	128
add_wait	1, 2, 3, 4	4
start_bus_cycle	ON, OFF	OFF
trigger_mode	network_triggered (NT), application_triggered (AT)	NT
max_dlsdu_size_from_req	1 ... 256 (see note)	256
max_dlsdu_size_from_res	1 ... 256 (see note)	256
max_receiving_queue_depth	1 ... 256 (see note)	256
max_sending_queue_depth	1 ... 256 (see note)	256
NOTE Only for PDL_Data_Ack services and each link		

32.3.4.4.3 PDL_Get_Value

The PDL_Get_Value service is optional. The PNM2 transfers a PDL_Get_Value.request primitive to the PDL to read out the current value of a defined PDL variable. After the PDL has received this primitive, it tries to select the defined variable and to transfer its current value to the PNM2 by means of a PDL_Get_Value.confirm primitive (see Table 253).

Table 253 – PDL_Get_Value

Parameter name	Request	Confirm
Argument variable_name	M M	
Result(+) current_value		M M

variable_name:

This parameter defines the PDL variable, whose value should be read.

current_value:

This parameter contains the desired value of this PDL variable.

Only those PDL variables can be read, which can also be written by the service PDL_Set_Value.request.

32.3.4.4.4 PDL_Event

The PDL_Event service is mandatory. The PDL transfers a PDL_Event.indication primitive to the PNM2 to inform it about detected events or errors in the PDL (see Table 254).

Table 254 – PDL_Event

Parameter name	Indication
Argument event	M M

event:

This parameter defines the value of the detected event or error cause in the PDL according to Table 255:

Table 255 – Events

Name	Meaning	Mandatory/optional
PDL_cycle_end	The receive update memory was updated, and the contents of the transmit update memory are transmitted to the BLL	O

32.3.5 Data transfer procedures from a queue

32.3.5.1 Bus access and data transfer mechanism

32.3.5.1.1 Synchronization cycle

Before starting of data transfer between the master and the slave(s), the PDL layers on all devices shall start with a synchronization cycle. In this cycle, a synchronization message resets the frame count bit flags in all devices to a defined value. In addition, the master started with the transmit of configure data to all slaves. After receiving the new configure data, all slaves shall initialize themselves with the new received configure values.

The frame count bits prevent a multiplication of messages at the confirming and/or responding device (responder), as these would cause the loss of positive acknowledgements.

A synchronization cycle only takes place for one communication relationship, that is, between the PDL protocol machine of the master and the PDL protocol machine of a slave. A synchronization cycle is initiated in the following cases:

- after a hardware reset,
- after a reset of the PDL layer by the PDL-user,
- after the detection of protocol errors,
- after a multiple data cycle error (`max_swa_count` time expired), and
- after a multiple `SPA_acknowledge_timeout` (the SPA acknowledge timeout occurred `max_spa_retry-times`).

In the first two cases the buffers and queues of the protocol layer for sending and receiving of messages are cleared from the concerned devices. Thus, all requests, confirmations and indication stored in these buffers are lost. In the remote device, however, no buffers are cleared. After the synchronization cycle this device tries to transmit the interrupted send message again.

In all other cases no buffers are cleared in any device. Upon a successful synchronization, both devices re-try to carry out orders of the application which have not yet been completed.

32.3.5.1.2 SVA message

Upon a successful synchronization and before interrupted messages are sent again, the master sends a SVA Message ("Send Value with Acknowledge") to the slave. The SVA Message transfers variables for the parameterisation of the PDL protocol machine.

The SVA Message transmits `max_swa_count`. The `max_swa_count` variable has a default value of 128 and can be parameterized by means of `PDL_Set_Value`. The slave accepts this value as its own `max_swa_count`.

The `max_swa_count` variable shall be transferred. In addition, other variables may be specified.

32.3.5.1.3 Frame count bit

The frame count bit (FCB) prevents a multiplication of messages at the confirming and/or responding device (responder), as this would cause the loss of positive acknowledgements.

If a positive acknowledgement is lost for whatever reason, the requester tries to send the previous message again. When this message has already been correctly received by the responder, this is indicated by an unchanged FCB. In this case the responder again sends the acknowledgement to the requester, directly after the receipt of the first message segment. Then, the requester stops the repeated sending.

If a new message is to be sent the FCB shall be changed. To ensure that the requester FCB (transmit FCB) and the responder FCB (receive FCB) of the remote device have the same initial value after the initialization of the layer 2 and after protocol errors, there will be a synchronization with `FCB_SET` messages. The FCBs are set to '1' if the synchronization was successful.

There is a FCB pair for both transmission directions (one transmit and one receive FCB for each direction) (see Figure 205).

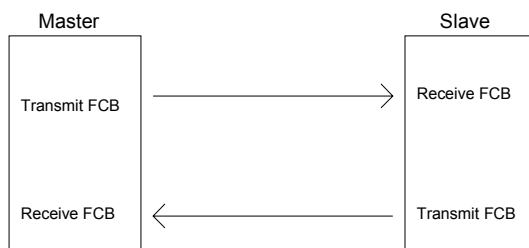


Figure 205 – Transmit and receive FCBs on the master and slave sides

32.3.5.1.4 Data transmission of bus cycles with errors

If a data cycle error occurs during the transmission of a SPA or SVA Message, the queue is not completely transmitted again. The transmission is rather continued with the queue parts that follow the error.

The master responds to cycle errors. Thus, a distinction is to be made between the two transmission directions **master → slave** and **slave → master**. If a cycle error occurs, this does not have any influence on the PDL protocol machine.

1) Data transmission: master → slave

If the master detects a data cycle error while queue is transmitted, the transmission shall be repeated from the error onwards. In this case the master communicates the error to the slave by means of a SWA Message.

Figure 206 and Figure 207 clarify the transmission master → slave with SWA Message. The numbering corresponds to the time sequence:

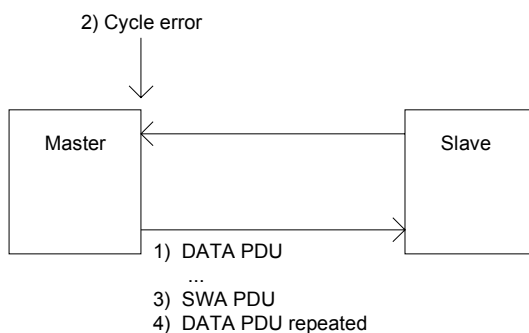


Figure 206 – Data transmission master → slave with SWA Message

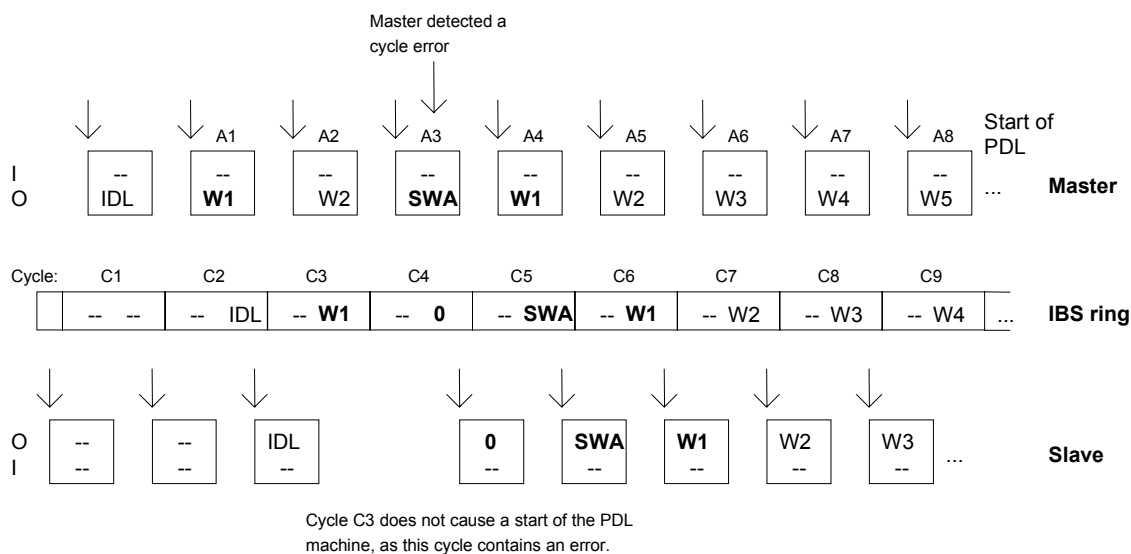


Figure 207 – Time sequence of the data transmission master → slave with SWA Message

2) Data transmission: slave → master:

If the master detects a cycle error when it receives a PDU, the slave will be announced immediately from the master by means of a RWA PDU. The slave shall confirm this RWA PDU with a SWA PDU, before the DATA PDU is sent again. The master uses the SWA PDU to mark the beginning of repeated data transmission.

An outstanding data transmission sequence from master → slave is interrupted during the RWA Message transfer and after the exception handling the data transmission can be continued.

Figure 208 and Figure 209 clarify the transmission slave → master with RWA Message or SWA Message:

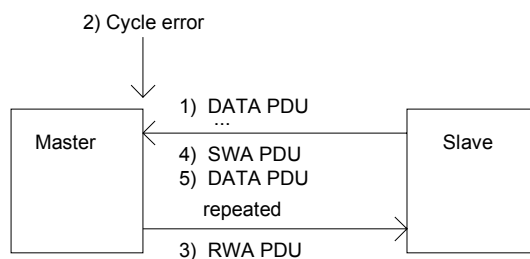


Figure 208 – Data transmission slave → master with SWA/RWA Message

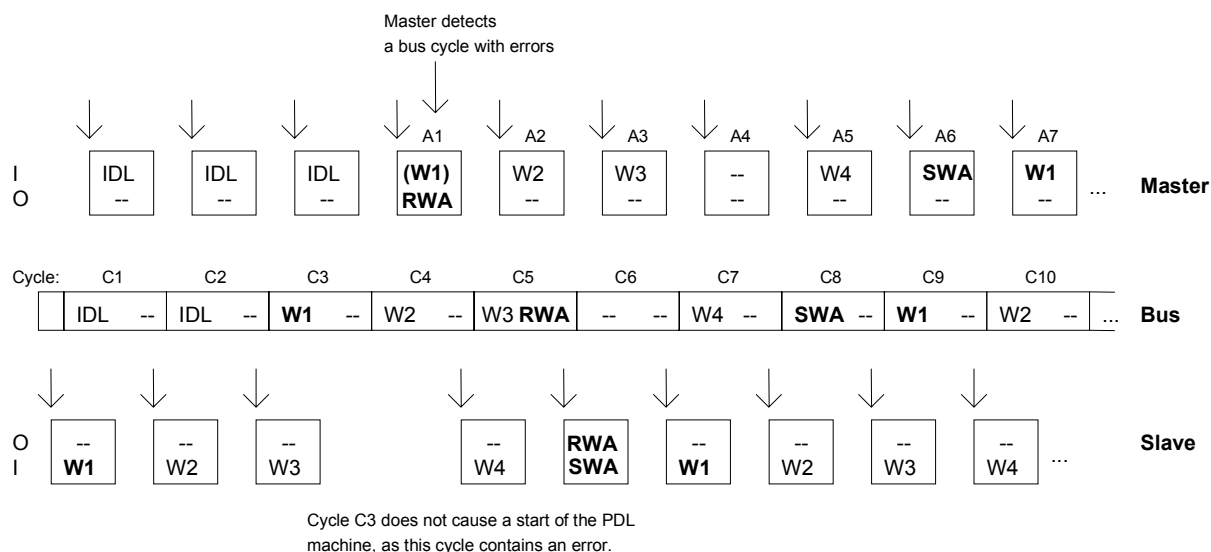


Figure 209 – Time sequence of the data transmission slave → master with SWA/RWA Message

32.3.5.2 Description of the time sequences

Master:

After each data cycle the PDL protocol machine is started once in the master for each slave in the ring having a parameter channel.

Among others, the protocol machine knows the following parameters:

Input parameters:

- The message segment which has been received during the last intact data cycle from the slave.
- The information whether a bus cycle error occurred during the last data cycle.

Output parameters:

- The message segment which is to be sent in the next cycle to the slave.

In Figure 210 these parameters are shown for each PDL protocol call A1...An, where I0...In identify the receive data for the master and the send data for the slave. Accordingly, O0...On are send data of the master and receive data of the slave.

Slave:

After the completion of a data cycle the PDL protocol machine is started on the slave, if the slave determined that the master did not send an IDL PDU and/or there is an outstanding send data request on the slave. IDL PDU are transmitted whenever there is no further user data are to be transmit. The last received message can be read and/or one outstanding send message can be prepared in the PDL protocol machine. The PDL pass the PDL PDU to the BLL for transmitting within the next data cycle to the master. The third line of the representation shows the starts of the PDL protocol machine of the slave (A1...A9) and the associated send and receive messages.

Bus:

In Figure 210, the middle row shows the cycles (C1...C9) and the messages which are transmitted in these cycles from the slave to the master and vice versa.

The transmission of a message from the TRANSMIT protocol machine of the master to the RECEIVE protocol machine of the slave requires two cycles, and the transmission from the TRANSMIT protocol machine of the slave to the RECEIVE protocol machine of the master requires three cycles. The TRANSMIT and RECEIVE protocol machines are components of the PDL protocol machine.

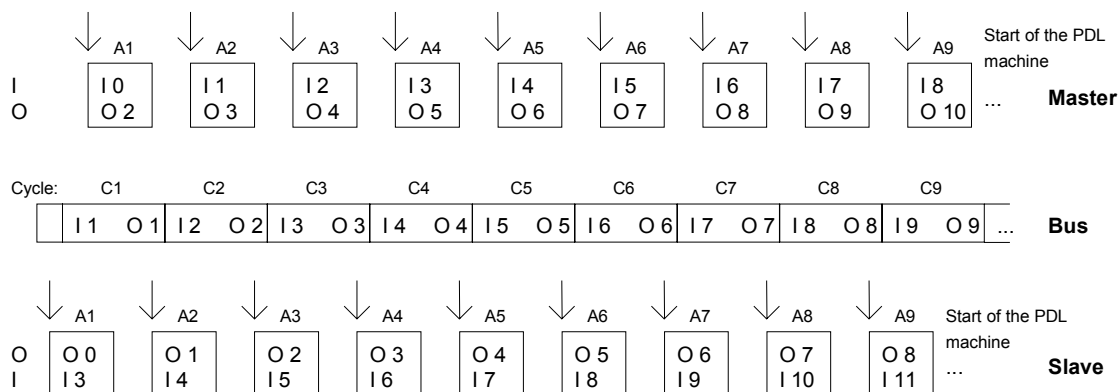


Figure 210 – Allocation of actions of the PDL protocol machines and data cycles

In the slave, the start of the PDL protocol machine for the sending and receiving of PDL PDU shall be completed before a further cycle end was indicated. Otherwise received data can be lost. The DLPDU cycle time depends with the respect from the number of slaves and from the data width of each slave which are connected to the DL-subnetwork.

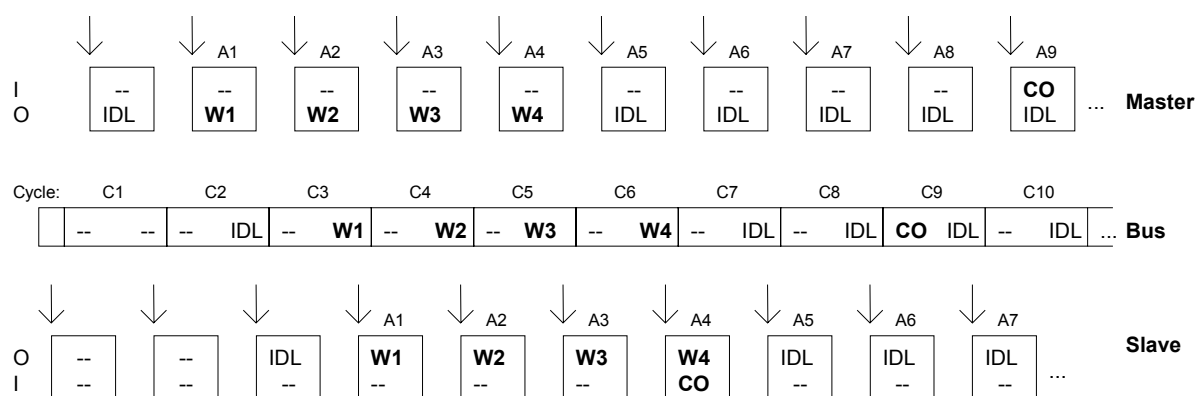


Figure 211 – Message transmission: master → slave

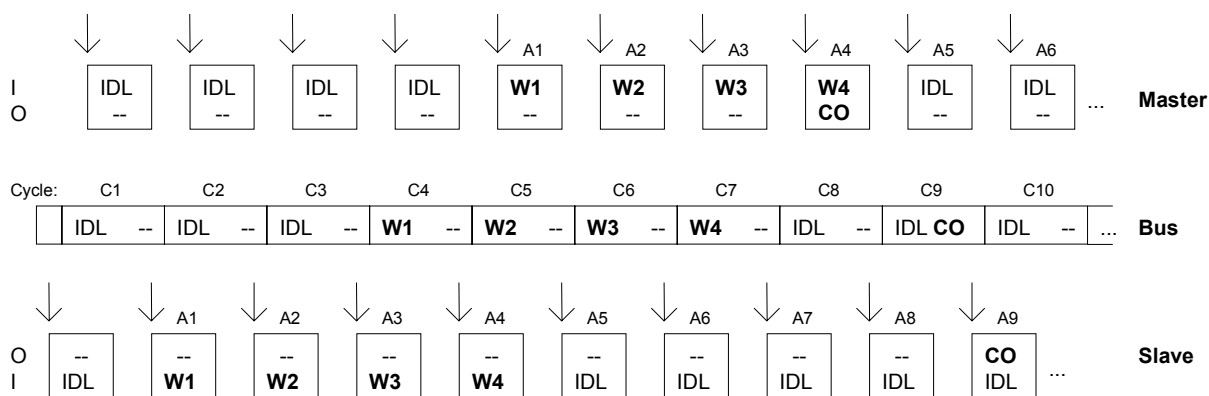


Figure 212 – Message transmission: slave → master

Figure 211 and Figure 212 show that at least DIST = 5 data cycles are needed between the sending of the last message, until a confirmation (CO) for this message is received.

Delays while confirmations may be also taken into the account, so the protocol need additional cycles for waiting of a confirmation. This number of additional cycles is stored in the variable "add_wait". On the master side the variable add_wait can be parameterized with the PDL_Set_Value (value range: 1 to 4). The contents of the variable add_wait is transmitted from the master to the slave within the FCB_SET PDU while the PDL protocol machines are synchronized.

32.3.5.3 Coding of the messages

32.3.5.3.1 Overview

The message length for a slave with the parameter channel consists at least one octet for FCT and additional the length of the data 1, 3 or 7 octets, depending from the size of the message for transmission.

32.3.5.3.2 Structure of the code octet

32.3.5.3.2.1 General

Figure 213 shows the structure of a code octet, Table 256 shows the L_status encoding and Table 257 the FCT code.

B16	B15	B14	B13	B12	B11	B10	B9
L_status			FCT code			FCB	IDL

Figure 213 – Code octet of a PDL PDU

Table 256 – Encoding of the L_status

L_status (B16 B15 B14)	Generation (Local/Remote)	Meaning
0 0 0	--	No confirmation
0 0 1	L / R	Acknowledgement positive
0 1 0	R	Acknowledgement negative, no resource available (buffer full)
0 1 1	L / R	Acknowledgement negative, multiple data cycle error (transmission via the bus only from the master to the slave)
1 0 0	R	Acknowledgement positive (repeated)
1 0 1	R	Acknowledgement negative, no resource available (repeated)
1 1 0	L	Acknowledgement negative, acknowledge_timeout
1 1 1	R	FCB_SET confirmation, the FCT code equals 0; FCB = 1!

Table 257 – FCT code (PDL PDU-Types)

Function code	PDL-PDU Types	Meaning
0 0 0	IDL PDU	The IDL PDU do not contain any information (except for a FCB_SET confirmation).
0 0 1	DATA PDU	The DATA PDU contains segments of user data or PDL variables.
0 1 0	SPA PDU	The SPA PDU defines the start of a new queue data transmission and contains at least the length information of DLSDU in octets-1 and segments of user data.
0 1 1	SVA PDU	The SVA PDU defines the start of a new variable data transmission and contains at least the length information of variable data in octets-1 and possibly PDL variables.
1 0 0	Don't spare	Reserved
1 0 1	RWA PDU	RWA PDU (Receive Word Again): The data octets of the message contain at least the amount of successfully received data octets+1 and possibly DLSDU data or PDL variables.
1 1 0	SWA PDU	SWA PDU (Send Word Again): The data octets of the message contain at least the amount of successfully sent data octets + 1 and possibly DLSDU data or PDL variables.
1 1 1	FCB_SET PDU	The FCB_SET PDU contains L_status = 0 and FCB = 1, for an FCB_SET request from the master to the slave. The first data octet of the message contains user data for the PDL protocol machine of the slave.

32.3.5.3.2.2 FCB

Frame count bit : 0/1, alternating bit.

The FCB is only relevant within the start segment of SPA Message and SVA Messages.

32.3.5.3.2.3 Idle message

If the IDL-Bit (see Figure 213; Bit 9) equals 0, the message does not contain any information and is called IDLE message.

32.3.5.3.3 Messages with a size of one word**32.3.5.3.3.1 General**

Figure 214 shows the structure of a message with a size of one word.

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	X	X	X	X	0	X	X	X	X	X	X	X	X

Figure 214 – Structure of a message with a size of one word**32.3.5.3.3.2 Call messages****1) SPA Message (PDL PDU-Type for transmission of queued user data):**

In the L_status of the code octets, a confirmation for a SPA Message of the remote device can be transmitted at the same time (see Figure 215).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	0	1	0	FC B	1	n-1: DLSDU length in octets-1: $1 \leq n \leq 256$							
X	X	X	0	0	1	X	1	1st DLSDU octet							
X	X	X	0	0	1	X	1	2nd DLSDU octet							
X	X	X	0	0	1	X	1	Nth DLSDU octet							

Figure 215 – Structure of a SPA Message

2) SVA Message (DLPDU-Type for transmission of configuration data):

After a confirmed FCB_SET, further variables are transmitted from the master to the slave with the 'Send Value with Acknowledge' message. On the master side these variables can be set with the PDL_Set_Value.request and are also valid for the slave after the transmission of the SVA Message. The number of variable octets-1 is transmitted in the data octet of the start segment.

The swa_count variable shall be transferred. In addition, further variables may be specified (see Figure 216).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	0	1	1	FC B	1	n-1: number of variable octets-1							
X	X	X	0	0	1	X	1	Swa_count (mandatory)							
X	X	X	0	0	1	X	1	2 nd variable octet (optional)							
...															
X	X	X	0	0	1	X	1	Nth variable octet (optional)							

Figure 216 – Structure of a SVA Message

3) FCB_SET Message

This message is used to synchronize the PDL protocol machines of a parameter channel on the master and on the slave, that is, to set the FCBs to a defined value (see 32.3.5.1.3).

In the first data octet, the master also transmits the protocol specific parameters which have been set with the PDL_Set_Value service (see Figure 217).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
0	0	0	1	1	1	1	1	Coding see below							

Figure 217 – Structure of a FCB_SET Message

32.3.5.3.3.3 Control segments

1) RWA Message:

With the RWA (Read Word Again) message the master indicates to the slave during a SPA Message that the slave has received one message without errors. In the first data octet the RWA Message contains the number of DLSDU octets which have been received without errors (see Figure 218).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	1	0	1	X	1	k Number of correctly received DLSDU octets during the current SPA Messages							

Figure 218 – Structure of a RWA Message

2) **SWA Message:**

For the SWA Message a difference between master and slave has to be observed, as only the master can immediately detect a bus cycle with errors (see Figure 219):

- When a SPA Message is sent from the slave to the master, it identifies the beginning of repeated data transmission. Here, the SPA Message is sent after a RWA Message.
- If the master detects a data cycle with errors when a SPA or SVA Message is sent to the slave, the transmission is repeated from the error onwards. This error-1 = number of PDU sent without errors is communicated to the slave by means of a SWA Message.

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	1	1	0	X	1	k Number of data octets sent without errors							
X	X	X	0	0	1	X	1	(k+1)th data octet							
X	X	X	0	0	1	X	1	(k+2)th data octet							
...															
X	X	X	0	0	1	X	1	N-th data octet							

Figure 219 – Structure of a SWA Message

32.3.5.3.3.4 **Response messages**1) **Data confirmation** (confirmation for SPA or SVA Messages)

The higher three bits of the code octet contain the confirmation for the call messages described above. This confirmation can also be transmitted together with a message start segment of SPA or SVA, control or user data segment (exception: FCB_SET PDU) (see Figure 220).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
L_status			X	X	X	X	1	X	X	X	X	X	X	X	X

Figure 220 – Structure of a confirmation for SPA or SVA Messages

2) **FCB_SET confirmation**

The FCB_SET confirmation acknowledges a FCB_SET request (see Figure 221).

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
1	1	1	0	0	0	1	1	Coding see below							

Figure 221 – Structure of a FCB_SET as confirmation

Coding of the data octet for FCB_SET Message as request and confirmation

In the data octet of the FCB_SET as request or the FCB_SET as confirmation, the master transmits the protocol specific parameters which have been set with the PDL_Set_Value service (see Figure 222):

Code octet								Data octet							
B16	B15	B14	B13	B12	B11	B10	B9	B8	B7	B6	B5	B4	B3	B2	B1
X	X	X	X	X	X	X	X	a)			b)			c)	

Figure 222 – Structure of the data octet for FCB_SET as requests and confirmations

a) **B8-B6: err_max**

The contents of the variable err_max should be multiplied with 2, accordingly to get the number of attempts for sending a same message

b) **B5-B3: Reserved**

c) **B2-B1: add_wait**

The variable add_wait-1 gives the number of cycles that is additionally waited for a confirmation.

32.3.5.3.4 Queue data with a segment size of more than one word

Example: Queue data size = 2 words, data transmission with SWA Message after a cycle error (see Figure 223).

Code octet				1st data octet		2nd data octet		3rd data octet							
X	X	X	1	1	0	X	1	k number of message octets sent without an error		(k+1)th message octet		(k+2)th message octet			
X	X	X	0	0	1	X	1	(k+3)th message octet		(k+4)th message octet		(k+5)th message octet			
X	X	X	1	1	0	X	1	(N-1)th message octet		N-th message octet		X	X	X	X

Figure 223 – Structure of a message with a size of more than one word

Number of segments to be transmitted for a message (DLSDU):

Calculation: $G_m(N) = (N-1)/m + 1$

where

N is the user data quantity (including the number of octets sent/received without errors or the data length):

M is the number of data octets per segment (PDU)

G_m(N) is the number of PDU

NOTE (N-1)/m is a not rounding integer division

Example:

A queue data with 13 octets is to be transmitted. $N = 14$, including the data length. The segment size is two words, that is, three data octets are available in the segment ($m = 3$):

$$G_3(14) = (14-1)/3 + 1 = 4 + 1 = 5$$

32.3.6 PDL protocol machines

32.3.6.1 PDL base protocol machine

32.3.6.1.1 Description of the states

The PDL base protocol machine provides the functionality of the PDL and has the following five states:

32.3.6.1.2 PDL_INIT

State after power on. This state is only left when all required PDL protocol machines have been generated by means of the communication relationship list entries and the PDL receive and transmit update memories are initialized.

32.3.6.1.3 PDL_RECEIVE_UPDATE

In this state the PDL takes the user data out of the PDLSDU which was received from the BLL and updates the PDL receive update memory of layer 2.

32.3.6.1.4 PDL_PM_ACTIVE

With the transition into this state all protocol machines are initiated at the same time. This state shall only be left after all protocol machines have been processed.

32.3.6.1.5 PDL_TRANSMIT_UPDATE

In this state the PDL takes the user data out of the transmit update memory of layer 2, generates the user data part of the PDLSDU to be sent to the BLL and transmits the PDLSDU with a BLL_Data.request (master side) or BLL_Data.response (slave side) primitive to the BLL.

32.3.6.1.6 PDL_WAIT_FOR_PDLSDU

In this state the PDL base protocol machine (see Figure 224) waits for a BLL_Data.confirm(master side) or BLL_Data.indication (slave side) primitive of the BLL.

In all states (except PDL_INIT) the PDL_Data_Ack.request primitives, which are passed from the PDL-user to the PDL, are processed as follows:

First, the service request is locally confirmed by means of a PDL_Data_Ack.confirm primitive. Then at the receiving side, the PDL generates a PDL_Data_Ack.indication primitive that is passed on to the PDL-user, which can be identified by the remote address.

A reset generally causes a transition to the PDL_INIT state.

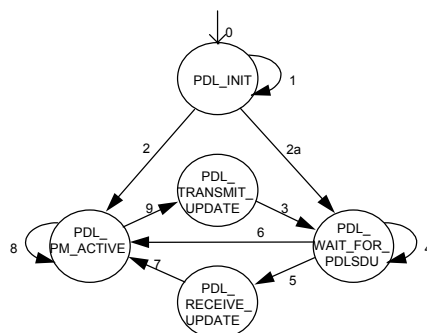


Figure 224 – PDL base protocol machine

The state transitions from every state (except PDL_INIT) to the same state when a PDL_Data_Ack.request primitive arrives from the PDL-user are not shown to simplify matters. However, they are listed in the state transition table.

32.3.6.1.7 Description of the transitions

Table 258 shows the state transitions of the PDL base protocol machine.

Table 258 – State transitions of the PDL base protocol machine

Initial state event \ condition ⇒ action	Transition	Follow-up state
After Power on	0	PDL_INIT
PDL_INIT \ initialization of the PDL protocol machines and the update memories are not yet completed	1	PDL_INIT
PDL_INIT (master side only) \ initialization completed ⇒ start of all PDL protocol machines	2	PDL_PM_ACTIVE
PDL_INIT (slave side only) \ initialization completed	2a	PDL_WAIT_FOR_PDLSDU
PDL_TRANSMIT_UPDATE ⇒ copy of user data from TUM to the PDLSDU AND send PDLSDU to BLL with BLL_Data.request/response	3	PDL_WAIT_FOR_PDLSDU
PDL_WAIT_FOR_PDLSDU \ no BLL_Data.confirm/indication received	4	PDL_WAIT_FOR_PDLSDU
PDL_WAIT_FOR_PDLSDU BLL_Data.confirm/indication received \ update_info == OK	5	PDL_RECEIVE_UPDATE
PDL_WAIT_FOR_PDLSDU BLL_Data.confirm received \ update_info == NOK ⇒ start of all PDL protocol machines ⇒ if trigger_mode == AT, then start_bus_cycle = OFF	6	PDL_PM_ACTIVE
PDL_RECEIVE_UPDATE ⇒ copy user data from PDLSDU to RUM ⇒ start all PDL protocol machines ⇒ if trigger_mode == AT, then start_bus_cycle = OFF	7	PDL_PM_ACTIVE
PDL_PM_ACTIVE \ not all PDL protocol machines are stopped OR (<i>master side only</i>) start_bus_cycle == OFF	8	PDL_PM_ACTIVE

Initial state event \ condition ⇒ action	Transition	Follow-up state
PDL_PM_ACTIVE \ all PDL protocol machines are stopped AND (<i>master side only</i>) start_bus_cycle == ON	9	PDL_TRANSMIT_UPDATE
Any_state PDL_Reset.request received ⇒ PDL_Reset.confirm		PDL_INIT
Any_state PDL_Data_Ack.request (...rem_add,...) ⇒ PDL_Data_Ack.confirm ⇒ PDL_Data_Ack.indication to PDL-user with rem_add		same_state
Any_state all PDL management services ⇒ process and confirm the services		same_state

32.3.6.2 PDL protocol machine

32.3.6.2.1 Overview

For each slave with a parameter channel in the ring, the PDL base protocol machine of the master manages a PDL protocol machine. The PDL base protocol machine of a slave with a parameter channel, however, has only one PDL protocol machine for this parameter channel.

A PDL protocol machine processes all PDL_Data_Ack services which are transmitted via the parameter channel to the PDL and has, in particular, the following tasks:

Connection:

- Establishing the PDL connection to the remote device

Sending side:

- Testing the send requests (PDL_Data_Ack.request) for plausibility
- Process the send requests
- Confirming the send requests (PDL_Data_Ack.confirm)
- Segmentation of the DLSDU to data segments
- Error detection while data transfer
- Passing the Message segments and confirmations with the BLL_Data.request/response to the Basic Link Layer (BLL)

Receiving side:

- Receiving confirmations and data segments with the BLL_Data.confirm
- Re-assembling the data segments to DLSDU
- Transferring received DLSDU to the PDL-user with PDL_Data_Ack.indication

Figure 225 explains the PDL protocol machine in general for a master and a slave. The differences between master and slave are given. Each PDL protocol machine controls and uses a TRANSMIT and a RECEIVE protocol machine.

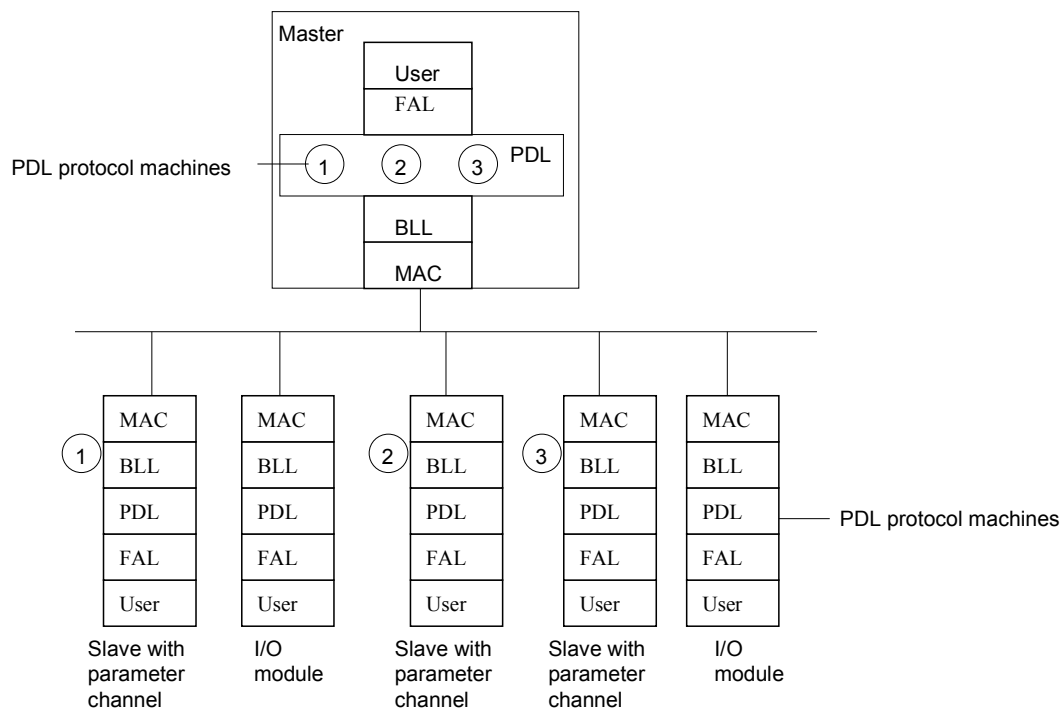


Figure 225 – Locations of the PDL and the PDL protocol machines in the master and slaves

32.3.6.2.2 Counters and flags in the protocol machines

32.3.6.2.2.1 Counters

The counters described in the following are used in the PDL protocol machines for the parameter channel (see Table 259):

Table 259 – Counters of the PDL protocol machines

Counter	Description	Used in protocol machine
Ccycle	Counter for the number of data cycles since the start signal of a SPA or SVA Message has been sent. This counter checks the correctness of repeated confirmations.	TRANSMIT
Ccerr	Counter for the number of defective data cycles after another while a message is sent.	TRANSMIT
Cconf	Counter for the number of cycles during which a confirmation is awaited.	PDL protocol and TRANSMIT
Creq_retry	Counter for the attempts to send messages when confirmations were lost due to defective data cycles.	TRANSMIT
CSWA	Counter for the number of cycles during which a SWA Message is awaited.	RECEIVE

32.3.6.2.2.2 Flags

connection

The PDL protocol machine manages a flag called "connection". This flag is the same as DISCONN, when the PDL protocol machine of the master is not synchronized with that of a slave. In this case, the master or slave continuously sends synchronization messages until one message is confirmed by the remote device. Then connection == READY is set, that is, the PDL protocol machines are synchronized (see Table 260).

Table 260 – Meaning of the "connection" flag

Connection	Description
DISCONN	The PDL protocol machines are not synchronized.
READY	The PDL protocol machines are synchronized. The PDL layer is ready to transmit a message.

32.3.6.2.3 Description of the states

32.3.6.2.3.1 Overview

Figure 226 shows the PDL protocol machine.

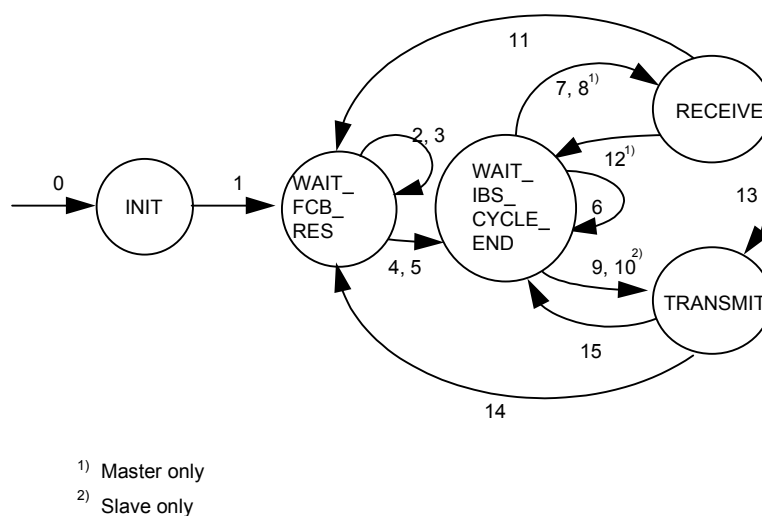


Figure 226 – PDL protocol machine

Transitions as a result of a received PDL_Reset.request primitive always go from every state to the INIT state. These transitions are not shown individually but are described by the transition 0.

32.3.6.2.3.2 INIT

Initialization of the PDL protocol machine

32.3.6.2.3.3 WAIT_FCB_RES

Synchronization

32.3.6.2.3.4 WAIT_IBS_CYCLE_END

The PDL protocol machine is synchronized. This state controls whether the RECEIVE and/or the TRANSMIT protocol machine is started.

32.3.6.2.3.5 RECEIVE

The RECEIVE protocol machine is started.

32.3.6.2.3.6 TRANSMIT

The TRANSMIT protocol machine is started.

32.3.6.2.4 Description of the transitions

The PDL protocol machine is started by the PDL base protocol machine when a data cycle has been completed and new received data (PDLSDU) from the BLL is available. If the received data has been processed, and new transmit data is not available, the PDL protocol machine stops itself. In the stopped state, the protocol machine does not respond to events. Only a PDL_Reset.request causes a reset of the protocol machine at any time (see Table 261).

Table 261 – State transitions of the PDL protocol machine

Initial state event condition ⇒ action	Transition	Follow-up state
PDL_Reset.request received ⇒ initialize PDL protocol machine, reject non-processed PDL_Data_Ack services	0 Reset	INIT
INIT initialization of the PDL protocol machine completed Cconf = 0, stop PDL protocol machine	1 TFCB_Req1	WAIT_FCB_RES
WAIT_FCB_RES Cconf ≤ DIST+add_wait neither FCB_SET request (slave side only) sent nor FCB_SET confirmation received ⇒ increment Cconf, stop PDL protocol machine	2 Wait1	WAIT_FCB_RES
WAIT_FCB_RES Cconf > DIST+add_wait neither FCB_SET request (slave side only) nor FCB_SET confirmation received ⇒ send FCB_SET request, Cconf = 0, stop PDL protocol machine	3 TFCB_Req2	WAIT_FCB_RES
WAIT_FCB_RES FCB_SET request received ⇒ send FCB_SET confirmation, set receive and transmit FCB-Fag, connection = READY, reset TRANSMIT and RECEIVE protocol machine, master side only: send enable of SVA PDU (see TRANSMIT protocol machine), stop PDL protocol machine	4 RFCB_Req1	WAIT_IBS_CYCLE_END

Initial state event condition ⇒ action	Transition	Follow-up state
WAIT_FCB_RES FCB_SET confirmation received ⇒ set receive and transmit FCB, connection = READY, reset TRANSMIT and RECEIVE protocol machine., master side only: send enable SVA PDU (see TRANSMIT protocol machine), stop PDL protocol machine	5 RFCB_Conf	WAIT_IBS_CYCLE_END
WAIT_IBS_CYCLE_END FCB_SET as request received ⇒ send FCB_SET as confirmation, set receive and transmit FCB, connection = READY, reset TRANSMIT and RECEIVE protocol machines, master side only: send enable SVA Message (see TRANSMIT protocol machine), stop PDL protocol machine	6 RFCB_Req2	WAIT_IBS_CYCLE_END
WAIT_IBS_CYCLE_END receipt of a confirmation and/or a message segments except FCB_SET request and FCB_SET confirmation, (slave side only), no RWA PDU ⇒ start RECEIVE protocol machine	7 Recv_Req	RECEIVE
WAIT_IBS_CYCLE_END (master side only) status of RECEIVE protocol machine == WAIT_SWA ⇒ start RECEIVE protocol machine	8 Wait_SWA	RECEIVE
WAIT_IBS_CYCLE_END receipt of an IDLE message ⇒ start TRANSMIT protocol machine	9 Send_Req	TRANSMIT
WAIT_IBS_CYCLE_END (slave side only) RWA PDU received ⇒ start TRANSMIT protocol machine (to send a SWA PDU)	10 Recv_RWA	TRANSMIT
RECEIVE RECEIVE protocol machine stopped AND connection == DISCONN ⇒ send FCB_SET request, Cconf = 0, stop PDL protocol machine	11 TFCB_Req3	WAIT_FCB_RES
RECEIVE (master side only) RECEIVE protocol machine stopped AND connection == READY AND RECEIVE protocol machine has sent RWA PDU ⇒ stop PDL protocol machine	12 TM_Disable	WAIT_IBS_CYCLE_END
RECEIVE RECEIVE protocol machine stopped AND connection == READY AND RECEIVE protocol machine (master side only) has not sent a RWA PDU ⇒ start TRANSMIT protocol machine	13 Recv_OK2	TRANSMIT
TRANSMIT TRANSMIT protocol machine stopped AND connection == DISCONN ⇒ send FCB_SET request, Cconf = 0, stop PDL protocol machine	14 TFCB_Req4	WAIT_FCB_RES
TRANSMIT TRANSMIT protocol machine stopped AND connection == READY ⇒ stop PDL protocol machine	15 Transm_OK	WAIT_IBS_CYCLE_END

32.3.6.3 TRANSMIT protocol machine

32.3.6.3.1 Description of the states

32.3.6.3.1.1 Overview

Figure 227 shows the TRANSMIT protocol machine.

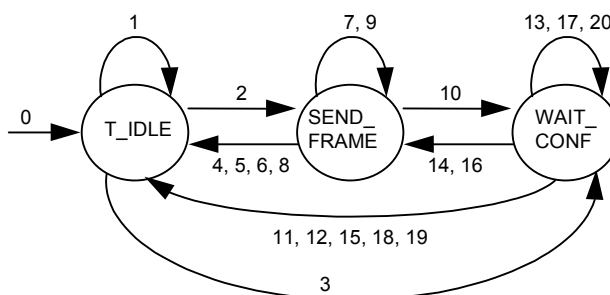


Figure 227 – TRANSMIT protocol machine

Transitions as a result of a PDL_Reset primitive always go from every state to the T_IDLE state. These transitions are not shown individually but are described by the transition 0.

32.3.6.3.1.2 T_IDLE

No message is being sent.

32.3.6.3.1.3 SEND_FRAME

A message is being sent.

32.3.6.3.1.4 WAIT_CONF

A message was sent. The confirmation of the remote device is being waited for.

32.3.6.3.2 Description of the transitions

The TRANSMIT protocol machine is started by the higher-level PDL protocol machine. Then the TRANSMIT protocol machine merely carries out one transition and stops itself. In the table which describes the state transitions the 'stop TRANSMIT protocol machine' action was not included for all transitions to simplify matters. In the stopped state the protocol machine does not respond to events. Only a PDL_Reset.request causes a reset of the protocol machine at any time (see Table 262).

Table 262 – State transitions of the TRANSMIT protocol machine

Initial state event condition ⇒ action	Transition	Follow-up state
PDL_Reset.request ⇒ reset TRANSMIT protocol machine	0 Reset	T_IDLE
T_IDLE there is no send enable for a SVA PDU (<i>master only</i>) and no PDL_Data_Ack.request ⇒ no action (waiting)	1 Wait1	T_IDLE
T_IDLE send enable for SVA PDU (<i>master only</i>) or PDL_Data_Ack.request PDU shall be transmitted with more than one segment ⇒ enter start segment of the SPA/SVA PDU, Ccycle = Ccerr = Creq_reply = 0	2 Request1	SEND_FRAME
T_IDLE send enable for SVA PDU (<i>master only</i>) or PDL_Data_Ack.request PDU can be transmitted with one segment ⇒ enter start segment of the SPA/SVA PDU, Ccycle = Ccerr = Creq_reply = 0, Cconf = 0	3 Request2	WAIT_CONF
SEND_FRAME repeated confirmation for SPA PDU received (repeated positive or repeated queue full) DIST ≤ Ccycle ≤ DIST+add_wait ⇒ change transmit FCB, sent SPA confirmation	4 Recv_Conf1	T_IDLE
SEND_FRAME repeated confirmation for SVA PDU received (repeated positive or repeated queue full) DIST ≤ Ccycle ≤ DIST+add_wait ⇒ change transmit FCB	5 Recv_Conf2	T_IDLE
SEND_FRAME repeated confirmation for SPA or SVA PDU received (repeated positive or repeated queue full) Ccycle < DIST OR Ccycle > DIST+add_wait ⇒ connection = DISCONN	6 Disconn1	T_IDLE
SEND_FRAME last data cycle contained errors Ccerr ≤ max_swa_count ⇒ increment Ccerr , enter SWA PDU, increment Ccycle	7 Cycl_Err1	SEND_FRAME
SEND_FRAME last data cycle contained errors Ccerr > max_swa_count ⇒ connection = DISCONN	8 Mult_Err1	T_IDLE
SEND_FRAME last data cycle completed without errors more than one data segment to be sent ⇒ enter DATA PDU, increment Ccycle	9 Send_Segm1	SEND_FRAME
SEND_FRAME last data cycle completed without errors last data segment to be sent ⇒ enter DATA PDU, Cconf = 0	10 Send_Segm2	WAIT_CONF

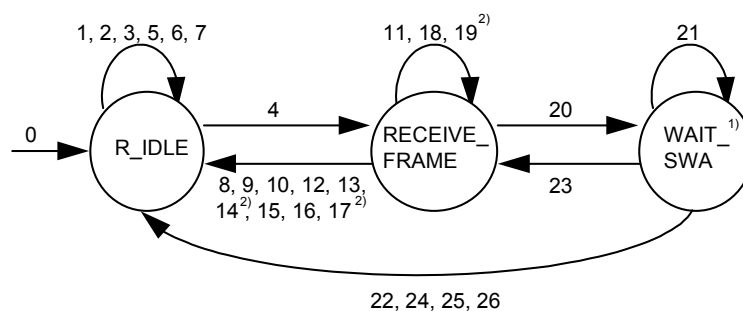
Initial state event condition ⇒ action	Transition	Follow-up state
WAIT_CONF confirmation SPA received (positive, repeated pos., queue full or repeated queue full) ⇒ change transmit FCB, enter SPA PDU	11 Recv_Conf3	T_IDLE
WAIT_CONF confirmation for SVA PDU received (positive, repeated pos., queue full or repeated queue full) ⇒ change transmit FCB	12 Recv_Conf4	T_IDLE
WAIT_CONF no confirmation received Cconf > DIST+add_wait AND Creq_retry ≤ max_req_retry AND queued data can be transmitted with one segment ⇒ increment Creq_retry , enter start segment of SPA or SVA PDU, Ccycle = Ccerr = 0, Cconf = 0	13 Timeout1	WAIT_CONF
WAIT_CONF no confirmation received Cconf > DIST+add_wait AND Creq_retry ≤ max_req_retry AND queued data shall be transmitted with more than one segment ⇒ increment Creq_retry , enter start segment of the SPA or SVA PDU, Ccycle = 0, Ccerr = 0	14 Timeout2	SEND_FRAME
WAIT_CONF no confirmation received Cconf > DIST+add_wait AND Creq_retry > max_req_retry ⇒ connection = DISCONN	15 Mult_TO	T_IDLE
WAIT_CONF last data cycle contained errors Ccerr ≤ max_swa_count AND more than one data segment is to be sent repeatedly ⇒ increment Ccerr , enter SWA PDU	16 Cycle_Err2	SEND_FRAME
WAIT_CONF last data cycle contained errors Ccerr ≤ max_swa_count AND SWA PDU can accept all data which is to be sent repeatedly ⇒ increment Ccerr, enter SWA PDU in PDLSDU, Cconf = 0	17 Cycle_Err3	WAIT_CONF
WAIT_CONF last data cycle contained errors Ccerr > max_swa_count ⇒ connection = DISCONN	18 Mult_Err2	T_IDLE
WAIT_CONF repeated confirmation for SPA or SVA PDU received (repeated positive or repeated queue full) Ccycle < DIST OR Ccycle > DIST+add_wait ⇒ connection = DISCONN	19 Disconn2	T_IDLE
WAIT_CONF no queue received Cconf ≤ DIST+add_wait ⇒ increment Cconf and Ccycle	20 Wait2	WAIT_CONF

32.3.6.4 RECEIVE Protocol Machine

32.3.6.4.1 Description of the states

32.3.6.4.1.1 Overview

Figure 228 shows the RECEIVE protocol machine.



- 1) The WAIT_SWA state as well as the transitions from and to WAIT_SWA are for the master only
 2) For a slave only

Figure 228 – RECEIVE protocol machine

Transitions as a result of PDL_Reset.request primitive always go from every state to the R_IDLE state. These transitions are not shown individually but are described by the transition 0.

32.3.6.4.1.2 R_IDLE

No message is being received.

32.3.6.4.1.3 SEND_FRAME

A message is being received.

32.3.6.4.1.4 WAIT_SWA (master side only)

A RWA PDU was sent. The responding SWA PDU is being waited for.

32.3.6.4.2 Description of the transitions

The RECEIVE protocol machine is started by the higher-level PDL protocol machine. Then, the RECEIVE protocol machine merely carries out one transition and stops itself. In Table 263 which describes the state transitions, the stop 'RECEIVE protocol machine' action was not included for all transitions to simplify matters. In the stopped state the protocol machine does not respond to events. Only a PDL_Reset.request causes a reset of the protocol machine at any time.

Table 263 – State transitions of the RECEIVE protocol machine

Initial state event condition ⇒ action	Transition	Follow-up state
PDL_Reset.request received ⇒ reset RECEIVE protocol machine	0 Reset	R_IDLE
R_IDLE no SVA (<i>slave only</i>) or SPA PDU received ⇒ no action (waiting)	1 Wait1	R_IDLE
R_IDLE SVA (<i>slave only</i>) or SPA PDU received receive FCB ≠ await FCB AND no confirmation has yet been sent for this receive FCB ⇒ connection = DISCONN	2 Disconn1	R_IDLE
R_IDLE SVA (<i>slave only</i>) or SPA PDU received receive FCB ≠ await FCB AND a confirmation has already been sent for this receive FCB ⇒ enter repeated confirmation in PDLSDU (repeated positive or repeated queue full)	3 Rep_Recept1	R_IDLE
R_IDLE SVA (<i>slave only</i>) or SPA PDU received start segment does not contain the complete message ⇒ accept data octets	4 RStart_Sgm1	RECEIVE_FRAME
R_IDLE SPA PDU received start segment does not contain a complete message AND memory available ⇒ accept data octets, enter positive confirmation in PDLSDU, change receive FCB, PDL_Data_Ack.indication to the PDL-user	5 RStart_Sgm2	R_IDLE
R_IDLE SPA PDU received start segment contains complete message AND no more memory available ⇒ enter negative confirmation (queue full) in PDLSDU, change receive FCB	6 RStart_Sgm3	R_IDLE
R_IDLE (slave only) SVA PDU received start segment contains complete message ⇒ accept data octets, enter positive confirmation in PDLSDU, change receive FCB	7 RStart_Sgm4	R_IDLE
RECEIVE_FRAME no DATA PDU, no SWA PDU and no SVA (<i>slave only</i>) or SPA PDU received ⇒ stop the sending of a message	8 Segm_Err1	R_IDLE
RECEIVE_FRAME SVA (<i>slave only</i>) or SPA PDU received receive FCB ≠ FCB in start segment AND no confirmation was sent for the received PDU ⇒ connection = DISCONN	9 Disconn2	R_IDLE
RECEIVE_FRAME SVA (<i>slave only</i>) or SPA PDU received receive FCB ≠ await FCB in start segment AND a confirmation has already been sent for the PDU ⇒ enter repeated confirmation in PDLSDU (repeated positive or repeated queue full)	10 Rep_Recept2	R_IDLE

Initial state event \condition ⇒ action	Transition	Follow-up state
RECEIVE_FRAME SVA (<i>slave only</i>) or SPA PDU received \receive FCB == await FCB in start segment AND start segment does not contain the complete message ⇒ accept data octets	11 RStart_Sgm5	RECEIVE_FRAME
RECEIVE_FRAME SPA PDU received \receive FCB == await FCB in start segment AND start segment contains a complete message AND memory available ⇒ accept data octets, enter positive confirmation in PDLSDU, change receive FCB, PDL_Data_Ack.indication to the PDL-user	12 RStart_Sgm6	R_IDLE
RECEIVE_FRAME SPA PDU received \receive FCB == await FCB in start segment AND start segment contains a complete message AND no more memory available ⇒ enter negative confirmation (queue full) in PDLSDU, change receive FCB	13 RStart_Sgm7	R_IDLE
RECEIVE_FRAME (<i>slave only</i>) SVA PDU received \receive FCB == await FCB in start segment AND start segment contains a complete message ⇒ accept data octets, enter positive confirmation in PDLSDU, change receive FCB	14 RStart_Sgm8	R_IDLE
RECEIVE_FRAME DATA PDU received \last data segment of SAP AND memory available ⇒ accept data octets, enter positive confirmation in PDLSDU, change receive FCB, PDL_Data_Ack.indication to PDL-user	15 RData_Sgm1	R_IDLE
RECEIVE_FRAME DATA PDU received \last data segment of a SPA PDU AND no more memory available ⇒ enter negative confirmation (queue full) in PDLSDU, change receive FCB	16 RData_Sgm2	R_IDLE
RECEIVE_FRAME (<i>slave only</i>) DATA PDU received \last data segment of SVA ⇒ accept data octets, enter positive confirmation in PDLSDU, change receive FCB	17 RData_Sgm3	R_IDLE
RECEIVE_FRAME DATA PDU received \not last data segment of a SVA (<i>slave only</i>) or SPA PDU ⇒ accept data octets	18 RData_Sgm4	RECEIVE_FRAME
RECEIVE_FRAME (<i>slave only</i>) SWA PDU received ⇒ accept data octets (observe new position in the data flow)	19 RSWA_Sgm1	RECEIVE_FRAME
RECEIVE_FRAME (<i>master only</i>) last data cycle contained errors ⇒ enter RWA PDU in PDLSDU (correct position in the message), CSWA = 0	20 Cycle_Err	WAIT_SWA

Initial state event condition ⇒ action	Transition	Follow-up state
WAIT_SWA (master only) no SWA PDU received condition: $\backslash C_{SWA} \leq DIST + add\_wait$ ⇒ increment C_{SWA}	21 WAIT2	WAIT_SWA
WAIT_SWA (master only) no SWA PDU received condition: $\backslash C_{SWA} > DIST + add\_wait$	22 Time_Out	R_IDLE
WAIT_SWA (master only) SWA PDU received condition: $\backslash SWA$ PDU does not contain the last data ⇒ accept data octets (observe new position in the data flow)	23 RSWA_Sgm2	RECEIVE_FRAME
WAIT_SWA (master only) SWA PDU received condition: $\backslash SWA$ PDU contains the last data of a SPA AND memory is available ⇒ accept data octets (observe new position in the data flow), enter positive confirmation in PDLSDU, change receive FCB, PDL_Data_Ack.indication to PDL-user	24 RSWA_Sgm3	R_IDLE
WAIT_SWA (master only) SWA PDU received condition: $\backslash SWA$ PDU contains the last data of SPA AND no more memory available ⇒ enter negative confirmation (queue full) in PDLSDU, change receive FCB	25 RSWA_Sgm4	R_IDLE
WAIT_SWA (master only) SWA PDU received condition: $\backslash SWA$ PDU contains the last data of SVA ⇒ accept data octets (observe new position in the data flow), enter positive confirmation in PDLSDU, change receive FCB	26 RSWA_Sgm5	R_IDLE

32.4 Basic Link Layer (BLL)

32.4.1 Functionality of the BLL

The Basic Link Layer is the component of a device that is responsible for the controlled bus access.

In the case of the master it makes the BLL_Data service available at its interface to the PDL. This service allows to run specific data cycles and to exchange data between PDL and BLL. In the slave the BLL ensures that received data is passed to the PDL and that new data are to be sent is accepted from the PDL (see Figure 229).

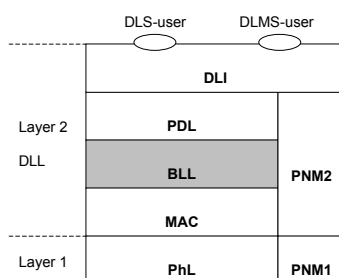


Figure 229 – Location of the BLL in the DLL

The BLL can be parameterized and reset via the interface to the PNM2.

The Basic Link Layer of the master is subdivided into the "BLL operating protocol machine" and "BLL-BAC protocol machine". A slave has only a simplified BLL operating protocol machine.

32.4.2 PDL-BLL interface

32.4.2.1 General

32.4.2 describes the BLL_Data data transmission service, which is available to the PDL, with its service primitives and the associated parameters. The BLL_Data service is mandatory. Figure 230 shows the interface between PDL and BLL in the layer model.

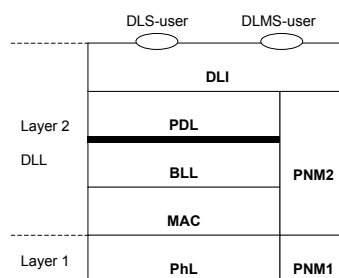


Figure 230 – Interface between PDL and BLL in the layer model

32.4.2.2 Overview of the service and interactions

32.4.2.2.1 BLL_Data

The BLL makes the BLL_Data service available to the PDL. With this service and a PDLSDU the PDL of the master transfers the OUT data within a data cycle to the slaves and receives simultaneous all IN data from the slaves with a PDLSDU. The OUT and IN data are separated with respect to time, that is, the OUT and IN data which are sent or received with a service call, need not belong to one and the same data cycle. Thus, PDL and PhL can operate independently of each other.

The slave behavior is similar to the master:

The BLL of a slave provides the new received OUT data to the PDL by means of indication. The PDL transmits the IN data for sending within next data cycle to the BLL by means of a response. The IN data will be sent in one of the next bus cycles over the physical medium to the master.

The BLL_Data service is provided by using four service primitives. The master uses a request primitive to request a service. A confirmation primitive is returned to the master after the service has been executed. The BLL sends new IN data with the indication primitive to the PDL. The PDL responds to this indication with a response primitive.

Service primitives:

- BLL_Data.request (master side only)
- BLL_Data.confirm (master side only)
- BLL_Data.indication (slave side only)
- BLL_Data.response (slave side only)

32.4.2.3 Overview of the interactions

Figure 231 shows the time relations of the primitives for the BLL_Data service:

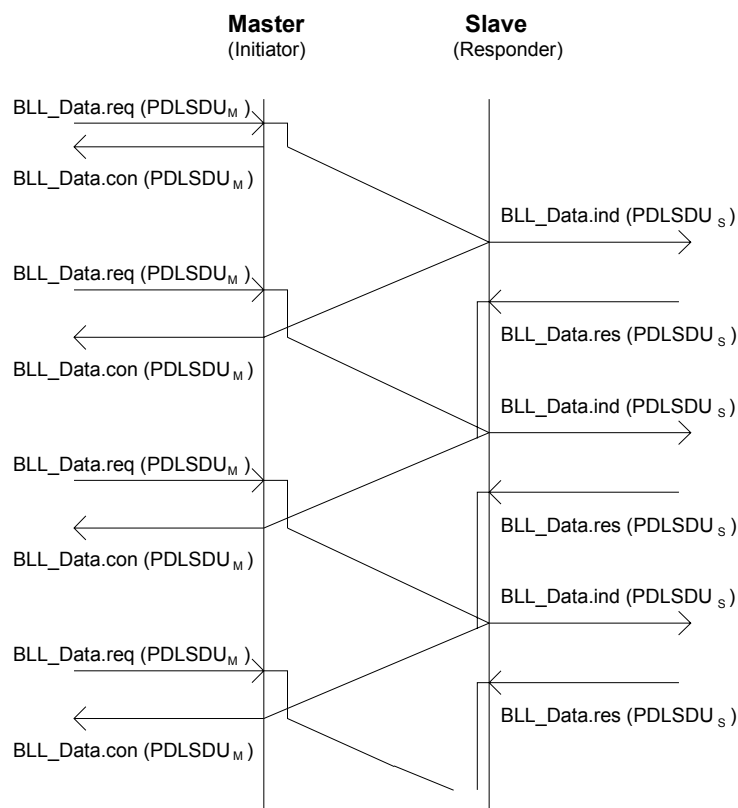


Figure 231 – BLL_Data service

32.4.2.4 Detailed definitions of the services and interactions

32.4.2.4.1 BLL_Data

The BLL_Data service is mandatory.

Using BLL_Data.request (master side only), the PDL of the master shall use this service primitive for sending of a PDLSDU within next data cycle. The PDLSDU shall contain all data which is to be transmitted over the bus in a data cycle. If the BLL of the master received new data, it passes this data as a PDLSDU to the PDL using a BLL_Data.confirm. The update_info parameter contains the information whether the data is valid. The received data is not valid when a data cycle contained errors. The BLL immediately confirms a BLL_Data.request by means of a BLL_Data.confirm with result (-), provided that no valid bus configuration exists or the BLL cannot accept further OUT data owing to a shortage of resources.

The BLL of a slave transfers new received data as a PDLSDU to the PDL using the BLL_Data.indication primitive. The BLL does not get any received data when there are any errors in the data cycles and accordingly does not generate a BLL_Data.indication. Using a BLL_Data.res primitive, the PDL of the slaves sends new transmit data in a PDLSDU to the BLL (see Table 264).

Table 264 – BLL_Data

Parameter name	Request	Indication	Response	Confirm
Argument PDLSDU	M M	M M		
Result(+) PDLSDU update_info			M M	S M M
Result(-) error_code				S M

argument:

The argument contains the service-specific parameters of the service call.

PDLSDU:**Request:**

The PDLSDU parameter contains the OUT data to all slaves, which is to be transferred in one data cycle. The BLL passes the data on to the subordinate MAC layer.

Indication:

The PDLSDU parameter contains the OUT data which was received in the last data cycle without errors.

result(+):

This parameter indicates that the service was executed successfully.

Confirmation:

This parameter contains the IN data which the master received in the last data cycle.

Response:

The PDLSDU contains the IN data of a slave which is to be transmitted in a data cycle.

update_info:

This parameter describes the validity of the IN data. Possible codes are:

- a) OK — the PDLSDU contains valid IN data.
- b) NOK — the PDLSDU does not contain any valid IN data.

result(-):

This parameter indicates that the service could not be executed successfully.

error_code:

This parameter indicates the reason why the service could not be executed successfully. Possible error codes are:

STATE_CONFLICT

The PDL sent a BLL_Data.request, although no valid bus configuration exists.

NO_RESRC

The PDL sent a BLL_Data.request, although the BLL is not ready to accept new OUT data.

32.4.3 PNM2-BLL interface

32.4.3.1 General

The management of the BLL is the part of the BLL that provides the management functionality of the BLL requested by the PNM2. The management of the BLL handles the initialization, the monitoring, and the error recovery in the BLL.

32.4.3 defines the administrative BLL management services which are available to the PNM2, together with their service primitives and associated parameters. Figure 232 shows the interface between PNM2 and BLL in the layer model.

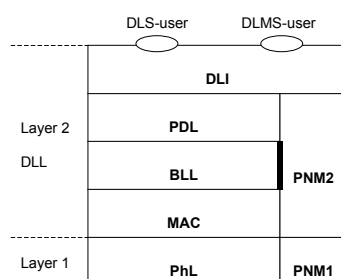


Figure 232 – Interface between PNM2 and BLL in the layer model

The service interface between PNM2 and BLL provides the following functions:

- Reset of the BLL
- Request and change of the current operating parameters of the BLL
- Indication of unexpected events, errors, and status changes which occurred or were detected in the BLL.

32.4.3.2 Overview of the services

32.4.3.2.1 Available services

The BLL makes the following services available to the PNM2

- BLL_ID (acquire bus configuration)
- Reset BLL
- Set Value BLL
- Get Value BLL
- Event BLL

The BLL services are described with the primitives (beginning with BLL_...).

32.4.3.2.2 BLL_ID

The BLL (master side only) makes the required BLL_ID service available to the PNM2. With this service and a BLLSDU the PNM2 of the master transfers the control codes for an identification cycle to the slaves and receives all device codes of an identification cycle from the slaves with a BLLSDU.

Service primitives:

- BLL_ID.request (master only)
- BLL_ID.confirm (master only)

32.4.3.2.3 BLL_Reset

The PNM2 uses this required service to reset the BLL. Upon execution of the service the PNM2 receives a confirmation.

Service primitives:

- BLL_Reset.request
- BLL_Reset.confirm

32.4.3.2.4 BLL_Set_Value

The PNM2 uses this optional service to set a new value to the variables of the BLL. Upon completion, the PNM2 receives a confirmation from the BLL whether the defined variables accepted the new value.

Service primitives:

- BLL_Set_Value.request
- BLL_Set_Value.confirm

32.4.3.2.5 BLL_Get_Value

The PNM2 uses this optional service to read the variables of the BLL. The current value of the defined variable is transmitted in the response of the BLL.

Service primitives:

- BLL_Get_Value.request
- BLL_Get_Value.confirm

32.4.3.2.6 BLL_Event

The BLL uses this required service to inform the PNM2-user about certain events or errors in the BLL.

Service primitive:

- BLL_Event.indication

32.4.3.3 Overview of the interactions

Figure 233 and Figure 234 show the time relations of the service primitives:

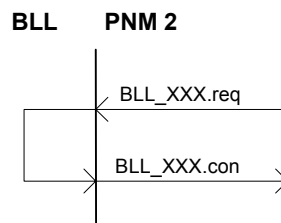


Figure 233 – Reset, Set Value and Get Value BLL services

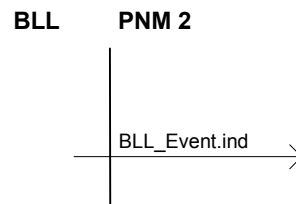


Figure 234 – Event BLL service

32.4.3.4 Detailed definitions of the services and interactions

32.4.3.4.1 BLL_ID (master side only)

Using a BLL_ID.request, PNM2 transfers a BLLSDU to the BLL. The BLL shall initiate an identification cycle when it receives this request. The BLLSDU shall contain all data which is to be transmitted over the bus in an identification cycle. If the BLL of the master received new received data in an identification cycle, it passes this data as a BLLSDU to the PNM2 using a BLL_ID.confirm. The received data is not valid when an identification cycle contained errors (see Table 265).

Table 265 – BLL_Data

Parameter name	Request	Confirm
Argument SDU	M M	
Result(+) SDU		S M
Result(-) error_code		S M

argument:

The argument contains the service-specific parameters of the service call.

SDU:

Request: The SDU parameter contains the control codes to all slaves, which is to be transferred in one identification cycle. The BLL passes the data on to the subordinate MAC layer.

result(+):

This parameter indicates that the service was executed successfully.

Confirmation: This parameter contains the device codes which the master received in the last identification cycle.

result(-):

This parameter indicates that the service could not be executed successfully.

error_code:

This parameter indicates the reason why the service could not be executed successfully.

32.4.3.4.2 BLL_Reset

The BLL_Reset service is mandatory. The PNM2 transfers a BLL_Reset.request to the BLL to reset it (see Table 266).

Table 266 – BLL_Reset

Parameter name	Request	Confirm
Argument	M	
Result(+)		M

32.4.3.4.3 BLL_Set_Value

The BLL_Set_Value service is mandatory. The PNM2 transfers a BLL_Set_Value.request primitive to the BLL to set a defined BLL variable to a desired value. After receipt of this primitive, the BLL tries to select the variable and to set the new value. Upon completion, the BLL transmits a BLL_Set_Value.confirm primitive to the PNM2 (see Table 267).

Table 267 – BLL_Set_Value

Parameter name	Request	Confirm
Argument	M	
variable_name	M	
desired_value	M	
Result(+)		M

variable_name:

This parameter defines the BLL variable which is set to a new value.

desired_value:

This parameter declares the new value for the BLL variable.

Table 268 provides information on which BLL variables may be set to which new values.

Table 268 – BLL variables

Name of BLL variable	Value range	Default
update_time T_{UP}	$(0..2^{15}) * 0,1 \text{ ms}$	
bus_timeout T_{TO_BUS}	$(0..2^{15}) * 1 \text{ ms}$	
Configuration_valid	true, false	false
BLL_access_control	locked, req_to_lock, unlocked	locked

32.4.3.4.4 BLL_Get_Value

The BLL_Get_Value service is optional. The PNM2 transfers a BLL_Get_Value.request primitive to the BLL to read out the current value of a specified BLL variable. After receipt of this primitive, the BLL tries to select the specified variable and transmit its current value to the PNM2 with a BLL_Get_Value.confirm primitive (see Table 269).

Table 269 – BLL_Get_Value

Parameter name	Request	Confirm
Argument variable_name	M M	
Result(+) current_value		M M

variable_name:

This parameter specifies the BLL variable the value of which is to be read out.

desired_value:

This parameter contains the read-out value of the BLL variable.

The BLL variables to be read are exactly those variables that can be written to with BLL_Set_Value.

32.4.3.4.5 BLL_Event

The BLL_Event service is mandatory. The BLL transfers a BLL_Event.indication primitive to the PNM2 to inform it about important events or errors in the BLL (see Table 270 and Table 271).

Table 270 – BLL_Event

Parameter name	Indication
Argument event	M M

event:

This parameter specifies the event which occurred or the error source in the BLL and can assume the following values:

Table 271 – BLL_Event

Name	Meaning	Mandatory / optional
BLL_bus_timeout	There is a bus timeout t_{TO_BUS} , that is, the time between two data cycles completed without errors was too long. The BLL declared the bus configuration invalid.	O
BLL_update_timeout	The was an update timeout before the next data cycle was started.	O
BLL_cycle_error	Identifies cycles with errors, BLL interrupts data cycles, waits for the enabling of PNM2 by means of the BLL_Set_Value (variable: BLL_access_control = unlocked)	M

32.4.4 Protocol machines of the BLL

32.4.4.1 BLL protocol machines of the master

32.4.4.1.1 Overview

The BLL operating protocol machine of the master receives the OUT data of the higher-level PDL layer and passes it to the BLL-BAC-PM. The BLL-BAC-PM then starts a data cycle which is controlled by a timer. Moreover, after a data cycle the BLL operating protocol machine receives the IN data from the BLL-BAC-PM and passes it to the higher-level PDL. It is capable of initiating the next data cycle while the PDL is still processing the data of the last cycle. Another functionality is the monitoring of the bus timeout t_{O_BUS} . After a data cycle with errors, existing PDLSDUs in the BLL are rejected owing to the method of functioning of the PDL protocol machines.

In the master, the BLL-BAC-PM (BAC: 'Basic Access Control') ensures that the update_time t_{UP} is kept for data cycles. For this, it passes, initiated by a timer, BAC_Cycle.requests from the BLL operating protocol machine to the MAC layer as MAC_Cycle.requests. Furthermore, the BLL_Access_Control variable can be used to interrupt the starting of bus cycles by the PNM2 in order to start ID cycles there. As the diagnostic application will possibly run an identification cycle after a data cycle with errors, the BLL-BAC-PM sets the BLL_Access_Control variable after a data cycle error automatically to locked. Upon completion of the ID cycle, the PNM2 shall again enable the starting of the data cycles with a BLL_Set_Value service.

32.4.4.1.2 BLL-internal functions

The BLL-MAC-PM makes the BAC_Cycle service available to the BLL operating protocol machine of the master. In the BLL-MAC-PM the service is mapped onto the MAC_Cycle service of the MAC sublayer. The structure of the BAC_Cycle service exactly corresponds to that of the MAC_Cycle service.

The difference between the two services is that with the BAC_Cycle a bus cycle is not immediately started after the service call, but the cycle start is synchronized with the clock of a timer. The PNM2 can also prevent the start of the cycle with the BLL_Set_Value service (BLL_access_control variable).

32.4.4.1.3 BAC_Reset

After a BLL_Reset, the operating protocol machine resets the BAC-PM with the BAC_Reset service. The service is immediately confirmed.

32.4.4.1.4 BLL operating protocol machine

Figure 235 shows the BLL operating protocol machine of a master.

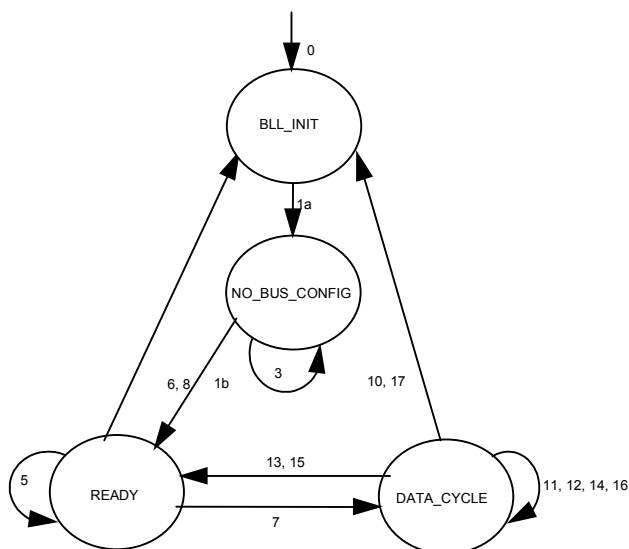


Figure 235 – BLL operating protocol machine of the master

Transitions as a result of BLL_Reset primitive always go from every state to the NO_BUS_CONFIG state. These transitions are not specified individually but are described with the transition 0.

States of the BLL operating protocol machine of the master

32.4.4.1.4.1 BLL_INIT

The Basic Link Layer, including the BLL_access_control and configuration_valid variables, is initialized and/or reset.

32.4.4.1.4.2 NO_BUS_CONFIG

There is no valid active bus configuration. No data cycles can be run.

32.4.4.1.4.3 READY

There is no valid active bus configuration. A data cycle can be initiated with a BLL_Data.request.

32.4.4.1.4.4 DATA_CYCLE

A data cycle was initiated with a BLL_Data.request.

Table 272 describes the state transitions.

The following events may occur in each state and shall be taken into consideration:

- Change of the value of the configuration_valid variable (The PNM2 can change this value with BLL_Set_Value).
- BLL_Data.request
- BAC_Cycle.confirm (if a BAC_Cycle.request was sent before)
- Time t_1 expired, that is, the bus timeout t_{TO_BUS} is exceeded
- BLL_Reset.request

Receive- and transmit BLLSDU:

In the BLL operating protocol machine a distinction is made between receive and transmit BLLSDU. The receive BLLSDU (BLL_RSDU) contains the data which was received by the MAC sublayer after a data cycle, while the transmit BLLSDU (BLL_TSDU) contains all data which was sent to the MAC sublayer prior to a data cycle. As BLL_RSDU is not necessarily passed on to the PDL immediately after the cycle end, the BLL_RSDU is buffered together with a SDU_status_{BLL_RSDU} in the BLL:

SDU_status_{BLL_RSDU}

- a) OK — there is valid input data since a data cycle was completed without errors.
- b) NOK — there is no valid input data since a data cycle has not yet been run or because the last data cycle contained errors.

Table 272 – State transitions of the BLL operating protocol machine of the master

Initial state event \\condition ⇒ action	Transition	Follow-up state
After power on	0	BLL_INIT
BLL_INIT ⇒ Initialize operating PM, reject BLL_TSDUs and BLL_RSDUs which have not yet been processed	1a	NO_BUS_CONFIG
NO_BUS_CONFIG configuration_valid == true (edge) ⇒ generate BLL_RSDU with SDU_status _{BLL_RSDU} = NOK	1b	READY
NO_BUS_CONFIG BLL_Data.request ⇒ BLL_Data.confirm (-) with error_code = STATE_CONFLICT	3	NO_BUS_CONFIG
READY configuration_valid == false (edge)	6	BLL_INIT
READY BLL_Data.request ⇒ accept PDLSDU as BLL_TSDU , BLL_Data.confirm(+) with PDLSDU = BLL_RSDU and update_info = SDU_status _{BLL_RSDU} , BAC_Cycle.request with BLLSDU = BLL_TSDU	7	DATA_CYCLE
READY timer T ₁ expired ⇒ BLL_Event.indication with event = BLL_bus_timeout	8	BLL_INIT
DATA_CYCLE configuration_valid == false (edge)	10	BLL_INIT
DATA_CYCLE BLL_Data.request \\still resources available ⇒ accept PDLSDU as BLL_TSDU	11	DATA_CYCLE
DATA_CYCLE BLL_Data.request \\no more resources available ⇒ BLL_Data.confirm (-) with error_code = NO_RESRC	12	DATA_CYCLE
DATA_CYCLE BAC_Cycle.confirm with BLLSDU and result == OK \\no further BLL_TSDU available ⇒ set timer T ₁ to value T _{TO_BUS} , buffer BLLSDU as BLL_RSDU with SDU_status _{BLL_RSDU} = OK	13	READY

Initial state event \condition ⇒ action	Transition	Follow-up state
DATA_CYCLE BAC_Cycle.confirm with BLLSDU and result == OK \further BLL_TSDU available ⇒ set timer T ₁ to the value T _{TO_BUS} , BLL_Data.confirm(+) with PDLSDU = BLLSDU and update_info = OK, BAC_Cycle.request with BLLSDU = BLL_TSDU	14	DATA_CYCLE
DATA_CYCLE configuration_valid == false (edge)	15	READY
DATA_CYCLE BLL_Data.request \still resources available ⇒ accept PDLSDU as BLL_TSDU	16	DATA_CYCLE
DATA_CYCLE BLL_Data.request \no more resources available ⇒ BLL_Data.confirm (-) with error_code = NO_RESRC	17	BLL_INIT
DATA_CYCLE BAC_Cycle.confirm with BLLSDU and result == OK \no further BLL_TSDU available ⇒ set timer T ₁ to value T _{TO_BUS} , buffer BLLSDU as BLL_RSDU with SDU_statusBLL_RSDU = OK		same_state
DATA_CYCLE BAC_Cycle.confirm with BLLSDU and result == OK \further BLL_TSDU available ⇒ set timer T ₁ to the value T _{TO_BUS} , BLL_Data.confirm(+) with PDLSDU = BLLSDU and update_info = OK, BAC_Cycle.request with BLLSDU = BLL_TSDU		BLL_INIT

32.4.4.1.5 BLL-BAC protocol machine

Figure 236 shows the BLL-BAC protocol machine.

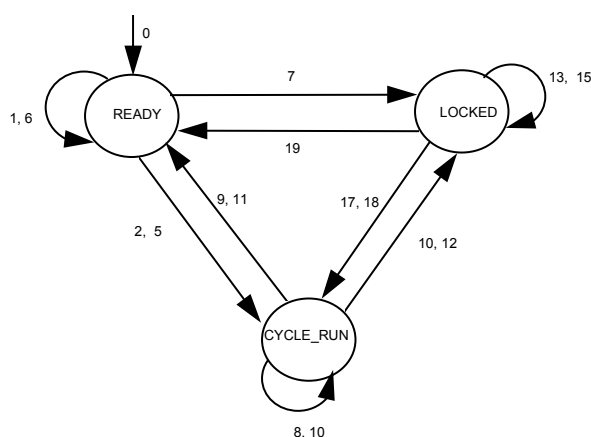


Figure 236 – BLL-BAC protocol machine

Transitions as a result of a BLL_Reset.request primitive always go from every state to the READY state. These transitions are not shown individually but are described by the transition 0.

States of the BLL-BAC-PM (protocol machine)

32.4.4.1.5.1 READY

The initiating of bus cycles is enabled. No cycle is run because the BLL operating protocol machine and diagnostic application did not send a BAC_Cycle.request or because the update time t_{UP} has not yet expired. t_{UP} is considered for data cycles only.

32.4.4.1.5.2 CYCLE_RUN

The initiating of bus cycles is enabled. A bus cycle is running. In this state the protocol machine waits for the end of the bus cycle.

32.4.4.1.5.3 LOCKED

The initiating of bus cycles is locked. However, a running bus cycle can still be completed. If there is a BAC_Cycle.request in this state, the BLLSDU transferred with the request is buffered. Only after the enabling of the bus cycles will a new bus cycle be started. Only one BLLSDU can be buffered.

Table 273 describes the state transitions.

The following events may occur and shall be taken into consideration:

- BAC_Cycle.request
- MAC_Cycle.confirm (if a MAC_Cycle.request was sent before)
- Timer T_2 , that is, the update_time T_{UP} expired
- Change of the BLL_access_control variable
- BAC_Reset.request

Table 273 – State transitions of the BLL-BAC protocol machine

Initial state event \condition ⇒ action	Transition	Follow-up state
After power on ⇒ reset BAC-PM	0	READY
READY BAC_Cycle.request (data_cycle, BLLSDU) \T ₂ started and not yet expired ⇒ retain BLLSDU	1	READY
READY BAC_Cycle.request (data_cycle, BLLSDU) \T ₂ expired or not started ⇒ MACSDU = BLLSDU, MAC_Cycle.request (data_cycle, MACSDU), set T ₂ to T _{UP} , start T ₂	2	CYCLE_RUN
READY T ₂ expired \BAC_Cycle request already received but not yet executed ⇒ MACSDU = BLLSDU, MAC_Cycle.request (data_cycle, MACSDU), set T ₂ to T _{UP} , start T ₂	5	CYCLE_RUN
READY T ₂ expired \no pending BAC_Cycle service available ⇒ BLL_Event.indication with event = BLL_update_timeout	6	READY
READY BLL_access_control == req_to_lock (edge)	7	LOCKED

Initial state event \condition ⇒ action	Transition	Follow-up state
CYCLE_RUN BAC_Cycle.request ⇒ BAC_Cycle.confirm with result = NO	8	CYCLE_RUN
CYCLE_RUN MAC_Cycle.confirm with result == OK ⇒ BAC_Cycle.confirm with result = OK to BLL operating protocol machine	9	READY
CYCLE_RUN MAC_Cycle.confirm with result == NO ⇒ BAC_Cycle.confirm with result = NO to BLL operating protocol machine, BLL_access_control == locked, BLL_Event: BLL_cycle_error	10	LOCKED
CYCLE_RUN T ₂ expired ⇒ BLL_Event.indication with event = BLL_update_timeout	11	CYCLE_RUN
CYCLE_RUN BLL_access_control == req_to_lock (edge)	12	LOCKED
LOCKED BAC_Cycle.request (data_cycle, BLLSDU) \no pending BAC_Cycle service available ⇒ retain BLLSDU	13	LOCKED
LOCKED MAC_Cycle.confirm with result == OK/NO ⇒ BAC_Cycle.confirm with result == OK/NO to BLL operating protocol machine, BLL_access_control = locked, Result == NO → BLL_Event.indication with event = BLL_cycle_error	15	LOCKED
LOCKED BLL_access_control == unlocked (edge) \bus cycle not yet been completed	17	CYCLE_RUN
LOCKED BLL_access_control == unlocked (edge) \bus cycle completed, but BAC_Cycle service is still pending ⇒ MACSDU = BLLSDU, MAC_Cycle.request (data_cycle, MACSDU)	18	CYCLE_RUN
LOCKED BLL_access_control == unlocked (edge) \bus cycle completed AND no BAC_Cycle service is pending	19	READY
any_state BAC_Reset.request ⇒ reset BAC-PM, BAC_Reset.con		READY

32.4.4.2 BLL protocol machine of the slave

32.4.4.2.1 Overview

The BLL-BAC protocol machine does not exist in the slave, since only the master can initiate bus cycles. The BLL operating protocol machine of the slave is greatly simplified. It only passes the data from the MAC to the PDL and vice versa.

32.4.4.2.2 **BLL operating protocol machine**

Figure 237 shows the BLL operating protocol machine of a slave.

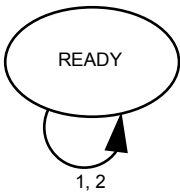


Figure 237 – BLL operating protocol machine of the slave

States of the BLL operating protocol machine of the slave

32.4.4.2.2.1 **READY**

The BLL operating protocol machine is ready to accept new output data in form of a MACSDU from the MAC and passes it to the PDL as a BLLSDU. IN data is also passed from PDL (BLLSDU) to the MAC (MACSDU). The BLL is only responsible for the transmission of data by means of data cycles.

Table 274 describes the state transitions.

The following events may occur in every state and shall be taken into consideration:

- BLL_Data.res
- MAC_Cycle_ind

The MAC_Cycle_ind is a combination of the following interactions at the MAC ↔ MAC-user interface.

- a) MAC_Data.indication (Data_Cycle)
- b) MAC_Get_Data.request (Data_Receive)
- c) MAC_Get_Data.confirm (Data_Receive, OK, MACSDU)

Table 274 – State transitions of the BLL operating protocol machine of the slave

Initial state event \\condition ⇒ action	Transition	Follow-up state
READY MAC_Cycle_ind with MACSDU ⇒ BLLSDU = MACSDU, BLL_Data.indication (BLLSDU)	1	READY
READY BLL_Data.res (BLLSDU) ⇒ MACSDU = BLLSDU, MAC_Cycle_res (MACSDU)	2	READY

MAC_Cycle_res is a combination of the following interactions at the MAC-MAC-user interface:

- 1) MAC_Put_Data.request (Data_Transmit, MACSDU)
- 2) MAC_Put_Data.confirm (Data_Transmit, OK)

32.5 Medium Access Control (MAC)

32.5.1 Location of the MAC in the DLL

The Medium Access Control (MAC) is the lowest sublayer of the Data Link Layer (DLL) and is based on the PhL (PhL) (see Figure 238).

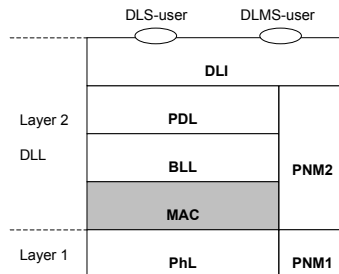


Figure 238 – Location of the MAC in the DLL

As shown in Figure 239, the PhL is also subdivided into several sublayers, the functionality of which follows from Figure 239 as well. The sublayer is called MAC-user and corresponds to the BLL sublayer.

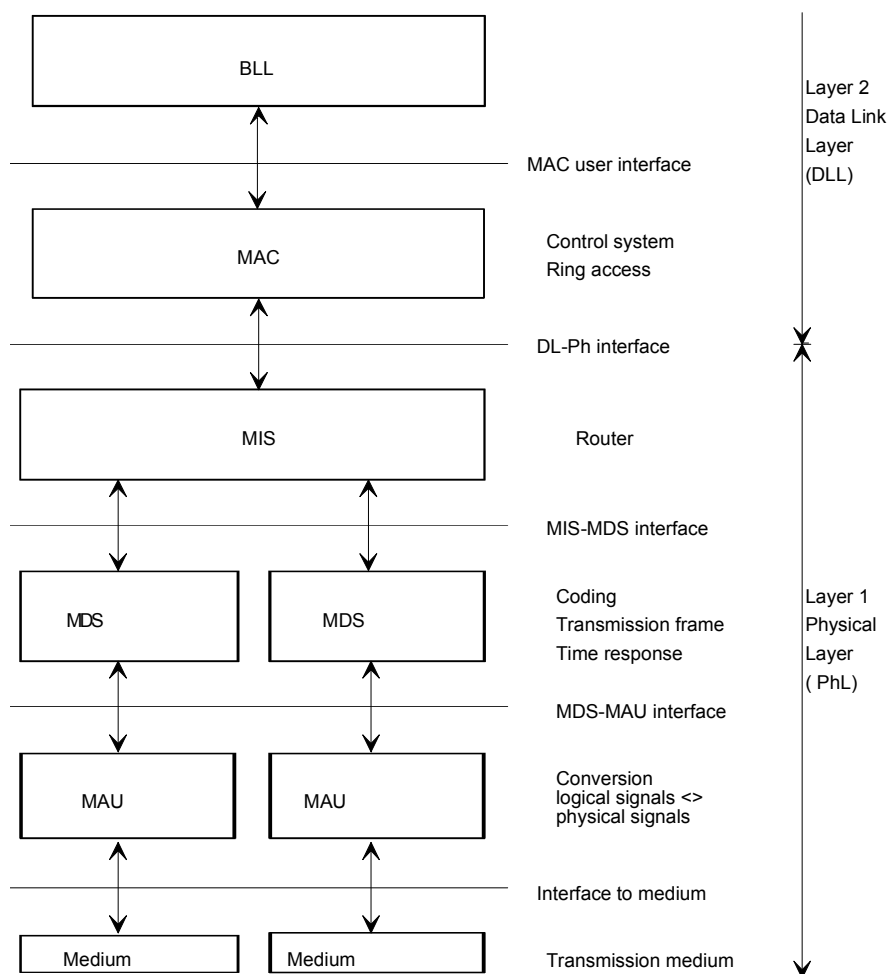


Figure 239 – Model details of layers 1 and 2

32.5.2 Functionality of the MAC

The MAC sublayer controls the access to the transmission medium and ensures that the received and transmitted user data is checked in the form of a 16-bit CRC polynomial.

The MAC sublayer transmits the data, which are received the MAC-user, in one DLPDU cycle as a sequence of binary data units via the DL-Ph interface to the PhL. At the same time the MAC sublayer receives data units via the DL-Ph interface from the PhL. If the PhL has received these data units without errors it makes them available to the MAC-user. It is always the master that initiates a DLPDU cycle, so that a difference is to be made between the active MAC sublayer of a master and the passive MAC sublayer of the slave.

A DLPDU cycle consists of a data sequence which may follow a check sequence (see Figure 240 and Figure 241):

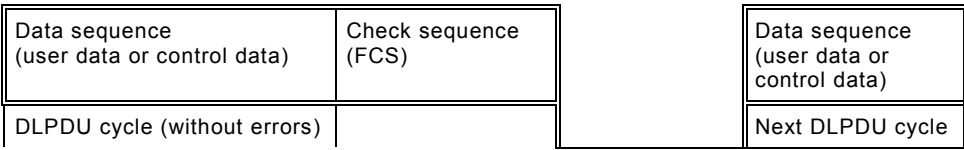


Figure 240 – DLPDU cycle of a data sequence without errors

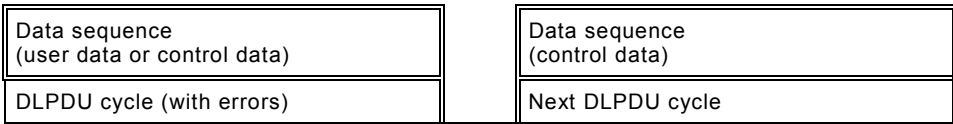


Figure 241 – DLPDU cycle of a data sequence with errors

A data sequence may either include a data cycle for the transmission of user data from the process data channel and, if applicable, from the parameter channel, or an identification cycle for the transmission of data for configuration and error diagnostics. If the data sequence was transmitted without any errors, a check sequence follows which is initiated by the master. This check sequence first transmits the CRC polynomial (checksum) of the data transmitted in the data sequence before, followed by the receive status (checksum status). Should there be an error in the data sequence, and this sequence is followed by another data sequence, the check sequence is omitted and another data sequence is sent, which shall be part of an identification cycle.

The Medium Access Control (MAC) sublayer is a part of the DLL and controls the secured data transmission between the devices over the transmission medium. It transmits the MACSDU from the MAC-user, generates the accordingly DLPDU and transmits it via the DL-Ph interface to the PhL. Conversely, it receives a DLPDU via the DL-Ph interface, and generates the MACSDU from it and transmits it to the MAC-user.

To secure the data transmission, the MAC sublayer generates the checksum of the DLPDU to be transmitted in the form of a CRC polynomial and transmits this sum via the DL-Ph interface to the PhL. Conversely, the MAC sublayer generates the checksum of a received DLPDU in the form of a CRC polynomial and compares it with the received checksum.

32.5.3 Master

32.5.3.1 DLPDU Structure

A distinction is made between the following DLPDU formats: **data sequence DLPDU** and **check sequence DLPDU**.

32.5.3.1.1 Data Sequence DLPDU

The MAC sublayer of a master shall generate the data sequence DLPDU according to Figure 242 by adding the loopback word (LBW) to the MACSDU. The DLPDU thus generated is transmitted from left to right to the PhL in the form of PhIDUs so that the LBW is transmitted first, followed by the MACSDU.

When the number of the data bits transmitted by the BLL cannot be divided by eighth without a remainder, the loopback word (LBW) shall be preceded by extra bits of any contents. The number of these extra bits is 8 bits minus the number of remainder bits.

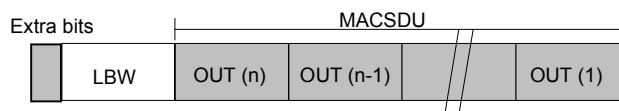


Figure 242 – Data sequence DLPDU transmitted by the master

Conversely, the MAC sublayer of a master shall remove the LBW from a data sequence DLPDU received according to Figure 243 and compare it with the LBW and the extra bits of the last data sequence DLPDU transmitted to the PhL. If both words are identical, the MAC sublayer shall transmit the received MACSDU via the MAC-user interface to the MAC-user. The data sequence DLPDU is received from left to right, starting with the MACSDU.

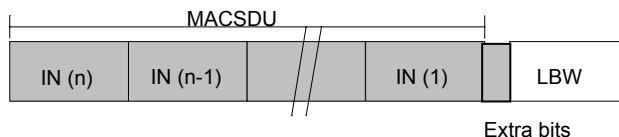


Figure 243 – Data sequence DLPDU received by the master

32.5.3.1.2 Check sequence DLPDU

For a secured transmission of the data sequence DLPDU, the MAC sublayer shall generate a check sequence DLPDU according to Figure 244 after the data sequence DLPDU was transmitted successfully. For this, the MAC sublayer generates a checksum for the transmitted data sequence DLPDU and transmits it together with the checksum status in a check sequence as a check sequence DLPDU via the DL-Ph interface to the PhL.

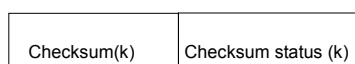


Figure 244 – Check sequence DLPDU

Conversely, the MAC sublayer of the master compare the checksum of a received check sequence DLPDU according to Figure 244 with the checksum calculated for the data sequence DLPDU that was received immediately before. Then it has to evaluate the checksum status of the received check sequence DLPDU. The result is communicated to the MAC-user.

32.5.3.2 Loopback word (LBW)

To secure the data transmission, the MAC shall generate a loopback word (LBW) with a structure as shown in Figure 245.

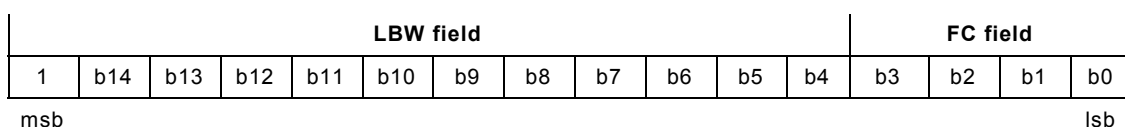


Figure 245 – Loopback word (LBW)

The loopback word is transmitted from right to left, starting with the least significant bit (lsb), and ending with the most significant bit (msb).

In every transmitted LBW the binary value of b3...b0 are used as frame counter (FC), with b3 as the msb and b0 as the lsb, is decremented by the value 1 compared with the value represented by b3...b0 of the last LBW, without considering a possible carry. The MAC-user defines the values for b14...b4 via the management interface by means of the loopback word variable of the MAC.

32.5.3.3 Checksum

To secure the transmission of a data sequence DLPDU against errors, the MAC sublayer of a master generates an independent checksum in the form of a 16-bit FCS value of the data sequence DLPDU to be transmitted as well as of a received data sequence DLPDU. The FCS value is the division remainder resulting from a continuous division of the DLPDU, starting with the lsb, by the standard CCITT polynomial $X^{16} + X^{12} + X^5 + 1$ with pre- and post-division adjustment, all as specified in 5.1 and 5.1.1. However, the transmission procedure differs from that of 5.1.1; for this protocol the FCS is transmitted in a separate DLPDU from the protected data bits.

NOTE 1 Because the FCS value is transmitted in a check sequence DLPDU separate from the data sequence DLPDU, the concatenated reception method of 5.1.2 does not apply.

The checksum is transmitted in a check sequence DLPDU, beginning with the lsb and ending with the msb.

NOTE 2 The checksum may be generated synchronously with the transmission or receipt of the data sequence DLPDU.

NOTE 3 If a transmission error is detected when a data sequence DLPDU is received, the checksum for the received data sequence DLPDU is set to its initial value, L(X), as specified in 5.1 and 5.1.1.

32.5.3.4 Checksum status

The MAC sublayer of a master generates a checksum status as shown in Figure 246.

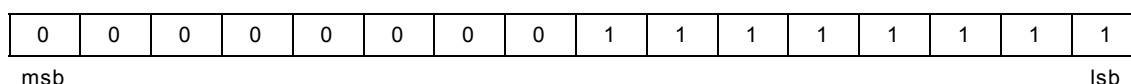


Figure 246 – Checksum status generated by the master

In a check sequence DLPDU transmission takes place immediately after the transmission of the checksum, starting with the lsb and ending with the msb.

In the same way the MAC sublayer of a master shall evaluate a checksum status received in a check sequence DLPDU as shown in Figure 247.

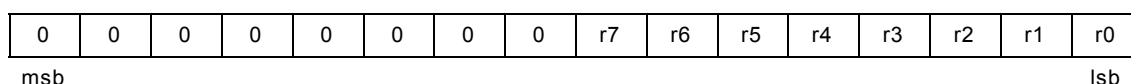


Figure 247 – Checksum status received by the master

For the logical binary values $r7...r0$ means: $r7=r6=r5=r4=r3\wedge r2\wedge r1\wedge r0$, where " $\wedge$ " represents the bit-by-bit AND operator.

NOTE 1 If the expression $r3\wedge r2\wedge r1\wedge r0$ assumes the value logical 1, both the data sequence DLPDUs and the checksum between two devices was transmitted without errors.

NOTE 2 If the expression $r3\wedge r2\wedge r1\wedge r0$ assumes the value 0, then a transmission error occurred either when the data sequence DLPDUs were transmitted or the checksums between two devices.

The checksum status is received from right to left in a check sequence DLPDU immediately after the checksum. The lsb is transmitted first and the msb last.

32.5.3.5 Bus access control

The data transmission is described with separate protocol machines for the send and receive part.

32.5.3.5.1 Sender

Figure 248 shows the state transitions of the MAC sublayer for the transmission of a message in a data sequence (identification cycle or data cycle) from the master to the passive slaves as well as the mechanisms for the data transmission security (check sequence).

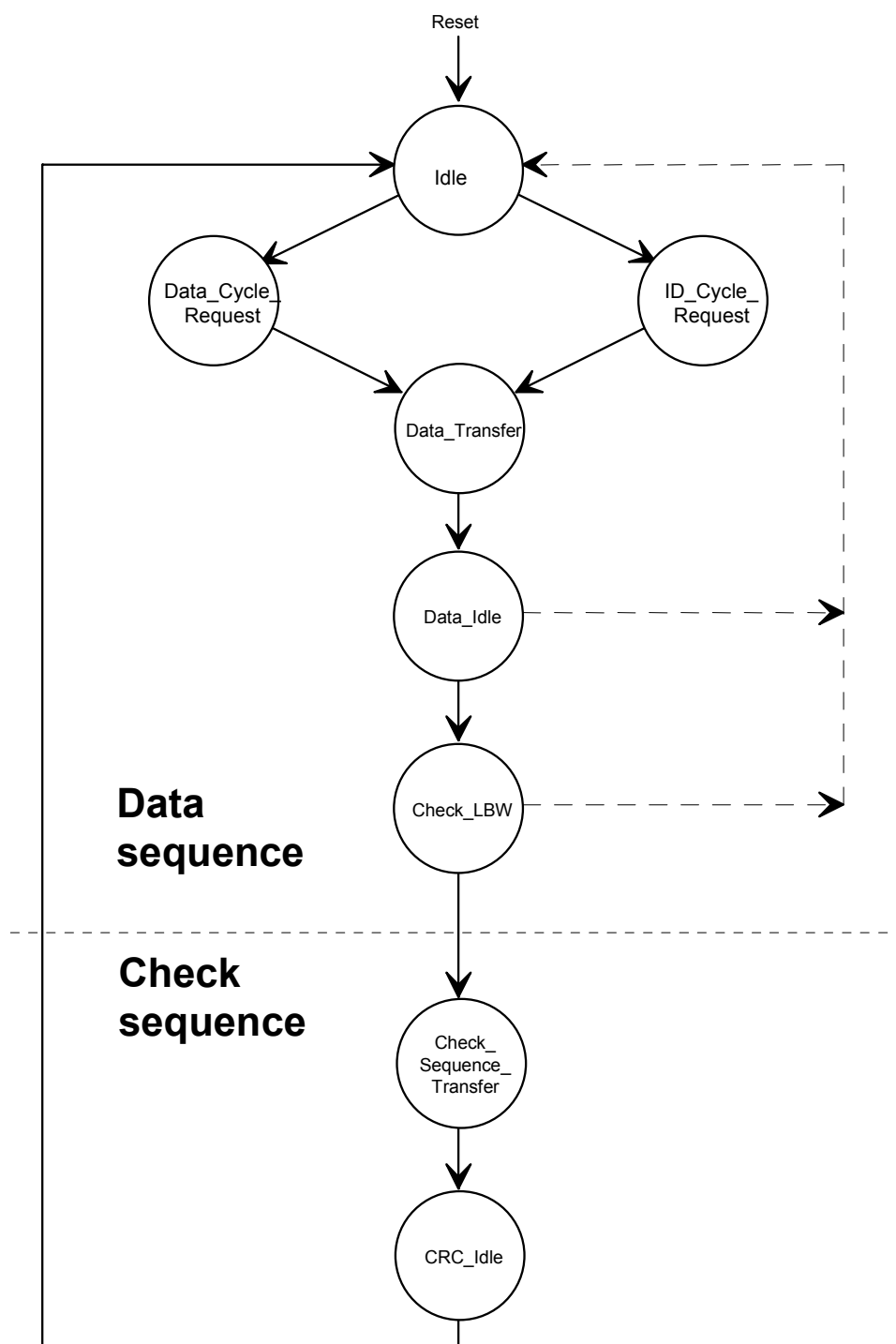


Figure 248 – MAC protocol machine of a master: transmission of a message

32.5.3.5.1.1 Data Sequence

32.5.3.5.1.1.1 Idle

In this state the MAC sublayer of a master shall wait for the request to start a DLDPDU cycle through the MAC-user by means of a MAC_Data.request primitive.

If the MAC sublayer receives the request to start an identification cycle (cycle=ID_cycle) from the MAC-user by means of a MAC_Data.request primitive, it shall generate the MACSDU to be transmitted from the data which is available in the ID_transmit buffer for the transmission and assume the ID_cycle_request state. While data is accepted, any other access to the ID_transmit buffer is locked.

If the MAC sublayer receives the request to start a data cycle (cycle=data_cycle) from the MAC-user by means of a MAC_Data.request primitive, it shall generate the MACSDU to be transmitted from the data which is available in the data_transmit buffer for the transmission and assume the Data_Cycle_Request state. While data is accepted, any other access to the data_transmit buffer is locked.

32.5.3.5.1.1.2 ID_Cycle_Request

In this state the MAC sublayer shall first generate the data sequence DLPDU to be transmitted and then start an identification cycle via the DL-Ph interface by means of a Ph-DATA.request primitive (PhICI=start_ID_cycle). After the confirmation through the PhL by means of a Ph-DATA.confirm primitive, the MAC sublayer shall assume the Data_Transfer state.

32.5.3.5.1.1.3 Data_Cycle_Request

In this state the MAC sublayer shall first generate the data sequence DLPDU to be transmitted and then start a data cycle via the DL-Ph interface by means of a Ph-DATA.request primitive (PhICI=start_data_cycle). After the confirmation through the PhL by means of a Ph-DATA.confirm primitive, the MAC sublayer shall assume the Data_Transfer state.

32.5.3.5.1.1.4 Data_Transfer

In this state the MAC sublayer shall first transmit sequentially the data sequence DLPDU by means of the Ph-DATA.request primitives (PhICI=user_data) via the DL-Ph interface to the PhL. The MAC sublayer receives a confirmation for every Ph-DATA.request primitive through the PhL by means of a Ph-DATA.confirm primitive. At the same time the check sequence DLPDU is generated synchronously.

After a complete transmission of the data sequence DLPDU, the MAC sublayer shall assume the Data_Idle state.

32.5.3.5.1.1.5 Data_Idle

In this state the MAC sublayer requests a Ph-DATA.request (PhICI=User_Data_Idle) and waits until the data sequence DLPDU has been completely received. If the receive time monitoring circuit responded, since an identification or data cycle has been started and before the data sequence DLPDU has been completely received, the MAC sublayer shall terminate the data sequence with a corresponding MAC_Data.confirm primitive to the MAC-user and assume the Idle state.

After the data sequence DLPDU has been completely received, the MAC sublayer shall assume the Check_LBW state.

NOTE If a transmission error was detected during the data sequence, the MAC-user started with an identification cycle in the next DLPDU cycle.

32.5.3.5.1.1.6 Check_LBW

In this state, the MAC sublayer first checks whether a transmission error has been detected between the start of an identification or data cycle and the complete receipt of the data sequence DLPDU. If this is the case, it shall end the data sequence with a corresponding MAC_Data.confirm primitive to the MAC-user and assume the Idle state.

If no receive error was detected, the received loopback word is compared with the loopback word transmitted from the MAC sublayer. If both words are identical, the MAC sublayer terminates the data sequence, generates the check sequence DLPDU and assumes the Check_Sequence_Transfer state.

If the received loopback word is not identical with the sent word, a transmission error occurred and the MAC sublayer shall terminate the data sequence with a corresponding MAC_Data.confirm primitive to the MAC-user and assume the Idle state.

32.5.3.5.1.2 Check sequence

32.5.3.5.1.2.1 Check_Sequence_Transfer

In this state, the MAC sublayer shall transmit the check sequence DLPDU sequentially via the DL-Ph interface to the PhL. The transmission starts with the checksum, which is transmitted by means of the Ph-DATA.request primitives (PhICI=CRC_data), followed by the checksum status which is transmitted by means of the Ph-DATA.request primitives (PhICI=CRC_status).

After the check sequence DLPDU has been completely transmitted the MAC sublayer shall assume the CRC_Idle state.

32.5.3.5.1.2.2 CRC_Idle

In this state the MAC sublayer requests a Ph-DATA.request (PhICI=CRC_Data_Idle) and waits until the check sequence DLPDU has been completely received. If the receive time monitoring circuit responded, since the transmission of the check sequence DLPDU started and before the complete receipt of the check sequence DLPDU, the MAC sublayer shall terminate the check sequence with a corresponding MAC_Data.confirm primitive to the MAC-user and assume the Idle state.

After the check sequence DLPDU has been completely received, the MAC sublayer checks whether a transmission error was detected in the check sequence. If no transmission error was detected, the MAC sublayer makes the received MACSDU available to the MAC-user in the ID_receive buffer, when the MACSDU was received in an identification cycle, or in the data_receive buffer, when the MACSDU was received in a data cycle. Afterwards, it completes the DLPDU cycle with a MAC_Data.confirm primitive (status=OK) to the MAC-user and assumes the Idle state. During the data transfer any other access to the corresponding buffer is locked.

If the check sequence detected a transmission error, the MAC sublayer completes the DLPDU cycle with a corresponding MAC_Data.confirm primitive to the MAC-user and assumes the Idle state.

NOTE If a transmission error was detected during the check sequence, the MAC-user started with an identification cycle in the next DLPDU cycle.

32.5.3.5.2 Receiver

The receive part of the MAC sublayer is described with two protocol machines: one protocol machine describes the receipt of a message, consisting of a data sequence and a check sequence, and the second protocol machine describes the identification of a data sequence as an identification or data cycle.

32.5.3.5.2.1 Message receiver

Figure 249 shows the state transitions in the MAC sublayer when a message is received in a data sequence (identification cycle or data cycle) by the master and the mechanisms for checking a secured data transmission (check sequence).

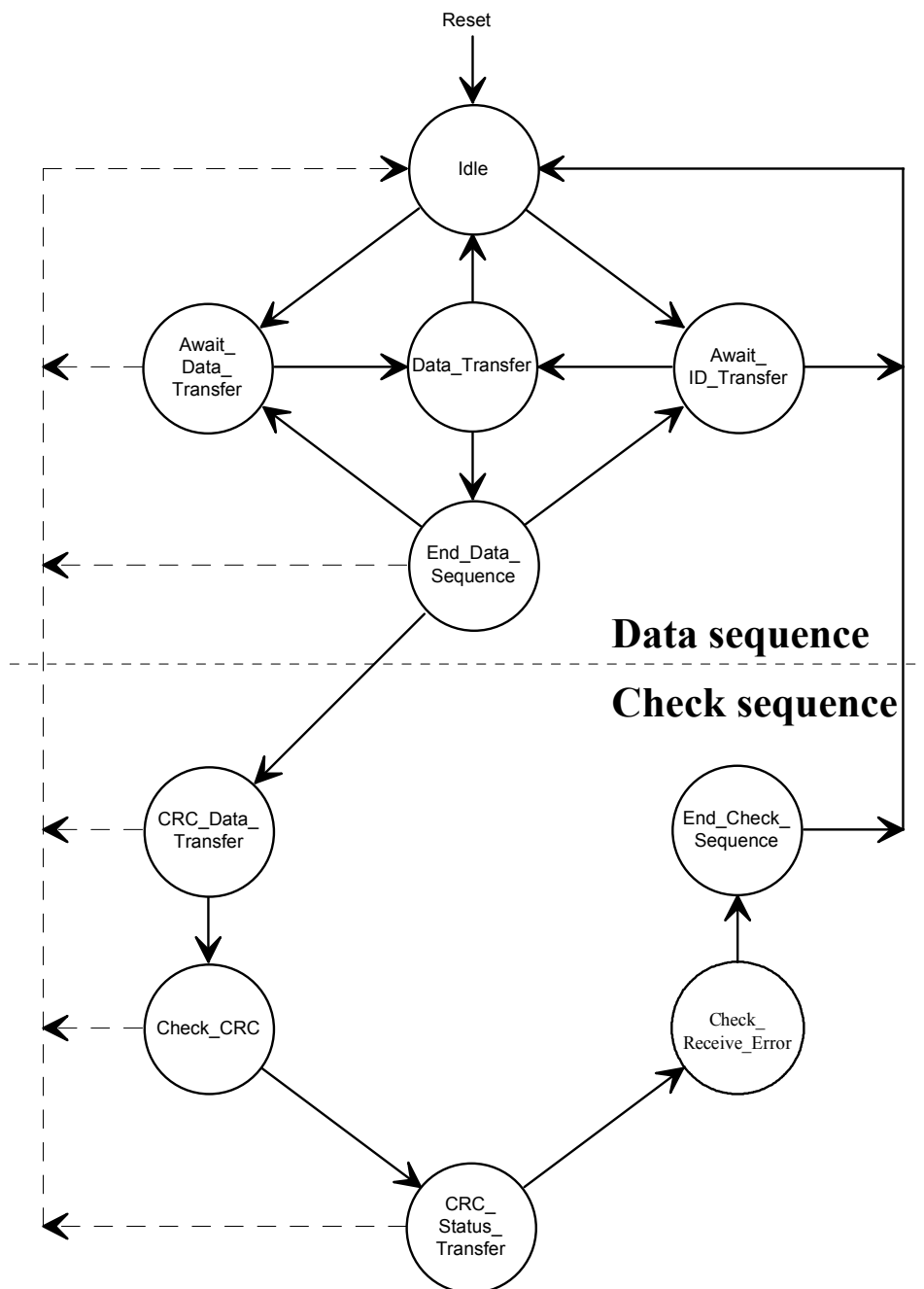


Figure 249 – MAC protocol machine of a master: receipt of a message

32.5.3.5.2.1.1 Data sequence

32.1.5.3.5.2.1.1 Idle

In this state the MAC sublayer shall wait until the MAC-user starts an identification or data cycle. If the MAC-user requests an identification cycle by means of a MAC_Data.request primitive (cycle=ID_cycle), the MAC sublayer assumes the Await_ID_Transmit state. If the MAC-user requests a data cycle by means of a MAC_Data.request primitive (cycle=data_cycle), it assumes the Await_Data_Cycle state.

If the MAC sublayer receives characters of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface before an identification or data cycle is started, there is a transmission error which is reported to the MAC-user with MAC_Event.indication primitive (event=data_noise). The received characters are rejected.

In this case the MAC-user shall request an identification cycle to restore the data consistency.

If the MAC sublayer receives characters of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface before an identification or data cycle is started, there is a transmission error which is reported to the MAC-user with MAC_Event.indication primitive (event=CRC_noise). The received characters are rejected.

In this case the MAC-user shall request an identification cycle to restore the data consistency.

32.2.5.3.5.2.1.1 Await_ID_Transfer

In this state the MAC sublayer waits for the beginning of the data sequence DLPDU. If it receives the first character of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface, it shall assume the Data_Transfer state.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall stop the receipt of the data sequence DLPDU and assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

If the MAC sublayer receives characters of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface, there is a transmission error which is reported to the MAC-user with MAC_Event.indication primitive (event=CRC_noise). The received characters are rejected. In this case the MAC-user shall request an identification cycle to restore the data consistency.

32.3.5.3.5.2.1.1 Await_Data_Transfer

In this state the MAC sublayer waits for the beginning of the data sequence DLPDU. If it receives the first character of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface, it shall assume the Data_Transfer state.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall stop the receipt of the data sequence DLPDU and assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

If the MAC sublayer receives characters of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface, there is a transmission error which is reported to the MAC-user with MAC_Event.indication primitive (event=CRC_noise). The received characters are rejected. In this case the MAC-user shall request an identification cycle to restore the data consistency.

32.4.5.3.5.2.1.1 Data_Transfer

In this state the MAC sublayer shall receive the data sequence DLPDU which is transmitted character by character with the Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface to the MAC sublayer. After the data sequence DLPDU has been completely received it shall assume the End_Data_Sequence state.

NOTE The quantity of characters which the MAC sublayer of a master received from the PhL in a data sequence DLPDU is always equal to the quantity of characters which is transmitted from this MAC sublayer in a data sequence DLPDU to the PhL.

There is a transmission error if the receive time monitoring circuit responds before the data sequence DLPDU has been completely received. The MAC sublayer shall stop the receipt of the data sequence DLPDU and assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the data sequence DLPDU has been completed.

If the MAC sublayer receives characters of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface, there is a transmission error which is reported to the MAC-user with a MAC_Event.indication primitive (event=CRC_noise). The received characters are rejected. In this case the MAC-user shall request an identification cycle to restore the data consistency.

32.5.5.3.5.2.1.1 End_Data_Sequence

In this state the MAC sublayer shall terminate the receipt of the data sequence DLPDU, generate the received MACSDU, and wait for the beginning of a check sequence DLPDU or the start of a new data sequence by the MAC-user.

If the MAC sublayer receives the first character of a check sequence DLPDU via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=CRC_data), it shall assume the CRC_Data_Transfer state and start to receive the check sequence DLPDU.

If the receive time monitoring circuit responds, there is a transmission error and the MAC sublayer shall assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the check sequence DLPDU has been completed.

There is a transmission error if the MAC sublayer receives a character of a data sequence DLPDU via the DL-Ph interface with a Ph-DATA.indication primitive (PhICI=user_data). This transmission error is communicated to the MAC-user with a MAC_Event.indication primitive (event=data_noise) and, if necessary, in the MAC_Data.confirm primitive after the check sequence DLPDU has been completely transmitted. The MAC sublayer shall reject this character.

If the received and sent loopback word are not identical the MAC sublayer assumes the Idle state.

32.5.3.5.2.1.2 Check sequence

32.1.5.3.5.2.1.2 CRC_Data_Transfer

In this state the MAC sublayer shall receive the checksum transmitted via the DL-Ph interface to the MAC sublayer. The checksum is received with Ph-DATA.indication primitives (PhICI=CRC_data). After the checksum has been completely received, the MAC sublayer shall assume the Check_CRC state.

If the receive time monitoring circuit responds, there is a transmission error and the MAC sublayer shall assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm after the transmission of the check sequence DLPDU has been completed.

There is a transmission error if the MAC sublayer receives a character of a data sequence DLPDU via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=user_data). This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the check sequence DLPDU has been completed. The MAC sublayer shall treat this character like a character of a check sequence DLPDU and continue to receive the checksum.

32.2.5.3.5.2.1.2 Check_CRC

In this state the MAC sublayer shall compare the received checksum with the one it previously generated of the last data sequence DLPDU. If the two checksums are identical, the data sequence DLPDU received last was received without errors. Otherwise, there is a transmission error which is communicated to the MAC-user after the end of the check sequence in the MAC_Data.confirm primitive.

If the MAC sublayer receives the first character of a checksum status (PhICI=CRC_status) via the DL-Ph interface by means of a Ph-DATA.indication primitive, the MAC sublayer shall assume the CRC_Status_Transfer state and start to receive the checksum status.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the check sequence DLPDU has been completed.

32.3.5.3.5.2.1.2 CRC_Status_Transfer

In this state the MAC sublayer shall receive the checksum status transmitted via the DL-Ph interface to the MAC sublayer. The checksum status is received character by character with the Ph-DATA.indication primitives (PhICI=CRC_status). After the first four characters of the checksum status (r0...r3) have been completely received, the MAC sublayer shall assume the Check_Receive_Error state.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the check sequence DLPDU has been completed.

32.4.5.3.5.2.1.2 Check_Receive_Error

In this state, the MAC sublayer shall evaluate the logic state of the first four characters r0...r3 of the received checksum status. If the bit-by-bit logical AND combination $r3 \wedge r1 \wedge r0$ assumes the binary value of logical 1, the data sequence DLPDU received last was received without errors. Otherwise, there is a transmission error which is communicated to the MAC-user in the MAC_Data.confirm primitive after the check sequence has been completed.

If the MAC sublayer received a character of a check sequence via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=CRC_status), it shall assume the End_Check_Sequence state and continue to receive the checksum status.

32.5.5.3.5.2.1.2 End_Check_Sequence

In this state the MAC sublayer shall continue to receive the checksum status. After the checksum status has been completely received, the MAC sublayer shall terminate the receipt of the check sequence DLPDU and assume the Idle state.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall assume the Idle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the check sequence DLPDU has been completely transferred.

32.5.3.5.2.2 Data sequence identification

Figure 250 shows the state transitions in the MAC sublayer of a master for the identification of a data sequence as an identification cycle or data cycle.

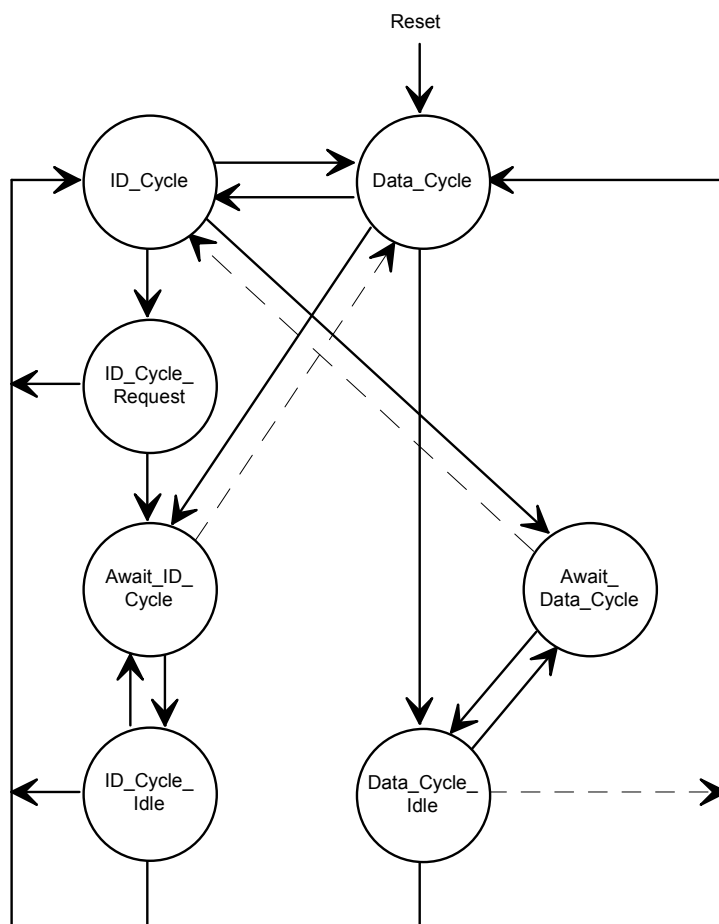


Figure 250 – MAC sublayer of a master: data sequence identification

32.5.3.5.2.2.1 Identification cycle

32.1.5.3.5.2.2.1 ID_Cycle

In this state, the MAC sublayer shall wait for the MAC-user to start an identification or data cycle. If the MAC-user requests an identification cycle by means of a MAC_Data.request primitive (cycle=ID_cycle), the MAC sublayer assumes the ID_Cycle_Request state. If the MAC-user requests a data cycle by means of a MAC_Data.request primitive (cycle=data_cycle), it assumes the Await_Data_Cycle state.

If the MAC receives the beginning of a data cycle via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=data_transfer), there is an error and the MAC sublayer assumes the Data_Cycle state.

32.2.5.3.5.2.2.1 ID_Cycle_Request

If the MAC sublayer receives in this state the request for a data cycle (PhICI=data_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive, it assumes the Await_ID_Cycle state.

If the receive time monitoring circuit responds, there is a transmission error and the MAC sublayer shall assume the ID_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

If the MAC sublayer receives the first character of a data sequence DLPDU (PhICI=user_data), there is a transmission error and the MAC sublayer shall assume the ID_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

If the MAC sublayer receives the first character of a check sequence DLPDU (PhICI=CRC_data), there is also a transmission error and the MAC sublayer shall assume the ID_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the data sequence DLPDU has been completed.

32.3.5.3.5.2.2.1 Await_ID_Cycle

In this state the MAC sublayer waits for the beginning of an identification cycle. If the MAC sublayer receives the beginning of an identification cycle via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=ID_transfer), it assumes the ID_Cycle_Idle state.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall assume the Data_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

If the MAC sublayer receives the first character of a data sequence DLPDU (PhICI=user_data), there is a transmission error and the MAC sublayer shall assume the Data_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the data sequence DLPDU has been completely transmitted.

If the MAC sublayer receives the first character of a check sequence DLPDU (PhICI=CRC_data), there is also a transmission error and the MAC sublayer shall assume the Data_Cycle state.

32.4.5.3.5.2.2.1 ID_Cycle_Idle

In this state the MAC sublayer waits for the beginning of the data transmission. If it receives the first character of a data sequence DLPDU (PhICI=user_data) via the DL-Ph interface by means of a Ph-DATA.indication primitive, it assumes the ID_Cycle state. There is a transmission error if the layer receives the first character of a check sequence DLPDU (PhICI=CRC_data). The MAC sublayer shall also assume the ID_Cycle state.

If the MAC sublayer receives the request for an identification cycle (PhICI=data_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive there is an error, and the MAC sublayer assumes the Await_ID_Cycle state.

If the receive time monitoring circuit responds, there is a transmission error and the MAC sublayer shall assume the ID_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

32.5.3.5.2.2.2 Data cycle

32.1.5.3.5.2.2.2 Data_Cycle

In this state, the MAC sublayer shall wait for the MAC-user to start an identification or data cycle. If the MAC-user requests an identification cycle by means of a MAC_Data.request primitive (cycle=ID_cycle), the MAC sublayer assumes the Await_ID_Request state. If the MAC-user requests a data cycle by means of a MAC_Data.request primitive (cycle=data_cycle), it assumes the Data_Cycle_Idle state.

If the MAC receives the beginning of a data cycle via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=data_transfer), there is an error and the MAC sublayer assumes the ID_Cycle state.

32.2.5.3.5.2.2.2 Await_Data_Cycle

In this state, the MAC sublayer waits for the beginning of a data cycle. If the MAC sublayer receives the beginning of a data cycle via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=data_transfer), it assumes the Data_Cycle_Idle state.

If the receive time monitoring responds, there is a transmission error and the MAC sublayer shall assume the ID_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

If the MAC sublayer receives the first character of a check sequence DLPDU (PhICI=CRC_data), there is also a transmission error and the MAC sublayer shall assume the ID_Cycle state.

If the MAC sublayer receives the first character of a data sequence DLPDU (PhICI=user_data), there is a transmission error and the MAC sublayer shall assume the ID_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the data sequence DLPDU has been completely transmitted.

32.3.5.3.5.2.2.2 Data_Cycle_Idle

In this state the MAC sublayer waits for the beginning of the data transmission. If it receives the first character of a data sequence DLPDU (PhICI=user_data) via the DL-Ph interface by means of a Ph-DATA.indication primitive, it assumes the Data_Cycle state. There is a transmission error if the layer receives the first character of a check sequence DLPDU (PhICI=CRC_data). The MAC sublayer shall also assume the Data_Cycle state.

If the MAC sublayer receives the beginning of an identification cycle with a Ph-DATA.indication primitive (PhICI=ID_transfer) via the DL-Ph interface, there is an error and the MAC sublayer assumes the Await_Data_Cycle state.

If the receive time monitoring circuit responds, there is a transmission error and the MAC sublayer shall assume the Data_Cycle state. This transmission error is communicated to the MAC-user in the MAC_Data.confirm primitive after the transmission of the data sequence DLPDU has been completed.

32.5.4 Slave

32.5.4.1 DLPDU structure

The following DLPDU formats are distinguished: **data sequence DLPDU** and **check sequence DLPDU**.

32.5.4.1.1 Data sequence DLPDU

The MAC sublayer of a slave with the address k shall remove the MACSDU (shown at the right of Figure 251) from a data sequence DLPDU received via the DL-Ph interface as shown in Figure 251 and, if the transmission was error-free, transmit this MACSDU via the MAC-user interface to the MAC-user. The data sequence DLPDU is received from left to right.

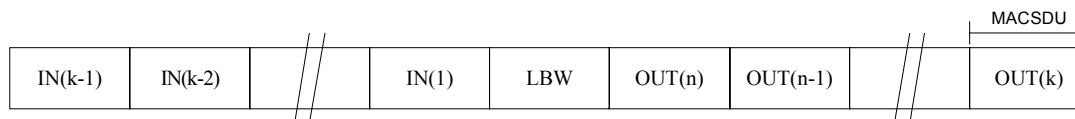


Figure 251 – Data sequence DLPDU received by a slave

Conversely, the MAC sublayer of a slave with the address k shall generate the data sequence DLPDU as shown in Figure 252, by removing the MACSDU destined for its MAC-user from the received data sequence DLPDU and transmitting the rest of the received data sequence DLPDU together with the MACSDU (shown at the left of Figure 252) via the DL-Ph interface to the PhL. This MACSDU was transmitted via the MAC-user interface to the MAC sublayer before. The data sequence DLPDU is transmitted from left to right in the form of PhIDUs, so that first the MACSDU sent from the MAC-user is transmitted to the MAC sublayer and then the received DLPDU without the MACSDU destined for the MAC-user.

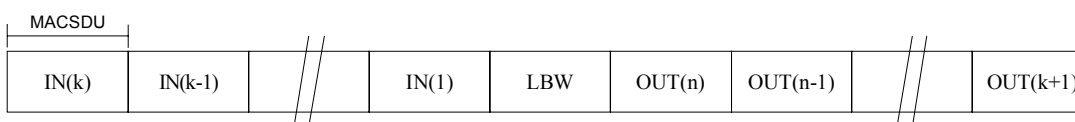


Figure 252 – Data sequence DLPDU transmitted by a slave

32.5.4.1.2 Check sequence DLPDU

For a secured transmission of the data sequence DLPDU the MAC sublayer of a slave transmits a check sequence DLPDU. For this purpose, the MAC sublayer for the transmitted data sequence DLPDU generates a checksum as specified in 5.1 and 5.1.1, but transmits the 16 redundant check bits together with the checksum status of a check sequence as a check sequence DLPDU via the DL-Ph interface to the PhL.

Conversely, the MAC sublayer of a slave shall compare the 16-bit checksum of a check sequence DLPDU with the checksum generated for the data sequence DLPDU received immediately prior and evaluate the checksum status. The result is communicated to the MAC-user.

32.5.4.2 Checksum

see 32.5.3.3

32.5.4.3 Checksum status

The MAC sublayer of a slave shall evaluate a checksum status according to Figure 253 that was received in a check sequence DLPDU.

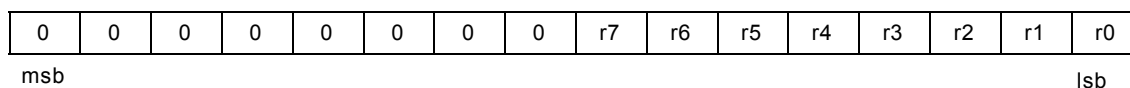


Figure 253 – Checksum status received by the slave

For the logical binary values $r7...r0$ is: $r7=r6=r5=r4=r3\wedge r2\wedge r1\wedge r0$, where " $\wedge$ " represents the bit-wise AND operator.

NOTE 1 If the expression $r3\wedge r2\wedge r1\wedge r0$ assumes the value logical 1, both the data sequence DLPDUs and the checksum between the master and the evaluating device were transmitted without errors.

NOTE 2 If the expression $r3\wedge r2\wedge r1\wedge r0$ assumes the value 0, then a transmission error occurred either when the data sequence DLPDUs were transmitted or the checksums between the master and the evaluating device.

The checksum status is received from right to left in a check sequence DLPDU immediately after the checksum has been received. The lsb is transmitted first and the msb last.

Conversely, the MAC sublayer of a slave shall generate a checksum status according to Figure 254.

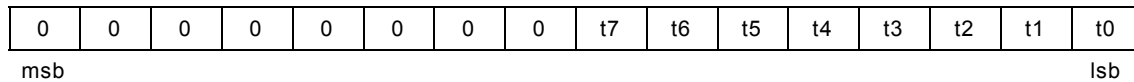


Figure 254 – Checksum status generated by the slave

The logical binary values t7...t0 mean the following:

t0 = CRC_receive_error $\wedge$ r0 $\wedge$ RxSL_Error

t1 = t0 $\wedge$ r1

t2 = t1 $\wedge$ r2

t3 = t2 $\wedge$ r3

t7 = t6=t5=t4=t3

Here r3...r0 represent the corresponding binary values of the received checksum status and " $\wedge$ " the bit-by-bit AND operator.

NOTE 3 If the data sequence DLPDU received last did not contain any errors, that is, the checksum received next in a check sequence DLPDU is identical with the checksum generated from the data sequence DLPDU received last, the CRC_receive_error assumes the value logical 1 and otherwise the value logical 0.

The checksum status is transmitted in a check sequence DLPDU immediately after the checksum has been transmitted, starting with the lsb and ending with the msb.

32.5.4.4 Bus access control

Figure 255 and Figure 256 together show the protocol machine of the MAC sublayer of a slave for the bus access control.

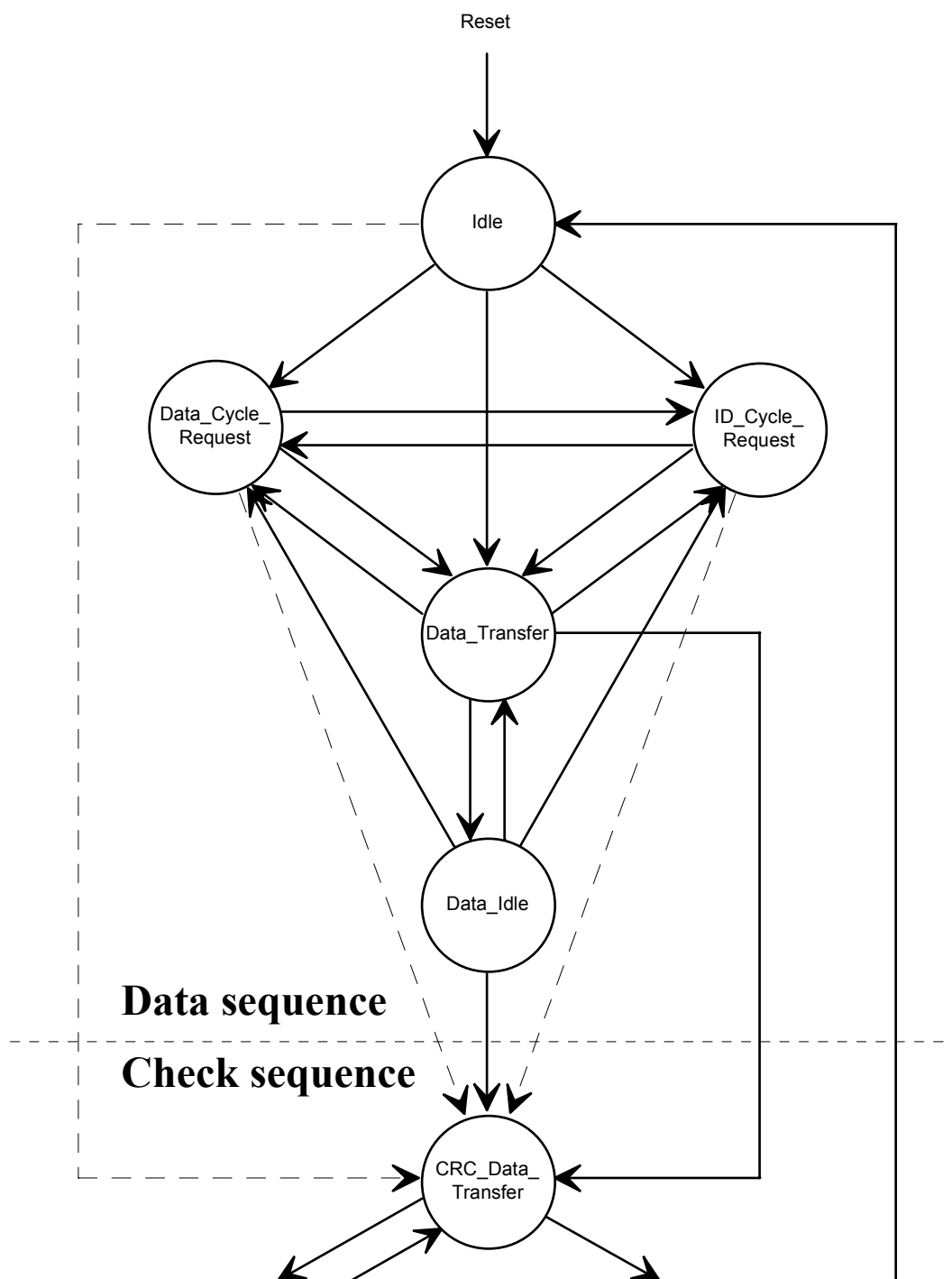


Figure 255 – State transitions of the MAC sublayer of a slave: data sequence

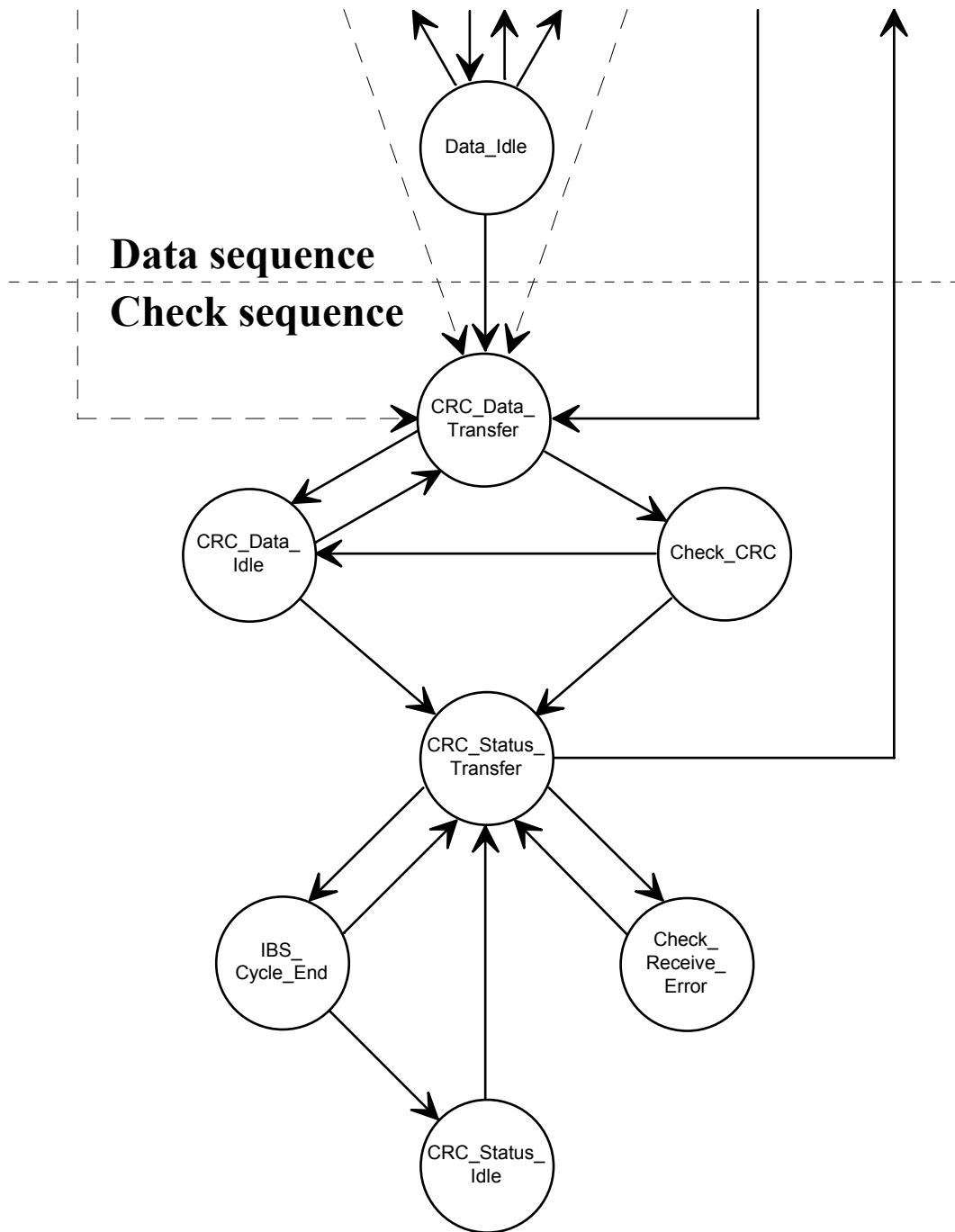


Figure 256 – State transitions of the MAC sublayer of a slave: check sequence

32.5.4.4.1 Basic state

Idle

In this state the MAC sublayer shall first set the checksums for the data sequence DLPDU to be received via the DL-Ph interface and the data sequence DLPDU to be transmitted to its initial value and then wait for the receipt of a Ph-DATA.indication primitive.

If the MAC sublayer receives the request for an identification cycle (PhICI=ID_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive, it shall assume the ID_Cycle_Request state. If it receives the request for a data cycle (PhICI=data_transfer), it shall assume the Data_Cycle_Request state.

If the MAC sublayer receives the first character of a data sequence DLPDU via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=user_data), it shall take the MACSDU to be transmitted from the Data_Transmit buffer, generate the data sequence DLPDU to be transmitted and assume the Data_Transfer state. While the MACSDU is accepted, any other access to the Data_Transmit buffer is locked.

An error occurred if the MAC sublayer receives a character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface. The MAC sublayer shall assume the CRC_Data_Transfer state in order to start to receive and transmit the check sequence DLPDU.

NOTE The MAC sublayer also assumes this state after power on or a reset.

32.5.4.4.2 Data sequence

ID_Cycle_Request

In this state the MAC sublayer shall request an identification cycle by means of a Ph-DATA.request primitive (PhICI=ID_transfer). The PhL confirms the request with a Ph-DATA.confirm primitive. Before that, the management is informed by means of a MAC_Event.indication primitive (ID_Cycle_Request).

If the MAC sublayer receives the first character of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface, it shall assume the Data_Transfer state. While the MACSDU is accepted.

If the MAC sublayer receives the request for a data cycle (PhICI=data_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive, it shall assume the Data_Cycle_Request state.

An error occurred if the MAC sublayer received a character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface. The MAC sublayer shall assume the CRC_Data_Transfer state in order to begin to receive and transmit the check sequence DLPDU.

Data_Cycle_Request

In this state the MAC sublayer shall request a data cycle by means of a Ph-DATA.request primitive (PhICI=data_transfer). The PhL confirms the request with a Ph-DATA.confirm primitive. Before that, the management is informed by means of a MAC_Event.indication primitive (Data_Cycle_Request).

If the MAC sublayer receives the first character of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface, it shall assume the Data_Transfer state.

If the MAC sublayer receives the request for an identification cycle (PhICI=ID_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive, it shall assume the ID_Cycle_Request state and accept the data from the ID_Transmit_Buffer.

An error occurred if the MAC sublayer received a character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface. The MAC sublayer shall assume the CRC_Data_Transfer state in order to begin to receive and transmit the check sequence DLPDU.

Data_Transfer

In this state, the MAC sublayer shall receive the data sequence DLPDU to be transmitted bit-by-bit via DL-Ph interface to the MAC sublayer by means of Ph-DATA.indication primitives (PhICI=user_data) and on its part transmit the data sequence DLPDU to be sent bit-by-bit by means of Ph-DATA.request primitives (PhICI=user_data) via the DLL-PhL interface to the PhL. The transmission of a character is confirmed to the MAC sublayer with a Ph-DATA.confirm primitive.

The data sequence DLPDU to be sent is transmitted synchronously with the receipt of a data sequence DLPDU, that is, each Ph-DATA.indication primitive with PhICI=user_data causes the sending of a Ph-DATA.request primitive with PhICI=user_data.

If the MAC sublayer receives the beginning of an identification cycle (PhICI=ID_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive, the MAC sublayer shall first terminate the receipt and the transmission of the data sequence DLPDUs and then assume the ID_Cycle_Request state.

If the MAC sublayer receives the beginning of a data cycle (PhICI=data_transfer), the MAC sublayer shall first complete the receipt and the transmission of the data sequence DLPDUs and then assume the Data_Cycle_Request state.

If the MAC sublayer is notified with a Ph-DATA.indication primitive (PhICI=user_data_idle) via the DL-Ph interface that the Data_Idle state was detected on the bus, it completes the receipt and transmission of the data sequence DLPDUs and assumes the Data_Idle state.

If the MAC sublayer receives the first character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface, it shall terminate the receipt and the transmission of the DLPDUs, generate the received MACSDU and assume the CRC_Data_Transfer state.

Data_Idle

If the MAC sublayer receives the character of a data sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=user_data) via the DL-Ph interface, it shall assume the Data_Transfer state and continue to receive and transmit the data sequence of DLPDUs.

If the MAC sublayer receives the beginning of an identification cycle (PhICI=ID_transfer) via the DL-Ph interface by means of a Ph-DATA.indication primitive, the MAC sublayer shall terminate the reception and the transmission of the DLPDUs by means of data sequence. Additional the MAC turns into the ID_Cycle_Request state.

If the MAC sublayer receives the beginning of a data cycle (PhICI=data_transfer), the MAC sublayer shall first terminate the receipt and the transmission of the data sequence DLPDUs and then assume the Data_Cycle_Request state. In addition, the error flag RxSL_Error is set.

If the MAC sublayer receives the first character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface, it shall terminate the receipt and transmission of the DLPDUs, generate the received MACSDU and assume the CRC_Data_Transfer state. In addition, the error flag RxSL_Error is set.

32.5.4.4.3 Check sequence

CRC_Data_Transfer

In this state, the MAC sublayer shall receive the checksum transmitted by means of the Ph-DATA.indication primitive (PhICI=CRC_data) via the DL-Ph interface to the MAC sublayer and on its part transmit the checksum to be sent by means of Ph-DATA.request primitive (PhICI=CRC_data) via the DL-Ph interface to the PhL. The checksums are received and transmitted character by character in a check sequence DLPDU. The transmission of a character is confirmed to the MAC sublayer with a Ph-DATA.confirm primitive.

An error occurred if the MAC sublayer receives a character of a data sequence DLPDU instead of a checksum character with a Ph-DATA.indication primitive (PhICI=User_Data). In this case, the MAC sublayer shall treat the received character like the character of a checksum, transmit on its part the next character to be sent of its checksum by means of a Ph-DATA.request primitive (PhICI=CRC_data) via the DL-Ph interface to the PhL and continue to receive and transmit the checksums.

The transmission of the check sequence DLPDU to be sent is done synchronously with the receipt of the check sequence DLPDU, that is, each Ph-DATA.indication primitive with PhICI=CRC_data causes the sending of a Ph-DATA.request primitive with PhICI=CRC_data.

After the checksums have been completely received and transmitted, the MAC sublayer shall assume the Check_CRC state to compare the received checksum with the one generated by the layer itself.

If the MAC sublayer is notified via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=CRC_data_idle) that the CRC_Data_Idle state was detected on the bus, it interrupts the receipt and the transmission of the check sequence DLPDUs and assumes the CRC_Data_Idle state.

CRC_Data_Idle

In this state, the MAC sublayer shall request the CRC_Data_Idle state on the bus by means of a Ph-DATA.request primitive (PhICI=CRC_data_idle) via the DL-Ph interface. The PhL confirms the request with a Ph-DATA.confirm primitive.

If the MAC sublayer receives a character of a check sequence DLPDU via the DL-Ph interface by means of a Ph-DATA.indication primitive, it shall assume the CRC_Data_Transfer state, if it was a checksum character (PhICI=CRC_data), and continue to transmit the checksums. If it was the first character of a checksum status (PhICI=CRC_status) the MAC sublayer shall assume the CRC_Status_Transfer state and begin to receive and transmit the checksum status.

An error occurred if the MAC sublayer receives a character of a data sequence DLPDU. In this case the MAC sublayer shall treat the received character like the character of a checksum and assume the CRC_Data_Transfer state.

Check_CRC

In this state the MAC sublayer shall compare the received checksum with the one previously generated of the last data sequence DLPDU. If both checksums are identical, the data sequence DLPDU received last was received without errors, and the CRC_receive_error flag assumes the value logical 1. Otherwise, there is a transmission error and the CRC_receive_error flag assumes the value logical 0.

If the MAC sublayer receives the first character of the checksum status (PhICI=CRC_status) via the DL-Ph interface by means of a Ph-DATA.indication primitive, the MAC sublayer shall assume the CRC_Status_Transfer state and begin to receive and transmit the checksum status.

If the MAC sublayer is communicated via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=CRC_data_idle) that the CRC_Data_Idle state was detected on the bus, then it interrupts the receipt and the transmission of the check sequence DLPDUs and assumes the CRC_Data_Idle state.

CRC_Status_Transfer

In this state the MAC sublayer shall receive the checksum status transmitted by means of Ph-DATA.indication primitive (PhICI=CRC_status) via the DL-Ph interface to the MAC sublayer and transmit on its part the checksum status to be sent by means of Ph-DATA.request primitive (PhICI=CRC_status) via the DL-Ph interface to the PhL. The checksum statuses are received and transmitted character by character in a check sequence DLPDU. The transmission of a character is confirmed in the MAC sublayer with a Ph-DATA.confirm primitive.

The check sequence DLPDU to be sent is transmitted synchronously with the receipt of the check sequence DLPDU, that is, each Ph-DATA.indication primitive with PhICI=CRC_status causes the sending of a Ph-DATA.request primitive with PhICI=CRC_status.

After the first four characters of the checksum statuses (r0...r3 and t0...t3) have been completely received, the MAC sublayer shall assume the Check_Receive_Error state in order to transfer the received MACSDU to the MAC-user when the transmission of the data sequence DLPDU received last contained an error.

After the first eight bits of the checksum status (r0...r7 and t0...t7) have been completely received and transmitted, the MAC sublayer shall assume the IBS_Cycle_End state to report the end of a DLPDU cycle to the MAC-user.

If the MAC sublayer is communicated via the DL-Ph interface by means of a Ph-DATA.indication primitive (PhICI=CRC_status_idle) that the CRC_Data_Idle state was detected on the bus, it interrupts the receipt and the transmission of the check sequence DLPDUs and assumes the CRC_Status_Idle state.

Check_Receive_Error

In this state, the MAC sublayer shall evaluate the logic state of the fourth character t3 that was transmitted last of the checksum status to be transmitted. If the binary value logical 1 was transmitted with the character t3, the data sequence DLPDU received last was transmitted without errors. In this case, the MAC sublayer shall generate the MACSDU from the received data sequence DLPDU, make it available to the MAC-user and communicate this event to the user by means of a MAC_Data.indication primitive. If the data sequence DLPDU originates in an identification cycle, the MACSDU is transmitted in the ID_Receive_Buffer, otherwise in the Data_Receive_Buffer.

A transmission error was recognized when the binary value of logical 0 was transmitted with the character t3. The MAC sublayer shall first destroy the data sequence DLPDU received last and, in this case, shall send no message to the MAC-user.

If the MAC sublayer receives a character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_status) via the DL-Ph interface it shall assume the CRC_Status_Transfer state and continue to receive the transmission of the check sequence DLPDUs with the checksum status.

IBS_Cycle_End

In this state the MAC sublayer shall report to its MAC-user the end of a DLPDU cycle on the DL-subnetwork by means of a MAC_Event.indication primitive (IBS_Cycle_End) and then set the checksums for the data sequence DLPDU to be received via the DL-Ph interface and for the data sequence DLPDU to be transmitted to their initial values. In addition the flags CRC_Receive_Error and RxSL_Error are to be set to false.

If the MAC sublayer receives the Ph-DATA.indication primitive (PhICI=ID_Transfer) upon the complete receipt of the check sequence DLPDU, it assumes the ID_Cycle_Request state.

If the MAC sublayer receives the Ph-DATA.indication primitive (PhICI=Data_Transfer) upon the complete receipt of the check sequence DLPDU, it assumes the Data_Cycle_Request state.

If the MAC sublayer receives the Ph-DATA.indication primitive (PhICI=User_Data) upon the complete receipt of the check sequence DLPDU, it assumes the Data_Transfer state.

NOTE This message is used to synchronize the MAC-users of the devices.

CRC_Status_Idle

In this state, the MAC sublayer shall request the CRC_Status_Idle state on the bus by means of a Ph-DATA.request primitive (PhICI=CRC_status_idle) via the DL-Ph interface. The PhL confirms the request with a Ph-DATA.confirm primitive.

If the MAC sublayer receives a character of a check sequence DLPDU by means of a Ph-DATA.indication primitive (PhICI=CRC_status) via the DL-Ph interface, it shall accept the data from the Data_Transmit_Buffer, assume the IBS_Cycle_End state and continue to receive and transmit the check sequence DLPDUs with the checksum status.

32.5.5 MAC-User – MAC interface

32.5.5.1 General

32.5.5 describes the services which the MAC makes available to the MAC-user. Figure 257 shows the interface between the MAC-user and the MAC in the layer model.

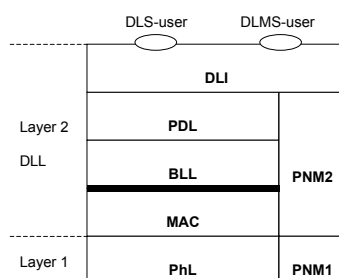


Figure 257 – Interface between MAC-user and MAC in the layer model

32.5.5.2 Overview of the services

The MAC makes the MAC_Cycle service available to the MAC-user. This service is described through the following service primitives:

MAC_Cycle

- MAC_Put_Data.request
- MAC_Put_Data.confirm
- MAC_Get_Data.request
- MAC_Get_Data.confirm
- MAC_Data.request
- MAC_Data.confirm
- MAC_Data.indication

32.5.5.3 Interactions at the MAC-user interface

Figure 258 and Figure 259 show exemplary sequences of interactions at the MAC-user interface for an identification cycle and subsequent data cycle.

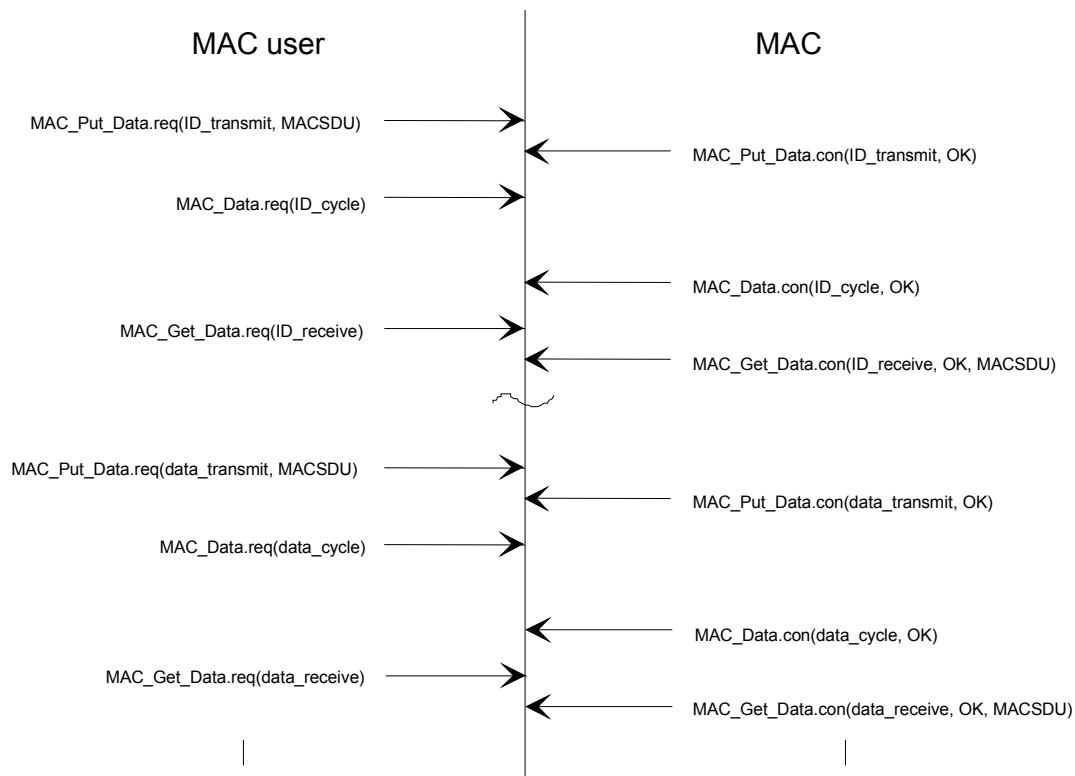


Figure 258 – Interactions at the MAC-user interface (master)

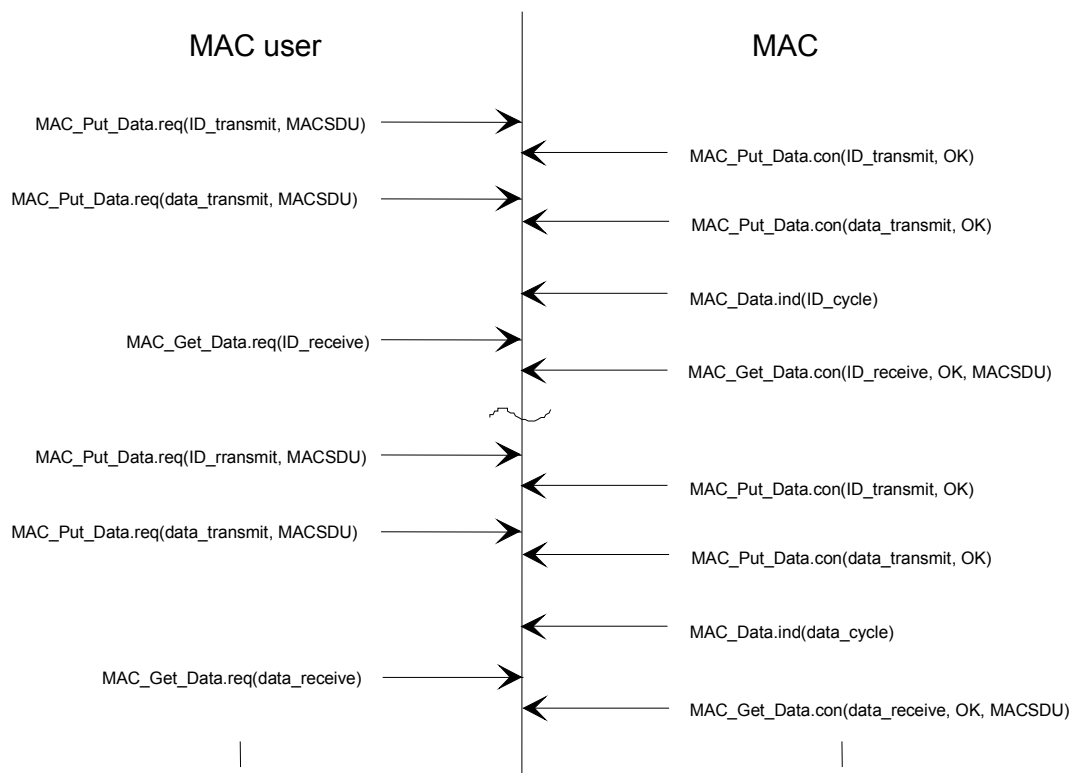


Figure 259 – Interactions at the MAC-user interface (slave)

32.5.5.4 Detailed definitions of the services and interactions

32.5.5.4.1 MAC_Put_Data.request (buffer, MACSDU)

With this service primitive the MAC-user makes the data to be transmitted available to the MAC sublayer. The parameters have the following meanings:

buffer:

This parameter determines the memory area which is to store the data transmitted by the MAC-user. Two memory areas are defined:

buffer = ID_transmit

This memory area is available to the management of the master for the transmission of control codes and the slave for the device codes and is transmitted in an identification cycle.

buffer = data_transmit

This memory area is available for the transmission of user data and is transmitted in a data cycle.

MACSDU:

This parameter contains the data to be transmitted. The data quantity depends on the number and type of devices in the one-total-DLPDU.

32.5.5.4.2 MAC_Put_Data.confirm (buffer, status)

This service primitive is the confirmation of the MAC sublayer for a MAC_Put_Data.request primitive. The parameters have the following meanings:

buffer:

This parameter has the same meaning as for the MAC_Put_Data.request primitive.

status:

This parameter indicates the result of the data transfer and can assume the following values:

status = OK

The data passed on with the MAC_Put_Data.request primitive could be accepted by the MAC sublayer.

status = NO

The data passed on with the MAC_Put_Data.request primitive could **not** be accepted by the MAC sublayer, since the requested memory area was occupied when the data was transmitted.

The MAC-user is responsible for the response to this service primitive.

32.5.5.4.3 MAC_Get_Data.request (buffer)

With this service primitive the MAC-user requests data from the MAC sublayer which has been transmitted from one device via the medium to the MAC sublayer. The data is transmitted by means of a MAC_Get_Data.confirm primitive from the MAC sublayer to the MAC-user.

The parameters have the following meanings:

buffer:

This parameter determines the memory area which contains the data to be transmitted to the MAC-user. Two memory areas are defined:

buffer = ID_receive

This memory area is available to the MAC-user on the slave for the storage of control codes and the master for device codes which were transmitted in an identification cycle.

buffer = data_receive

This memory area is available to the MAC-user for the storage of user data which was transmitted in a data cycle.

32.5.5.4.4 MAC_Get_Data.confirm (buffer, status, MACSDU)

This service primitive is the confirmation of the MAC sublayer to a MAC_Get_Data.request primitive. With this primitive the MAC sublayer transmits the received MACSDU from last DLPDU cycle to the MAC-user.

The parameters have the following meanings:

buffer:

This parameter has the same meaning as for the MAC_Get_Data.request primitive (as specified in 32.5.5.4.3).

status:

This parameter indicates the MAC-user the result of the data transmission and can assume the following values:

status = **OK**

The data could be transmitted to the MAC-user.

status = **NO**

The requested data could **not** be transmitted to the MAC-user.

MACSDU:

If status = OK this parameter contains the data read in during the last message transmission service.

The MAC-user is responsible for responding to this service primitive.

32.5.5.4.5 MAC_Data.request (cycle)

With this service primitive the MAC-user of an active device (master) starts a message DLPDU cycle. The data to be transmitted was previously transferred to the MAC sublayer by means of a MAC_Put_Data.request primitive before.

The parameters have the following meanings:

cycle:

cycle = **ID_cycle**

An identification cycle is started to transmit the control codes to the slaves and to request the device codes of the slaves.

cycle = **data_cycle**

A data cycle for the transmission of user data between the master and the slaves.

32.5.5.4.6 MAC_Data.confirm (cycle, status)

This service primitive is the confirmation of the MAC sublayer for a MAC_Data.request primitive. It indicates that the DLPDU cycle, started by means of the MAC_Data.request primitive, was terminated positively or negatively.

The parameters have the following meanings:

cycle:

This parameter has the same meaning as for the MAC_Data.request primitive.

status:

This parameter indicates whether the requested data transmission could be carried out successfully (status = **OK**), or whether an error occurred during the data transmission (status = **NO**).

The MAC-user is responsible for the response to this service primitive.

32.5.5.4.7 MAC_Data.indication (cycle)

With this service primitive the MAC sublayer indicates to the MAC-user that new data is available from a valid identification cycle or a valid data cycle. The MAC-user can use this data by means of a MAC_Get_Data.request primitive.

The parameter has the following meaning:

cycle:

This parameter indicates whether the received data stems from an identification cycle or a data cycle and in which memory area it is available to the MAC-user. Two values are defined:

cycle = **ID_cycle**

Control and/or ID data was received in an identification cycle. This data is available to the MAC-user in the ID_receive buffer.

cycle = **data_cycle**

User data for the process data channel and the Parameter data channel was received in a data cycle. This data is available to the MAC-user in the data_receive buffer.

The MAC-user is responsible for responding to this service primitive.

32.5.6 MAC-PNM2 interface

32.5.6.1 General

The management of the MAC is part of the MAC that provides the management functionality of the MAC requested by the PNM2. The management of the MAC handles the initialization, the monitoring and the error recovery in the MAC.

32.5.6 defines the administrative MAC management services which are available to the PNM2, together with their service primitives and the associated parameters. Figure 260 shows the interface between MAC and PNM2 in the layer model.

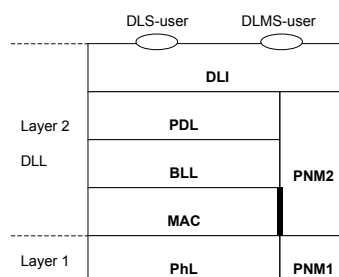


Figure 260 – Interface between MAC and PNM2 in the layer model

The service interface between MAC and PNM2 makes the following functions available:

- Reset of the MAC
- Request and change of the current operating parameters of the MAC
- Indication of unexpected events, errors, and status changes which occurred or were detected in the MAC

32.5.6.2 Overview of the services

The MAC makes the following services available to the PNM2:

- Reset MAC
- Set Value MAC or Get Value MAC
- Event MAC

The MAC services are described by primitives (beginning with MAC_...):

32.5.6.2.1 Reset MAC

The PNM2 uses this required service to reset the MAC. The reset is equivalent to power on. Upon execution, the PNM2 receives a confirmation.

Service primitives:

- MAC_Reset.request
- MAC_Reset.confirm

32.5.6.2.2 Set Value MAC

The PNM2 uses this optional service to set a new value to the MAC variables. Upon completion, the PNM2 receives a confirmation from the MAC whether the defined variables assumed the new value.

Service primitives:

- MAC_Set_Value.request
- MAC_Set_Value.confirm

32.5.6.2.3 Get Value MAC

The PNM2 uses this optional service to read the variables of the MAC. The current value of the defined variable is transmitted in the response of the MAC.

Service primitives:

- MAC_Get_Value.request
- MAC_Get_Value.confirm

32.5.6.2.4 Event MAC

The MAC uses this required service to inform the MAC-user about certain events or errors in the MAC.

Service primitive:

- MAC_Event.indication

32.5.6.3 Overview of the interactions

Figure 261 and Figure 262 show the time relations of the services primitives:

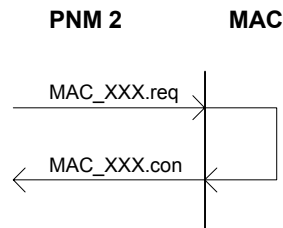


Figure 261 – Reset, Set Value and Get Value MAC services

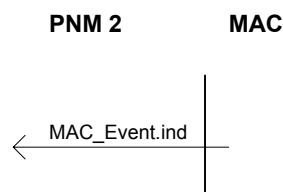


Figure 262 – Event MAC service

32.5.6.4 Detailed definitions of the services and interactions

32.5.6.4.1 MAC_Reset

The MAC_Reset service is mandatory. The PNM2 transfers a MAC_Reset.request primitive to the MAC to reset it (see Table 275).

Table 275 – MAC_Reset

Parameter name	Request	Confirm
Argument	M	
Result(+)		M

32.5.6.4.2 MAC_Set_Value

The MAC_Set_Value service is optional. The PNM2 transfers a MAC_Set_Value.request primitive to the MAC in order to set a defined MAC variable to a desired value. After receipt of this primitive, the MAC tries to select the variable and to set the new value. Upon completion, the MAC transmits a MAC_Set_Value.confirm primitive to the PNM2 (see Table 276).

Table 276 – MAC_Set_Value

Parameter name	Request	Confirm
Argument variable_name desired_value	M M M	
Result(+)		M

variable_name:

This parameter defines the MAC variable which is set to a new value.

desired_value:

This parameter declares the value for the new MAC variable.

Table 277 provides information on which MAC variables may be set to which new values.

Table 277 – MAC variables

Name of MAC variable	Value range	Remarks
Loopback_word (LBW)	Bit 15 = 1, Bit 14 to Bit 4 are used for setting the LBW, Bit 3 to Bit 0 are used as DLPDU counter (see 32.5.3.2)	master side only
Time_timeout	Value will be defined from the system management and depends from the actual configuration of the DL-subnetwork	master side only

32.5.6.4.3 MAC_Get_Value

The MAC_Get_Value service is optional. The PNM2 transfers a MAC_Get_Value.request primitive to the MAC to read out the current value of a specified MAC variable. After receipt of this primitive, the MAC tries to select the specified variable and to transmit its current value to the PNM2 with a MAC_Get_Value.confirm primitive (see Table 278).

Table 278 – MAC_Get_Value

Parameter name	Request	Confirm
Argument variable_name	M M	
Result(+) current_value		M M

variable_name:

This parameter defines MAC variable the value of which is to be read out.

desired_value:

This parameter contains the read-out value of the MAC variable.

The MAC variables to be read are exactly those variables that can be written to with the MAC_Set_Value.

32.5.6.4.4 MAC_Event

The MAC_Event service is mandatory. The MAC transfers a MAC_Event.indication primitive to the PNM2 to inform it about important events or errors in the MAC (see Table 279).

Table 279 – MAC_Event

Parameter name	Indication
Argument event	M M

event:

This parameter defines the event which occurred or the error source in the MAC and can assume the following values (see Table 280):

Table 280 – MAC_Event

Name	Meaning	Mandatory/optional
data_cycle_request	The request for a data cycle was detected on the transmission medium.	O
ID_cycle_request	The request for an identification data cycle was detected on the transmission medium.	O
IBS_cycle_end	A DLPDU cycle is ended.	M
data_noise	Before an identification or data cycle was started, characters of a data sequence DLPDU had been received. (bus master only)	M
CRC_noise	Before an identification or data cycle was started, characters of a check sequence DLPDU had been received. (bus master only)	M

32.6 Peripherals network management for layer 2 (PNM2)

32.6.1 Functionality of the PNM2

The management for layer 2 (PNM2) handles the initialization, the monitoring and the error recovery between PNM2-user and the logical functions in the MAC, BLL and PDL (see Figure 263).

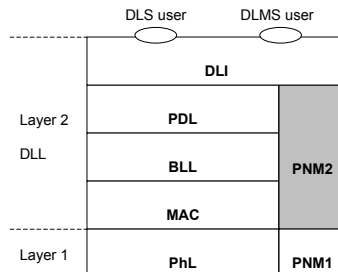


Figure 263 – Location of the PNM2 in the DLL

32.6.2 PNM2-User-PNM2 interface

32.6.2.1 General

32.6.2 defines the administrative PNM2 (Peripherals Network Management for the layer 2) services which are available to the PNM2-user, together with their service primitives and the associated parameters. Figure 264 shows the interface between PNM2-user and PNM2 in the layer model.

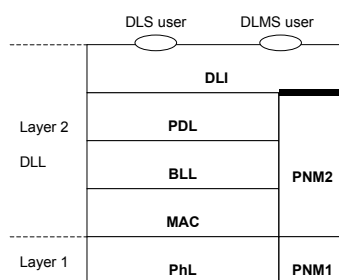


Figure 264 – Interface between PNM2-user and PNM2 in the layer model

The service interface between PNM2-user and PNM2 makes the following functions available:

- Reset of the layer 2 (local)
- Request and change of the current operating parameters of PDL, BLL, MAC (local)
- Indication of unexpected events, errors, and status changes (local and remote)
- Read-out of the active DL-subnetwork configuration
- Read-out of the current DL-subnetwork configuration
- Setting of a certain DL-subnetwork configuration

32.6.2.2 Overview of the services

32.6.2.2.1 Available services

The PNM2 makes the PNM2-user the following services available:

- Reset PNM2
- Set Value PNM2 or Get Value PNM2
- Event PNM2
- Get Current Configuration PNM2 (master only)
- Get Active Configuration PNM2 (master only)
- Set Active Configuration PNM2 (master only)

32.6.2.2.2 Reset PNM2

The PNM2-user uses this required service to cause the PNM2 to reset Layer 2 (DLL) and itself. The reset is equivalent to power on. The PNM2-user receives a confirmation for this service.

Service primitives:

- PNM2_Reset.request
- PNM2_Reset.confirm

32.6.2.2.3 Set value PNM2

The PNM2-user uses this optional service to set a new value to the variables of the layers 1 or 2. It receives a confirmation on whether the defined variables assumed the new value.

Service primitives:

- PNM2_Set_Value.request
- PNM2_Set_Value.confirm

32.6.2.2.4 Get value PNM2

The PNM2-user uses this optional service to read out variables of the layer 2. The current value of the defined variable is transferred in the response of the PNM2.

Service primitives:

- PNM2_Get_Value.request
- PNM2_Get_Value.confirm

32.6.2.2.5 Event PNM2

The PNM2 uses this required service to inform the PNM2-user on certain events or errors in the layer 2.

Service primitive:

- PNM2_Event.indication

32.6.2.2.6 Get current configuration PNM2 (master only)

The PNM2-user of the master uses this required service to read out the current DL-subnetwork configuration.

Service primitives:

- PNM2_Get_Current_Configuration.request
- PNM2_Get_Current_Configuration.confirm

32.6.2.2.7 Get active configuration PNM2 (master only)

The PNM2-user of the master uses this required service to read out the active DL-subnetwork configuration.

Service primitives:

- PNM2_Get_Active_Configuration.request
- PNM2_Get_Active_Configuration.confirm

32.6.2.2.8 Set active configuration PNM2 (master only)

The PNM2-user of the master uses this required service to set a certain DL-subnetwork configuration.

Service primitives:

- PNM2_Set_Active_Configuration.request
- PNM2_Set_Active_Configuration.confirm

32.6.2.3 Overview of the interactions

The PNM2 services are described by the following primitives (beginning with PNM2_...):

Figure 264, Figure 265, Figure 266 and Figure 267 show the time relations of the services primitives:

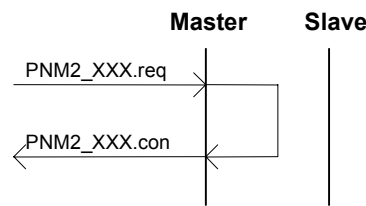


Figure 265 – Reset, Set Value, Get Value and Get Active Configuration services

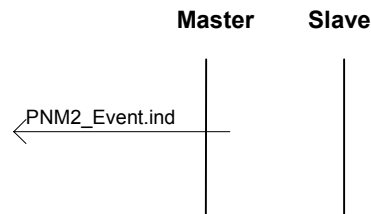


Figure 266 – Event PNM2 service

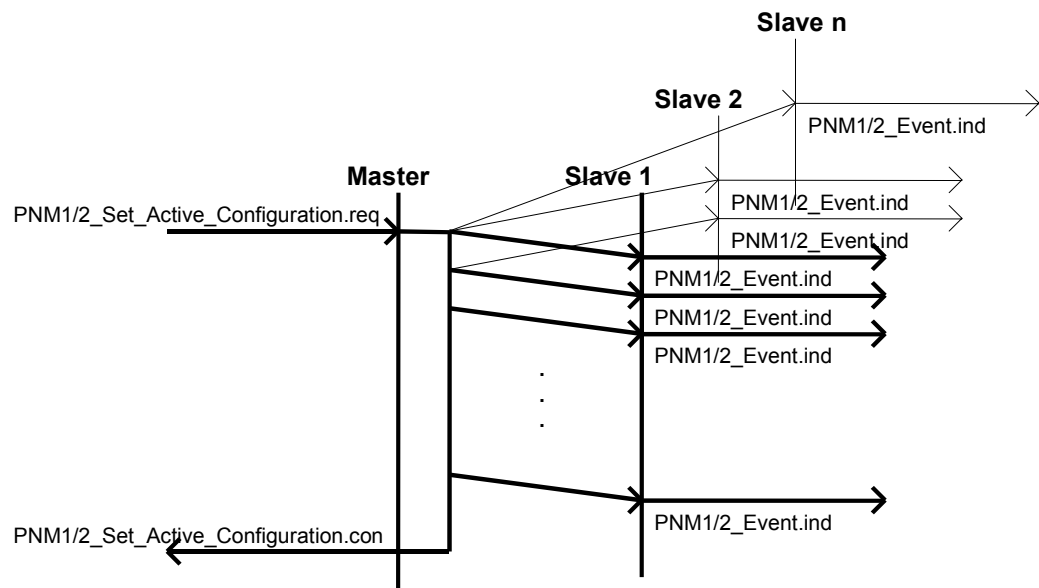


Figure 267 – Set Active Configuration, Get Current Configuration service

32.6.2.4 Detailed definition of the services and interactions

32.6.2.4.1 PNM2_Reset

The PNM2_Reset service is mandatory. The PNM2-user passes a PNM2_Reset.request primitive to the PNM2 to cause it to reset the layer 2.

After the confirmations of the PDL, BLL and MAC by means of corresponding confirmation primitives the PNM2 resets itself and communicates a PNM2_Reset.confirm primitive to the PNM2-user (see Table 281).

Table 281 – PNM2_Reset

Parameter name	Request	Confirm
Argument	M	
Result(+) M_status (=OK)		S M
Result(-) M_status (NOK)		S M

M_status:

This parameter contains a confirmation on the execution of the service. The following possible values are defined (see Table 282):

Table 282 – M_status values of the PNM2_Reset

Value	Meaning
OK	Positive confirmation; the reset function was carried out successfully.
NOK	Failure

32.6.2.4.2 PNM2_Set_Value

The PNM2_Set_Value-Service is optional. The PNM2-user transfers a PNM2_Set_Value.request primitive to the PNM2 to set a defined variable of the layer 2 to a requested value. The management transmits the individual PDL, BLL, MAC and/or Ph-SET-VALUE.request primitives to the corresponding layers and sends a PNM2_Set_Value.confirm primitive to the PNM2-user after it has received all associated confirmation primitives (see Table 283).

Table 283 – PNM2_Set_Value

Parameter name	Request	Confirm
Argument variable_name desired_value	M M M	
Result(+) M_status (=OK)		S M
Result(-) M_status (≠OK)		S M

variable_name:

This parameter contains a variable of the PDL, BLL or MAC. The selectable variables are defined in the corresponding subclauses of the individual layers.

desired_value:

This parameter contains a value for the selected variable. The permitted values or value ranges are determined in the corresponding subclauses of the individual layers.

M_status

This parameter contains a confirmation on the execution of the service. The following possible values are determined (see Table 284):

Table 284 – M_status values of the PNM2_Set_Value

Value	Meaning
OK	Positive confirmation; the variable has the new value.
NO	The variable does not exist or could not assume the new value.
IV	Invalid parameters in the request

32.6.2.4.3 PNM2_Get_Value

The PNM2_Get_Value service is optional. The PNM2-user transfers a PNM2_Get_Value.request primitive to the PNM2 in order to read out the current value of a specified variable of the layer 2. The management transfers the individual PDL, BLL, MAC and/or Ph_Get_Value.request primitives to the corresponding layers and sends a PNM2_Get_Value.confirm primitive with requested values to the PNM2-user after all associated confirmation primitives have been received (see Table 285).

Table 285 – PNM2_Get_Value

Parameter name	Request	Confirm
Argument variable_name	M M	
Result(+) current_value M_status (=OK)		S M M
Result(-) M_status (≠OK)		S M

variable_name:

This parameter contains a variable of the PDL, BLL, MAC or PhL. The selectable variables are defined in the corresponding subclauses of the individual layers.

current_value:

This parameter receives the current value for the selected variable.

M_status:

This parameter contains a confirmation about the execution of the service. The following possible values are defined (see Table 286).

Table 286 – M_status values of the PNM2_Get_Value

Value	Meaning
OK	Positive confirmation; the variable could be read.
NO	The variable does not exist or could not be read.
IV	Invalid parameters in the request

32.6.2.4.4 PNM2_Event

The PNM2_Event-Service is mandatory. After receipt from PDL, BLL and MAC a indication the PNM2 transfers a PNM2_Event.indication primitive to the PNM2-user to inform it about important events or errors in the layers. After DL-subnetwork errors have been reported, the PNM2 carries out a configuration check. If the configuration differs from the configuration prior to the DL-subnetwork error, the PNM2 automatically generates an event with information on the configuration change (see Table 287)

Table 287 – PNM2_Event

Parameter name	Indication
Argument	M
event	M
add_info	C

event:

This parameter defines the event which occurred or the error cause. The possible values are defined in the corresponding subclauses of the respective layers. The possible values of the errors which occurred in the PNM2 are defined in Table 288.

Table 288 – MAC Events

Name	Meaning
Configuration_change	The configuration of the DL-subnetwork changed during operation.

add_info:

This parameter contains additional information about the events or errors which occurred.

32.6.2.4.5 PNM2_Get_Current_Configuration

The PNM2-user of the master uses this service to read out the current configuration of the DL-subnetwork. To do so, the PNM2 carries out ID cycles to detect the currently connected slaves and transfers the detected configuration to the PNM2-user in the current_configuration parameter. The configuration of the DL-subnetwork after the service has been executed can be determined with the network_configuration parameter(see Table 289).

Table 289 – PNM2_Get_Current_Configuration

Parameter name	Request	Confirm
Argument	M	
network_configuration	M	
Result(+)		S
current_configuration		M
Result(-)		S
error_type		M

current_configuration:

This parameter contains the current configuration of the DL-subnetwork. The parameter is equivalent to the active_configuration parameter of the Get_Active_Configuration service.

network_configuration:

This parameter defines the configuration of the DL-subnetwork after the service has been carried out.

Closed: The outgoing interfaces of all slaves are closed.

Open: The outgoing interfaces of all slaves are open.

error_type:

This parameter specifies why the service could not be executed successfully. Possible error causes:

- An error was detected when a ring segment was connected.
- No ID cycles could be run (DL-subnetwork error).

add_info:

This parameter provides additional information about the error cause (for example, ring segment number when the outgoing interface could not be opened)

32.6.2.4.6 PNM2_Get_Active_Configuration

The PNM2-user of the master uses this service to read out the active configuration of the DL-subnetwork. The PNM2 transfers the currently active configuration in the active_configuration parameter to the PNM2-user. To provide the service, the PNM2 does not need to run ID cycles. The service is locally responded to. The PNM2 logs all changes of the configuration, so that the active_configuration parameter which is kept locally is always up to date (see Table 290).

Table 290 – PNM2_Get_Active_Configuration

Parameter name	Request	Confirm
Argument	M	
Result(+) active_configuration		S M
Result(-) error_type		S M

active_configuration:

This parameter contains the active configuration of the DL-subnetwork. The entries in the list are ordered according to the physical order of the slaves in the ring. The parameter has the following structure according to Figure 268:

ID code of the 1st slave	Ring segment level of the 1st slave
ID code of the 2nd slave	Ring segment level of the 2nd slave
...	...
ID code of the nth slave	Ring segment level of the nth slave

Figure 268 – The active_configuration parameter**error_type:**

This parameter indicates why the service could not be executed successfully.

32.6.2.4.7 PNM2_Set_Active_Configuration

The PNM2-user of the master uses this service to generate a certain active configuration of the DL-subnetwork. The PNM2 converts the target configuration in control commands for the switching on or off of certain ring segments. If the new configuration cannot be accepted, the exact error cause is communicated to the PNM2-user and the old configuration is retained (see Table 291).

Table 291 – PNM2_Set_Active_Configuration

Parameter name	Request	Confirm
Argument active_configuration	M M	
Result(+)		S
Result(-) error_type add_info		S M C

active_configuration:

This parameter contains the new active configuration of the DL-subnetwork to be generated. The structure corresponds to the structure of the Active_Configuration parameter of the Get_Active_Configuration service.

error_type:

This parameter indicates why the service could not be executed successfully. Possible error causes:

- An error was detected when a ring segment was connected to the ring. The new configuration could not be generated.
- No ID cycles could be run; a fatal bus error.

add_info:

This parameter provides additional information about the error cause (for example, ring segment number when the outgoing interface could not be opened).

32.7 Parameters and monitoring times of the DLL

The DLL parameters and times described below are used to monitor the DL-subnetwork operation in the DLL of the master. The monitoring times are measured either in seconds (s) or in the number of bus cycles.

32.7.1 PDL parameters

32.7.1.1 SPA_acknowledge_timeout $T_{TO\_SPA\_ACK}$

The SPA_acknowledge_timeout is the time which the local PDL waits for the associated PDL_Data_Ack.confirm primitive after the sending of a PDL_Data_Ack.request primitive. If there is a timeout, the local PDL shall attempt up to *max_spa_retry*-times to send the DLSDU to the remote PDL. If no attempt was successful, the PDL shall return a negative acknowledgement to the user. In addition, this communication relationship shall be locally disconnected and the PDL shall attempt to synchronize itself again with the corresponding PDL of the communication partner.

The SPA_acknowledge_timeout can be calculated as follows:

$$t_{TO\_SPA\_ACK} = (DIST + add\_wait) * t_{UP}$$

where

DIST is a constant number of 5 bus cycles;

add_wait is an additional redundancy of 1 to 4 bus cycles;

t_{UP} is the update time.

32.7.1.2 max_spa_retry

This DLL parameter specifies the maximum number of repeated attempts to send SPA PDUs. It can be parameterized by means of a PNM2_Set_Value.request primitive.

Value range : 0, 2, 4, 6 ... 14

32.7.1.3 max_swa_count

This DLL parameter specifies the maximum number of successive data cycles with errors which are allowed before the PDL protocol machine reports a multiple data cycle error and carries out a synchronization with the protocol machine of the remote device.

Value range : 0 ... 255

32.7.2 BLL parameters

32.7.2.1 update_time t_{UP}

The update time is the time which passes between two starts of bus cycles. By setting the update time with a PNM2_Set_Value.request primitive the time-equidistance of the DL-subnetwork can be obtained. The update time shall be greater or equal to the bus cycle time (except zero) and is preset by the system to the value zero (default).

The value zero means that the update time is not defined. Thus, the automatic start of a bus cycle merely depends on the end of the previous bus cycle and the complete processing of the PDL protocol machines and not of the timeout of the update timer. That means, the time aquidistance is deactivated by the default setting.

Parameter size: 4 octets

Settable values: t_{UP} × 0,1 ms

32.7.2.2 bus_timeout t_{TO_BUS}

The bus timeout is the maximum time which may pass between two valid data cycles. If this time is exceeded, there is a fatal bus error, which could not be repaired independently (for example, environment with strongly interference or broken cable). The bus timeout can be parameterized with a PNM2_Set_Value.request primitive. If the bus timeout is set to the value zero, the bus monitoring is disabled.

Parameter size: 4 octets

Settable values: t_{TO_BUS} × 1 ms

32.7.3 MAC parameters

32.7.3.1 Device code

Figure 269 shows the structure of the device code:

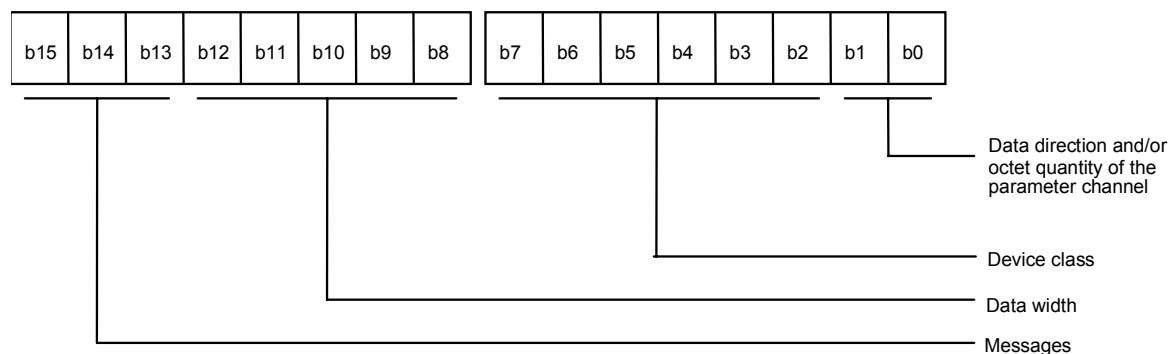


Figure 269 – Device code structure

Bits 0 and 1 have to be interpreted differently for devices with or without parameter channel. For devices without parameter channel the bits indicate the direction of the user data. For devices with parameter channel the bits indicate the number of octets which are used for the parameter channel.

Bits 6 and 7 of the device code distinguish whether the device has a parameter channel or not. For devices with a parameter channel the bits 6 and 7 shall have only the value combination Bit6 =1 and Bit7 =1.

32.7.3.2 Data direction (bit 0 and bit 1 ≠ 1)

If bits 6 and 7 ≠ 1, the bits indicate whether the device occupies input and/or output addresses (see Table 292).

Table 292 – Data direction

Bit 1	Bit 0	Meaning
0	0	No data address (for example, bus coupler)
0	1	Only output addresses occupied
1	0	Only input addresses occupied
1	1	Input and output addresses occupied

32.7.3.3 Number of octets occupied in the parameter channel (bit 7 = 1 and bit 6 = 1)

If bits 7 and 6 are both 1, bits 1 and 0 indicate how many octets of the parameter channel the device occupies (see Table 293).

Table 293 – Number of the occupied octets in the parameter channel

Bit 1	Bit 0	Number of occupied words of the parameter channel
0	0	4 octets
0	1	8 octets
1	0	Reserved
1	1	2 octets (standard)

32.7.3.4 Device class

Certain bit combinations of the bits 2 through 7 indicate the device class (see Table 294). The other combinations are reserved for the identification of device functions. These specifications should be described in device profiles.

Table 294 – Device class

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Device class
0	0	0	0	1	0	0	0	Bus coupler with local bus branch
0	0	0	0	1	1	0	0	Bus coupler with remote bus branch
0	0	0	0	1	0	1	1	Bus coupler with I/O data
0	1	1	1	1	1	x	x	Analog local bus device
1	0	1	1	1	1	x	x	Digital local bus device
1	1	0	1	1	1	x	x	Local bus device with parameter channel
0	0	0	0	0	0	x	x	Digital remote bus device
0	0	1	1	0	0	x	x	Analog remote bus device
1	1	1	1	0	0	x	x	Remote bus device with parameter channel
x "don't care"								

32.7.3.5 Control data

Bits 13 to 15 return control data from the device to the master (see Table 295).

Table 295 – Control data

Bit 15	Bit 14	Bit 13	Meaning
x	x	1	Reserved
x	1	X	CRC receive error
1	x	X	Reserved
x "don't care"			

32.7.3.6 Data width

The data width specifies how many bits the device occupies on the bus. If a device has, for example, 16 bit inputs and 32 bit outputs, it occupies 32 bit (4 octets) in the ring (the higher value is decisive) (see Figure 270 and Table 296).

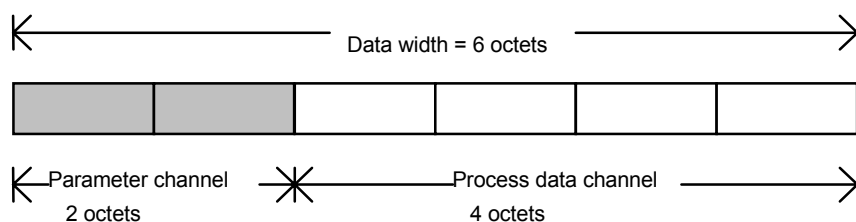


Figure 270 – Relations between data width, process data channel and parameter channel

Table 296 – Data width

Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Data width
0	0	0	0	0	0
0	1	1	0	0	1 bit
0	1	1	0	1	2 bits
0	1	0	0	0	4 bits
0	1	0	0	1	1 octet
0	1	0	1	0	12 bits
0	0	0	0	1	2 octets
0	1	0	1	1	3 octets
0	0	0	1	0	4 octets
0	0	0	1	1	6 octets
0	0	1	0	0	8 octets
0	0	1	0	1	10 octets
0	1	1	1	0	12 octets
0	1	1	1	1	14 octets
0	0	1	1	0	16 octets
0	0	1	1	1	18 octets
1	0	1	0	1	20 octets
1	0	1	1	0	24 octets
1	0	1	1	1	28 octets
1	0	0	1	0	32 octets
1	0	0	1	1	48 octets
1	0	0	0	1	52 octets
1	0	1	0	0	64 octets
1	0	0	0	0	Reserved
1	1	x	x	x	Reserved

x = "don't care"

32.7.3.7 Control code

Figure 271 shows the structure of the control code:

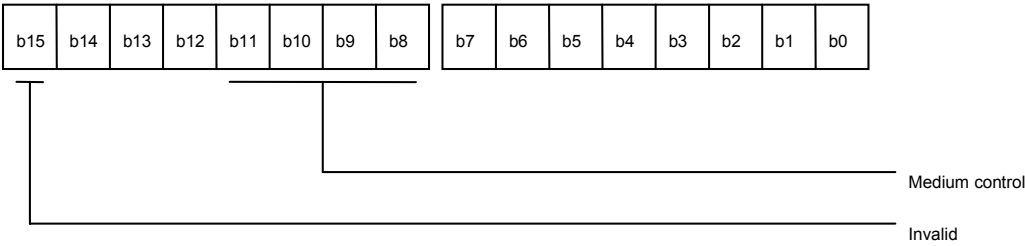


Figure 271 – Structure of the control code

32.7.3.8 Invalid

Bit 15 defines whether the control code is effective. If bit 15 equals 0, the code is effective.

32.7.3.9 Medium control (Bit 8 to Bit 11)

Bits 8 to 11 control the MAU of the outgoing interfaces (see Table 297).

Table 297 – Medium control

Bit 11	Bit 10	Bit 9	Bit 8	Meaning
X	x	X	1	Reset of the ring segment which is connected to the outgoing interface 1
X	x	1	X	Reset of the ring segment which is connected to the outgoing interface 2
X	1	X	X	Outgoing interface 1 disabled
1	x	X	x	Outgoing interface 2 disabled
x "don't care"				
NOTE The remaining bits of the control code are reserved.				

Annex A (informative)

Types 1, 2, 3 (synchronous), 5 (first FCS), 6, 7 and 8: Exemplary FCS implementations

This annex provides an example implementation of FCS generation and FCS syndrome checking for Types 1, 6 and 7, for Type 3 when used with a synchronous PhL, and for the first of the two frame check sequences of Type 5. Similar considerations with a different feedback pattern and a different expected residual apply for Type 2.

Similar preset and feedback patterns for FCS generation, but without complementation of the FCS during normal transmission or residual checking on reception, apply to Type 8.

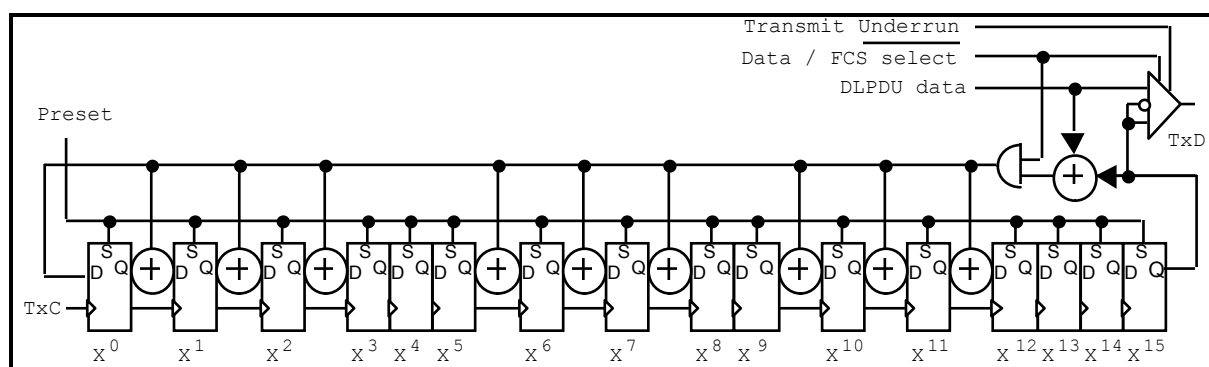


Figure A.1 – Example of FCS generation for Types 1, 3 (synchronous), 5 (first FCS), 6 and 7

In this example, shown in Figure A.1, the FCS is computed in a register consisting of 16 presettable master-slave flip-flops which are interconnected as a linear feedback shift register, with its least significant bit depicted on the left. The initial preset of the register before transmission serves to include the initial $L(X)$ term in the FCS computation. Feedback is disabled during transmission of the FCS itself, and the FCS is transmitted complemented to provide the final $L(X)$ inclusion in the FCS computation. Also shown is optional logic to inhibit the final complementation and transmit a massively incorrect FCS in the case of a transmitter underrun.

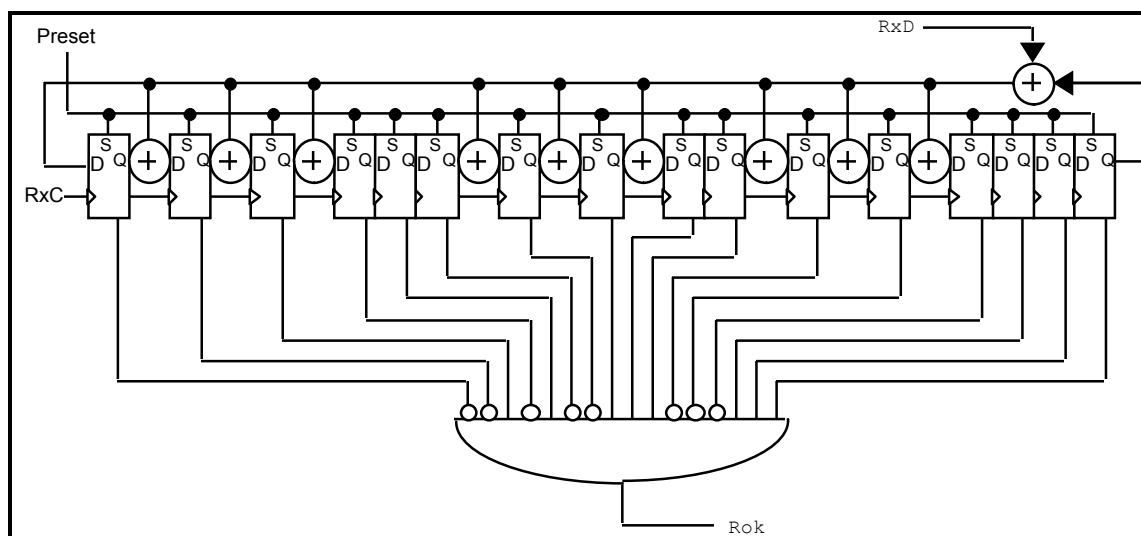


Figure A.2 – Example of FCS syndrome checking on reception for Types 1, 3 (synchronous), 5 (first FCS), 6 and 7

In this example, shown in Figure A.2, the residual FCS is computed in a similar register. The Q outputs of the 16 flip-flops are compared to the expected residual value by the 16-input “and” gate, half of whose inputs are complemented.

In this example, the FCS is computed in a register consisting of 16 presettable master-slave flip-flops which are interconnected as a linear feedback shift register, with its least significant bit depicted on the left. The initial pre-set of the register before transmission serves to include the initial $L(X)$ term in the FCS computation. Feedback is disabled during transmission of the FCS itself, and the FCS is transmitted complemented to provide the final $L(X)$ inclusion in the FCS computation. Also shown is optional logic to inhibit the final complementation and transmit a massively incorrect FCS in the case of a transmitter underrun.

Annex B (informative)

Type 1: Formal protocol finite state machines

NOTE 1 This annex specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. This specification is complementary to, and subordinate to, the textual specification in the body of this standard.

NOTE 2 The finite state machine descriptions given here are necessarily less than a complete description of an implementation. Additional requirements and considerations are found in the textual specification.

B.1 Basic reception and transmission FSMs

B.1.1 Nomenclature

$X \Leftarrow Y$	assignment of value of expression Y to name-expression X
$X \wedge Y$	logical AND of expressions X and Y with short-circuit evaluation (Y is evaluated only if X is TRUE)
$X \vee Y$	logical OR of expressions X and Y with short-circuit evaluation (Y is evaluated only if X is FALSE)
$! Y$	logical NOT of value of expression Y
$\exists(X, Y)$	THERE EXISTS AN X SUCH THAT Y — which is true if an object of the type of name-expression X satisfying Boolean condition Y exists, and false otherwise. If an object of the type of X satisfying Y exists, then one such object (usually the only one) is bound to the name-expression X for use throughout subsequent parts of the transition condition and the transition's actions.
$X \in \{ Y \dots \}$	CONTAINED IN — X is a member of the set of elements Y ...
$X \notin \{ Y \dots \}$	NOT CONTAINED IN — X is not a member of the set of elements Y ...
$X . Y$	field selection — the Y th component of name-expression X
$(X) \bullet$	pointer dereference (indirection) — the object to which name-expression X points

events, variables, procedures, and other FSM nomenclature

FCS_OK	FCS condition at RxA\ of just-received DLPDU; false if no DLPDU just received
C(CT)	8-bit down-counter, prescaled by V(ST), for LAS measurement of periods of bus inactivity: "Token Recovery" — preset with P(TRD), 15, or V(TN)
C(RR)	8-bit down-counter, prescaled by V(ST), for token-holder measurement of periods of bus inactivity: "Reply Recovery" — preset with V(MRD)+1
C(FD)	8-bit down-counter, prescaled by V(ST), for responder measurement of reply-forming delay: "Forming Delay" — preset with V(MRD)
V(TL)	16-bit link identifier: "This Link"
V(TN)	8-bit node identifier: "This Node"

V(DTA)	32-bit DL-address used to identify the current token holder: “Delegated Token Address”
V(DA)	32-bit DL-address used to retain the last DLPDU's “Destination Address” if a CA, CD, ED or RQ DLPDU, else null (invalid)
V(SA)	32-bit DL-address used to retain the last DLPDU's “Source Address” if a CA, CD, ED or RQ DLPDU, else null (invalid)
V(LN)	8-bit node identifier used to record the node-id of the current LAS: “LAS Node”
V(TT)	8-bit FC as “Token Type”: TL, ES, PT, CA, CD, ED
NOTE CA, CD, ED are considered to be half-transaction tokens).	
V(NTS)	16-bit node-time source link number
V(NTO)	56-bit DL-time offset with respect to sending DLE's node-time
C(NT)	56-bit frequency-corrected up-counter: “Node Time”
C(UC)	16-bit down-counter: “Unscheduled Capacity” used by LAS to generate ES and PT DLPDUs
C(RD)	16-bit down-counter: “Remaining Duration” used by token-holder to limit use of ES-token
Q(LR)	prioritized FIFO queue of unscheduled SPDUs for transmission to the local link's LAS
QN(US)	prioritized FIFO queue of unscheduled service requests for the DLE
QX.is_empty	Boolean indicating that the specified queue is or is not empty
QX(Y,min=PP)	subset of QX(Y) whose priority is PP or higher
rcv	record holding parsed version of just-received DLPDU
next	record holding parsed version of DLPDU in the selected transmit scheduling queue
reply	record holding constructed immediate reply parameters
xxx.yyy	field yyy of record xxx
xxx.FC.F	Boolean field of FC field of xxx DLPDU which indicates final transaction of token use
xxx.FC.DA	DA field is present in xxx DLPDU — determination based on FC value
xxx.FC.SA	SA field is present in xxx DLPDU — determination based on FC value
xxx.DA.exists	record xxx contains a DA field
xxx.SA.exists	record xxx contains an SA field

xxx.is_group variable xxx specifies a group DL-address

xxx.is_DLSAP_addr variable xxx specifies a DLSAP-address

xxx.is_DLCEP_id variable xxx specifies the DL-identifier of a DLCEP

is_usable(xxx) the implementation is capable of operation with the value of parameter xxx

falling_edge(xxx) event detecting falling (true → false) edge of signal xxx

dln_queue_deliver report received DLPDU to DL-management for processing

dln_event notify DL-management of specified event

process_xxx process specified type of received DLPDU

send_xxx(...) send specified DLPDU with specified explicit arguments and appropriate values for other arguments

send_item send specified DLPDU from parametric description, providing current values for the C(NT)-dependent arguments on TD, RQ and RR DLPDUs

local_addr DLE's address-recognition function — recognizes all but multi-peer subscriber DLCEPs

subscriber DLE's other address-recognition function — recognizes only multi-peer subscriber DLCEPs

create_object(X) create an object of the type of X and bind it to the name local name X

delete_object delete specified object

activate_addr_recognition activate recognition of the specified address

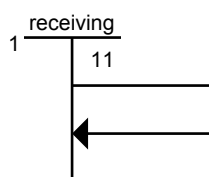
local_object local object addressed by specified DL-address, for which local_addr is true

FSM_state(FSM_name, FSM_state_number) Boolean condition, true if specified FSM is in specified state

FSM_transition(FSM_name, transition_state_number_pair) event, true when specified FSM executes the specified transition (from the first to the second specified state)

B.1.2 The receiver FSM

receiver FSM



receiver FSM

state	name	transitions	entry actions

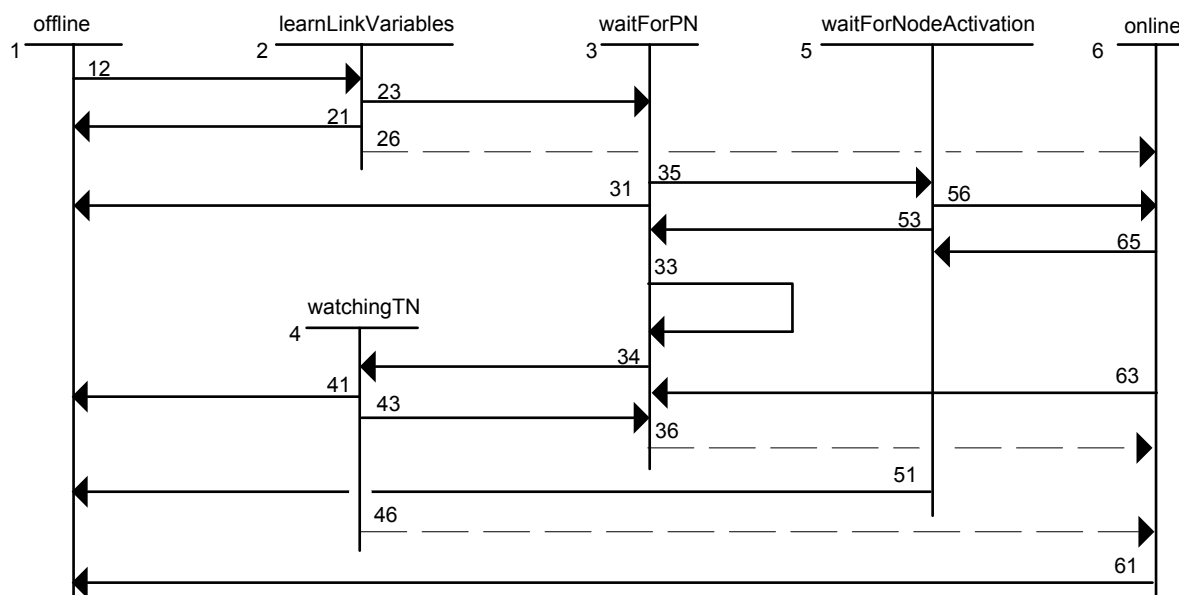
1	receiving	11	

transition condition : [actions]

11a	$\text{rcv.FC} = \text{EC} \wedge ((\text{rcv.FC.DA} \wedge \text{local\_addr}(\text{rcv.DA}) \vee (!\text{rcv.FC.DA} \wedge \text{subscriber}(\text{rcv.SA})))$: process_EC(rcv); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;
11b	$\text{rcv.FC} = \text{DC} \wedge ((\text{rcv.FC.DA} \wedge \text{local\_addr}(\text{rcv.DA}) \vee (!\text{rcv.FC.DA} \wedge \text{subscriber}(\text{rcv.SA})))$: process_DC(rcv); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;
11c	$\text{rcv.FC} = \text{RC} \wedge ((\text{rcv.FC.DA} \wedge \text{local\_addr}(\text{rcv.DA}) \vee (!\text{rcv.FC.DA} \wedge \text{subscriber}(\text{rcv.SA})))$: process_RC(rcv); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;
11d	$\text{rcv.FC} = \text{CA} \wedge \text{local\_addr}(\text{rcv.DA})$: process_CA(rcv); V(DA) $\leftarrow$ rcv.DA; V(SA) $\leftarrow$ rcv.SA;
11e	$\text{rcv.FC} = \text{CD} \wedge \text{local\_addr}(\text{rcv.DA})$: process_CD(rcv); V(DA) $\leftarrow$ rcv.DA; V(SA) $\leftarrow$ rcv.SA;
11f	$\text{rcv.FC} = \text{ED} \wedge \text{local\_addr}(\text{rcv.DA})$: process_ED(rcv); V(DA) $\leftarrow$ rcv.DA; V(SA) $\leftarrow$ rcv.SA;
11g	$\text{rcv.FC} \leftarrow \text{DT} \wedge ((\text{rcv.FC.DA} \wedge \text{local\_addr}(\text{rcv.DA})) \vee (!\text{rcv.FC.DA} \wedge \text{rcv.FC.SA} \wedge \text{subscriber}(\text{rcv.SA})) \vee (!\text{rcv.FC.DA} \wedge !\text{rcv.FC.SA} \wedge \text{subscriber}(V(\text{DA}))) \vee (!\text{rcv.FC.DA} \wedge !\text{rcv.FC.SA} \wedge \text{local\_addr}(V(\text{SA}))))$: process_DT(rcv); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;
11h	$\text{rcv.FC} = \text{TD}$: rcv.time_of_receipt $\leftarrow$ C(NT); V(LN) $\leftarrow$ rcv.SA.N; /* track LAS node-id */ process_TD(rcv); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;
11j	$\text{rcv.FC} \in \{\text{CL}, \text{TL}\}$: V(LN) $\leftarrow$ rcv.SA.N; /* track LAS node-id */ dIm_event(CL_or_TL_received); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;
11k	$\text{rcv.FC} = \text{WK} \wedge \text{rcv.DA.N} = V(\text{TN})$: power_up_local_circuitry; dIm_event(WK_received); V(DA) $\leftarrow$ null; V(SA) $\leftarrow$ null;

B.1.3 The DLE-state FSM

DLE-state FSM



responding FSM

state	name	transitions	entry actions
1	offline	12	
2	learnLinkVars	21, 23	
3	waitForPN	31, 33, 34, 35	if $V(TN) \geq x'F8'$ then $V(TN)_{1-0} \leftarrow (random \wedge x'03')$; $C(RR) \leftarrow V(MRD)+1$;
4	watchingTN	41, 43	
5	waitForNodeActivation	51, 53, 56	$V(RID) \leftarrow random$; send PR(V(RID));
6	online	61, 65	

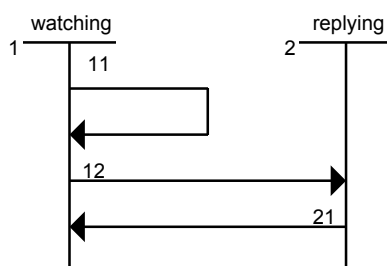
transition condition : [actions]

12	DLM_action("go online") :
n1	DLM_action("go offline") ($2 \leq n \leq 6$) :
23	rcv.FC=PN : set essential PhL and DLL variables from PN-parameters;
33a	rcv.FC=PT $\wedge$ rcv.DA.N=V(TN) $\wedge$ V(TN) $\geq x'F8'$:
33b	rcv.FC=PN $\wedge$ rcv.DA.N=V(TN) $\wedge$!(is_usable(V(MID)) $\wedge$ is_usable(V(MRD))) : $V(RID) \leftarrow random$; send PR(V(RID));
34	rcv.FC=PT $\wedge$ rcv.DA.N=V(TN) $\wedge$ V(TN) $< x'F8'$:
35	rcv.FC=PN $\wedge$ rcv.DA.N=V(TN) $\wedge$ is_usable(V(MID)) $\wedge$ is_usable(V(MRD)) :
43a	rcv.FC=any : $V(TN) \leftarrow x'F8'$;

43b $C(RR)=0$
 :
 53a $rcv.FC \in \{PN, PT\} \wedge rcv.DA.N=V(TN)$
 :
 53b $rcv.FC=DT \wedge !rcv.DA.is\_group \wedge rcv.DA.N=V(TN) \wedge rcv.DA.S=00$
 $\wedge (rcv.data.SPDU\_type \neq \text{"node activation"}$
 $\vee rcv.data.node\_activation\_SPDU.N(RID) \neq V(RID))$
 :
 56b $rcv.FC=DT \wedge rcv.DA.N=V(TN) \wedge rcv.DA.S=00$
 $\wedge rcv.data.SPDU\_type = \text{"node activation"}$
 $\wedge rcv.data.node\_activation\_SPDU.N(RID)=V(RID))$
 : set remaining basic DLL variables from SPDU;
 63 $rcv.FC=PN \wedge rcv.DA.N=V(TN) \wedge !(is\_usable(V(MID)) \wedge is\_usable(V(MRD)))$
 :
 65 $rcv.FC=PN \wedge rcv.DA.N=V(TN) \wedge is\_usable(V(MID)) \wedge is\_usable(V(MRD))$
 :
 n6a $DLE\_is\_LM \wedge DLE\_knows\_DLL\_variables \wedge FSM\_transition(link\_master\_FSM, 12)$
 (2≤n≤5) :

B.1.4 The responder FSM

responding FSM



responding FSM

state	name	transitions	entry actions
1	watching	11, 12	
2	replying	21	$C(FD) \Leftarrow V(MRD)$

transition condition : [actions]

```

11a      rcv.FC ∈ {CA,CD,ED,RQ} ∨ rcv.DA.is_group ∨ !local_addr(rcv.DA)
          ∨ rcv_data_length≠0
          :
11b      rcv.FC ∈ {CA,CD,ED} ∧ !rcv.DA.is_group ∧ local_addr(rcv.DA)
          ∧ rcv_data_length=0
          ∧ !( local_object(rcv.DA).typ ∈ {peer_DLCEP,publisher_DLCEP}
              ∧ (!rcv.FC.SA ∨ rcv.SA=local_object(rcv.DA).remote_DLCEP_addr) )
          :  send_SR(V(TN));

12a      rcv.FC ∈ {CA,CD,ED} ∧ !rcv.DA.is_group ∧ local_addr(rcv.DA)
          ∧ ( local_object(rcv.DA).typ ∈ {peer_DLCEP,publisher_DLCEP}
              ∧ (!rcv.FC.SA ∨ rcv.SA=local_object(rcv.DA).remote_DLCEP_addr) )
          :  form_DT_reply per rcv.DA;

12b      rcv.FC=RR ∧ local_addr(rcv.DA)
          :  form_RR_reply;

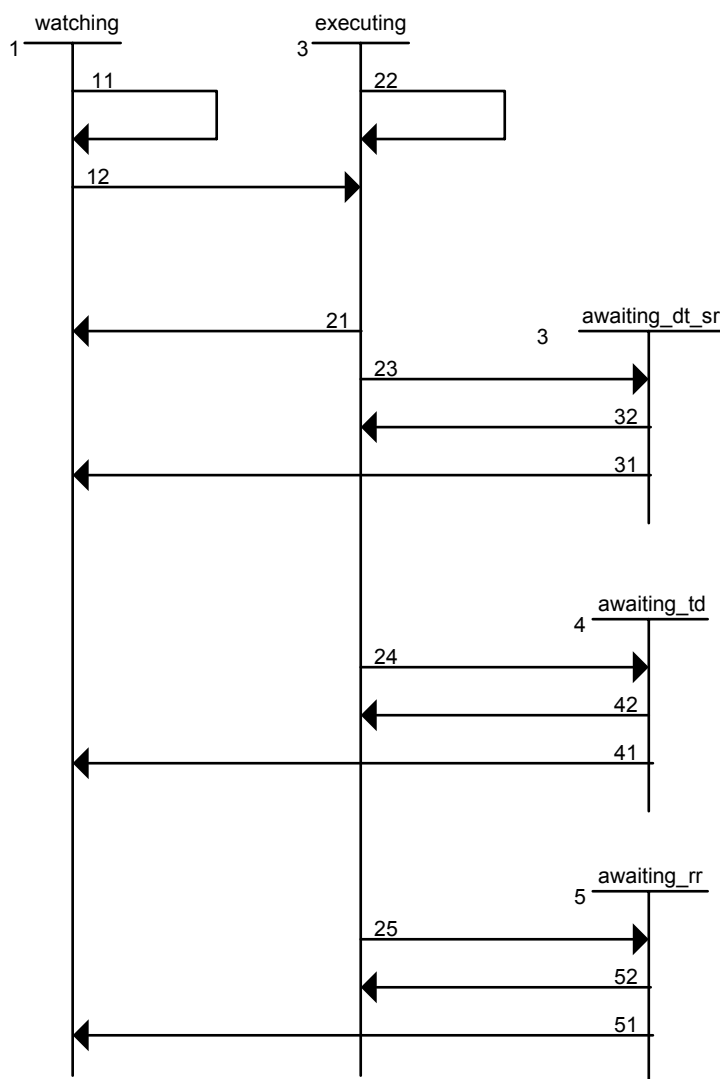
21a      reply_formed
          :  send_item(next);

21b      C(FD)=0
          :

form_DT_reply per rcv.DA
{
  next ← DT_reply_to(rcv.DA);
  if    QW(SS).is_empty ∧ !Q(LR).is_empty
  then  move first element of Q(LR) to end of appropriate priority of QW(SS);
  if    !QW(SS).is_empty
  then  next.FC.DD ← priority(QW(SS).head);
}

form_RR_reply
{
  next ← RR_reply_to(rcv.DA);
  add value of C(NT) at the time of sending;
}

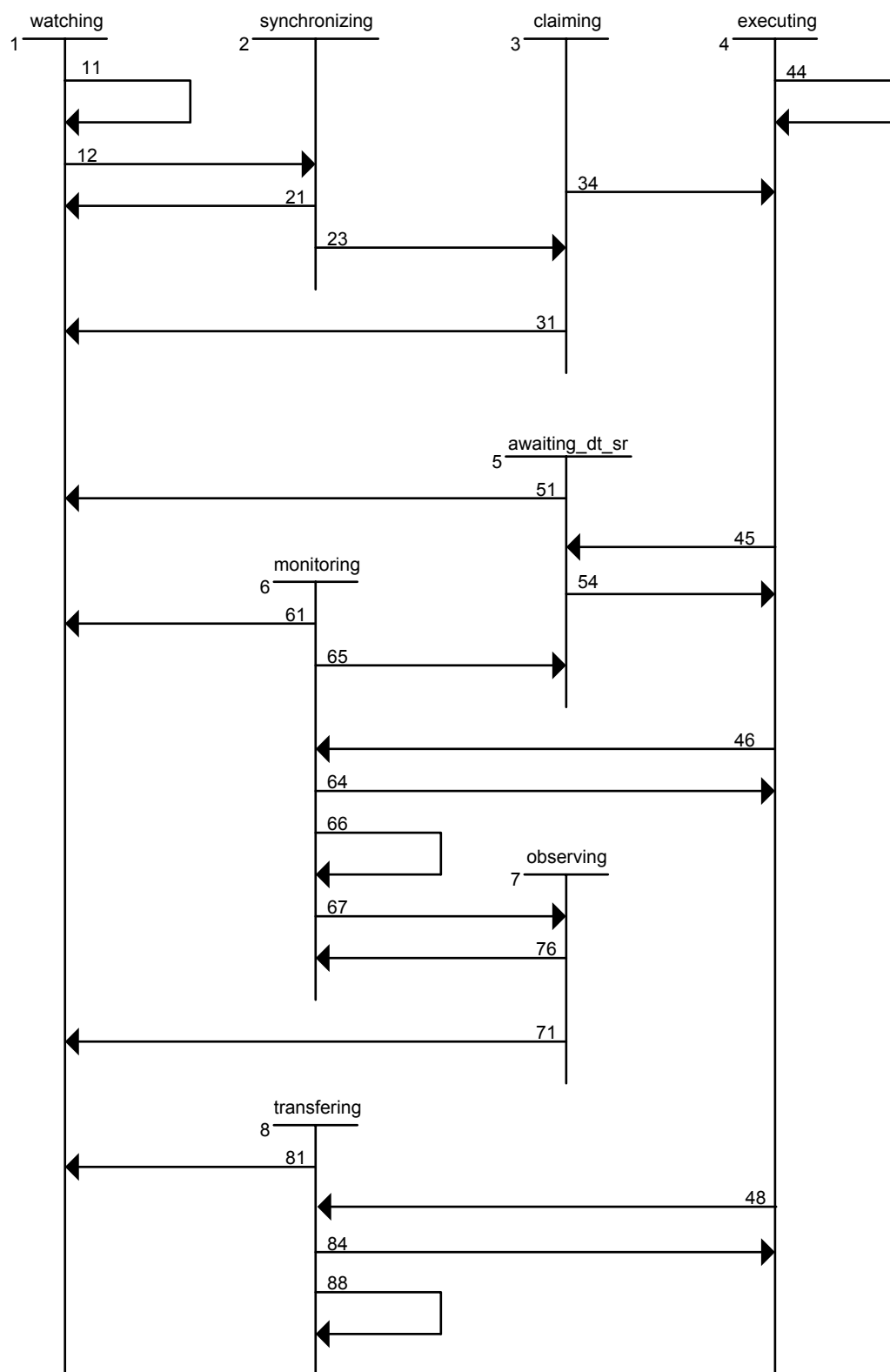
```

B.1.5 The token-holder FSM**token holding FSM**token holding FSM

state	name	transitions	entry actions
1	watching	12	
2	executing	21, 22, 23, 24, 25	select_next
3	awaiting_dt_sr	31, 32	$C(RR) \Leftarrow V(MRD)+1$
4	awaiting_td	41, 42	$C(RR) \Leftarrow V(MRD)+1$
5	awaiting_rr	51, 52	$C(RR) \Leftarrow V(MRD)+1$

transition condition : [actions]

12a	rcv.FC=PT $\wedge$ rcv.DA.N = TN : V(TT) $\Leftarrow$ rcv.FC; C(RD) $\Leftarrow$ rcv.PT_duration; next.FC.F $\Leftarrow$ NOT-FINAL;
12b	rcv.FC=ES $\wedge$!rcv.DA.is_group $\wedge$ local_addr(rcv.DA) $\wedge$ local_object(rcv.DA).typ = DLSEP : V(TT) $\Leftarrow$ rcv.FC; C(RD) $\Leftarrow$ rcv.ES_duration; next.FC.F $\Leftarrow$ NOT-FINAL;
21	next.FC $\in$ {RI,RT} $\vee$ (next.FC $\in$ {EC,DC,RC,DT,IDLE} $\wedge$ next.FC.F) : send_item(next);
22	next.FC $\in$ {EC,DC,RC,DT,IDLE} $\wedge$!next.FC.F : send_item(next);
23	next.FC $\in$ {CA,CD,ED} : send_item(next); V(DA) $\Leftarrow$ next.DA; V(SA) $\Leftarrow$ next.SA;
24	next.FC=CT : send_item(next);
25	next.FC=RQ : send_item(next);
31	!(rcv.FC=SR $\vee$ (rcv.FC $\in$ {DT} $\wedge$!next.FC.F $\wedge$ rcv.FC.DA $\wedge$ rcv.DA=V(SA))) :
32a	rcv.FC=SR $\vee$ (rcv.FC $\in$ {DT} $\wedge$!next.FC.F $\wedge$ rcv.FC.DA $\wedge$ rcv.DA=V(SA)) :
32b	C(RR)=0 :
41	rcv.FC $\neq$ TD : abort_token=TRUE;
42a	rcv.FC=TD :
42b	C(RR)=0 :
51	rcv.FC $\neq$ RR $\vee$ rcv.DA $\neq$ V(SA) : abort_token=TRUE;
52a	rcv.FC=RR $\wedge$ rcv.DA=V(SA) :
52b	C(RR)=0 :

B.1.6 The link-master FSM**link master FSM**

LM FSM

state	name	transitions	entry actions
1	watching	11, 12	$C(CT) \leftarrow V(TN)$
2	synchronizing	21, 23	$C(CT) \leftarrow V(TN)$
3	claiming	31, 34	$C(CT) \leftarrow \text{random} \wedge x'03'$
4	executing	44, 45, 46, 48	select next per schedule
5	awaiting_dt_sr	51, 54	$C(CT) \leftarrow V(MRD)+1$
6	monitoring	61, 64, 65, 66, 67	$C(CT) \leftarrow P(TRD)$
7	observing	71, 76	$C(CT) \leftarrow P(TRD)$
8	transferring	81, 84, 88	$C(CT) \leftarrow x'0F'$

transition condition : [actions]

11	falling_edge(RxA) :
12	$C(CT)=0 \wedge !FSM\_state(link\_master\_FSM, 1)$: send_CL(V(TN));
21	falling_edge(RxA) :
23	$C(CT)=0$: send_CL(V(TN));
31	falling_edge(RxA) :
34	$C(CT)=0$:
44	$next.FC \in \{EC, DC, RC, DT, TD, WK, IDLE\}$: send_item(next);
45	$next.FC \in \{CA, CD, ED\}$: send_item(next); V(TT) $\leftarrow$ next.FC; V(DA) $\leftarrow$ next.DA;
46a	$next.FC=PT$: send_item(next); V(TT) $\leftarrow$ next.FC; V(DTA) $\leftarrow$ next.DA;
46b	$next.FC=ES \wedge next.ES\_duration \neq 0$: send_item(next); V(TT) $\leftarrow$ next.FC; V(DTA) $\leftarrow$ next.DA;
46c	$next.FC=ES \wedge next.ES\_duration = 0$: next.ES_duration $\leftarrow$ C(UC); /* within the LAS' schedule an ES_duration */ send_item(next); /* of 0 requests a pseudo-circulated token */ V(TT) $\leftarrow$ next.FC; V(DTA) $\leftarrow$ next.DA;
48	$next.FC=TL$: send_item(next); V(TT) $\leftarrow$ next.FC; V(DTA) $\leftarrow$ next.DA;
51	falling_edge(RxA) ^ !(rcv.FC=SR $\vee$ (rcv.FC=DT $\wedge$!rcv.FC.F $\wedge$ (!rcv.FC.SA $\vee$ rcv.SA=V(DA)))) :
54	$rcv.FC=SR \vee (rcv.FC=DT \wedge !rcv.FC.F \wedge (!rcv.FC.SA \vee rcv.SA=V(DA)))$:

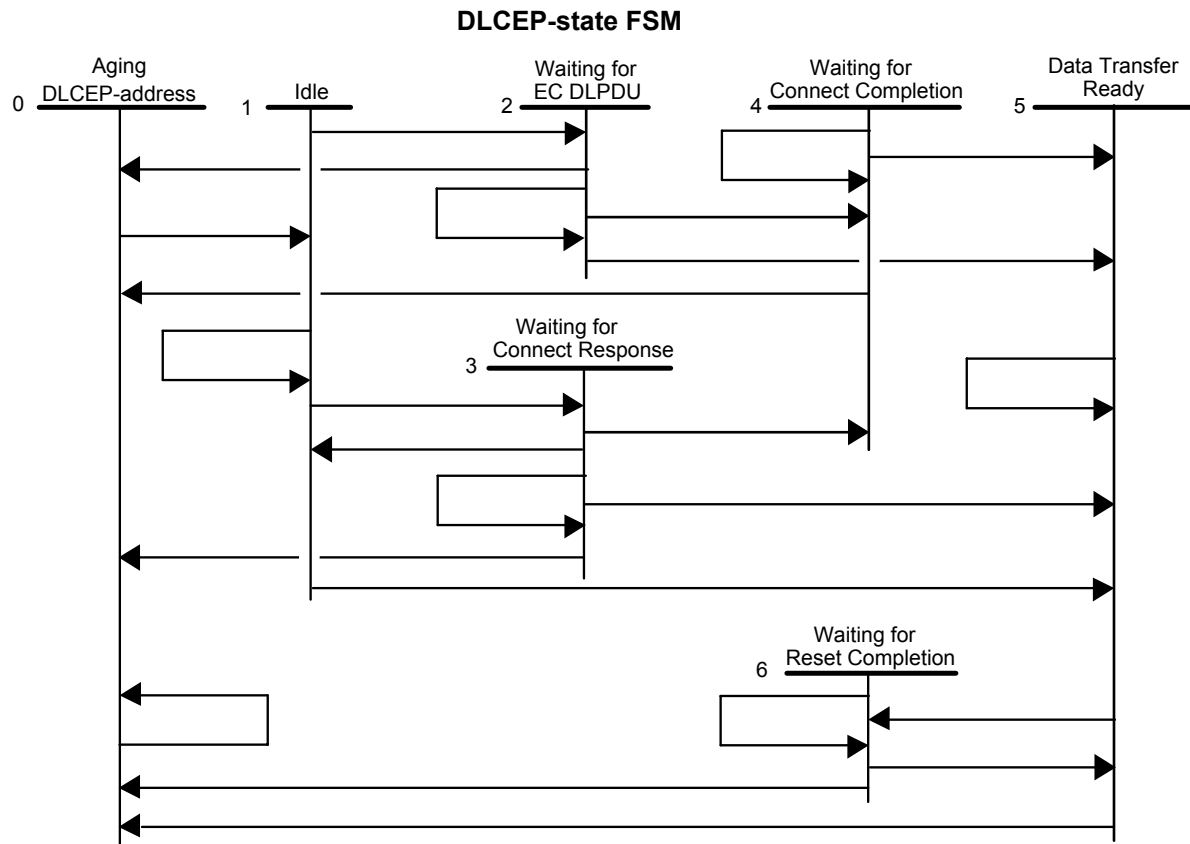

```

61      rcv.FC ∈ {WK,TD}
        :
64      rcv.FC ∈ {EC,DC,RC,DT} ∧ rcv.FC.F
        :   V(DA) ⇐ rcv.DA;
65      rcv.FC ∈ {CA,CD,ED} ∧ rcv.FC.F
        :   V(DA) ⇐ rcv.DA;
66a     rcv.FC ∈ {EC,DC,RC,DT} ∧ !rcv.FC.F
        :
66b     rcv.FC ∈ {IDLE}
        :
66c     rcv.FC=CT
        :   send_TD(V(NTS),V(NTQ),V(NTO),C(NT));
66d     C(CT)=0
        :   send_RT(V(TN));
67      rcv.FC ∈ {CA,CD,ED,RQ} ∧ !rcv.FC.F
        :   V(DA) ⇐ rcv.DA;
71      falling_edge(RxA)
        :   ∧ !(rcv.FC ∈ {SR,RR}
        :       ∨ (rcv.FC=DT ∧ !rcv.FC.F ∧ (!rcv.FC.SA ∨ rcv.SA=V(DA))))
        :
76a     rcv.FC ∈ {SR,RR} ∨ (rcv.FC=DT ∧ !rcv.FC.F ∧ (!rcv.FC.SA ∨ rcv.SA=V(DA)))
        :
76b     C(CT)=0
        :
81      falling_edge(RxA) ∧ FCS_OK      /* This is equivalent to rcv.FC in all */
        :
84      C(CT)=0
        :
88      falling_edge(RxA) ∧ !FCS_OK
        :

```

B.2 FSMs for DLCs

B.2.1 The DLCEP-state FSM



DLCEP-state FSM

state	name	transitions
0	agingDLCEPaddress	00, 01
1	idle	11, 12, 13, 15
2	waitingForEC_DLPDU	20, 22, 24, 25
3	waitingForConnectResponse	30, 31, 33, 34, 35
4	waitingForConnectCompletion	40, 44, 45
5	dataTransferReady	50, 55, 56
6	waitingForResetCompletion	60, 65, 66

NOTE Within states 2 to 6, non-hierarchical variable references are references to variables of the active DLCEP.

transition condition : [actions]

01	timeout : delete_object(DLCEP);
11a	DL-CONNECT.request ^ (calling_id=DLCEP_id ^ class≠PUBLISHER) ∨ !accepted_by_DLE : DL-DISCONNECT.indication;
11b	rcv.FC=EC ^ !accepted_by_DLE ^ rcv.DA_exists ^ !rcv.DA.is_group : send_DC(rcv.SA, rcv.DA, !reply_requested);
11c	rcv.FC=EC ^ !accepted_by_DLE ^ (!rcv.DA_exists ∨ rcv.DA.is_group) :

- 11d $\text{rcv.FC} = \text{EC} \wedge \text{rcv.DA\_exists} \wedge (\text{rcv.DA.is\_DLSAP\_addr} \vee \text{rcv.DA.is\_group})$
 $\wedge \exists (\text{DLCEP}, \text{DLCEP.remote\_DLCEP\_addr} = \text{rcv.SA})$
 $\wedge \text{DLCEP.remote\_DLSAP\_addr} \neq \text{rcv.SA2}$
 : $\text{send\_DC}(\text{rcv.SA}, \text{rcv.DA}, \text{!reply\_requested});$
- 11e $\text{rcv.FC} = \text{EC} \wedge \text{rcv.DA\_exists} \wedge (\text{rcv.DA.is\_DLSAP\_addr} \vee \text{rcv.DA.is\_group})$
 $\wedge \exists (\text{DLCEP}, \text{DLCEP.remote\_DLCEP\_addr} = \text{rcv.SA})$
 $\wedge \text{DLCEP.remote\_DLSAP\_addr} = \text{rcv.SA2}$
 :
- 12 $\text{DL-CONNECT.request} \wedge \text{accepted\_by\_DLE} \wedge \text{class} \neq \text{PUBLISHER}$
 : $\text{create\_object}(\text{DLCEP});$
 $\text{DLCEP.DLCEP\_addr} \leftarrow \text{requested\_or\_new\_DLCEP}(\text{req});$
 $\text{DLCEP.DLSAP\_addr} \leftarrow \text{req.calling\_DLSAP\_addr};$
 $\text{DLCEP.called\_addr} \leftarrow \text{req.called\_addr};$
 $\text{DLCEP.required\_interactions} \leftarrow \text{cnf};$
 $\text{activate\_addr\_recognition}(\text{DLCEP.DLCEP\_addr});$
 if $\text{DLCEP.called\_addr} \neq \text{UNKNOWN}$
 then $\text{send\_EC}(\text{DLCEP.called\_addr}, \text{DLCEP.DLCEP\_addr},$
 $\text{DLCEP.DLSAP\_addr}, \text{reply\_requested});$
 $\text{start\_timer}(\text{DLCEP.send\_params.max\_confirm\_delay\_for\_EC}$
 $/ (\text{V}(\text{MRC}) + 1));$
- 13 $\text{rcv.FC} = \text{EC} \wedge \text{rcv.DA\_exists} \wedge (\text{rcv.DA.is\_DLSAP\_addr} \vee \text{rcv.DA.is\_group})$
 $\wedge \exists (\text{DLCEP}, \text{DLCEP.remote\_DLCEP\_addr} = \text{rcv.SA})$
 : $\text{create\_object}(\text{DLCEP});$
 $\text{DLCEP.remote\_DLCEP\_addr} \leftarrow \text{rcv.SA};$
 $\text{DLCEP.remote\_DLSAP\_addr} \leftarrow \text{rcv.SA2};$
 $\text{DLCEP.required\_interactions} \leftarrow \text{rsp\_ind};$
 $\text{DL-CONNECT.indication}(\text{DLCEP});$
 $\text{start\_timer}(\text{DLCEP}, \text{rcv.EC.p.send\_params.max\_confirm\_delay\_for\_EC});$
 --
 -- The following assignment is needed temporarily, until a DL-CONNECT.response is
 -- received, for use with any requested or internally-generated DC DLPDU.
 -- It will not be used if rcv.DA was a group DL-address.
 --
 $\text{DLCEP.DLCEP\_addr} \leftarrow \text{rcv.DA};$
- 15 $\text{DL-CONNECT.request} \wedge \text{accepted\_by\_DLE} \wedge \text{class} = \text{PUBLISHER}$
 : **if** $\text{!req.DLCEP.calling\_id.is\_DLCEP\_id}$
 then $\text{create\_object}(\text{DLCEP});$
 $\text{DLCEP.DLCEP\_addr} \leftarrow \text{requested\_or\_new\_DLCEP}(\text{req});$
 $\text{activate\_addr\_recognition}(\text{DLCEP.DLCEP\_addr});$
 $\text{DLCEP.DLSAP\_addr} \leftarrow \text{req.calling\_DLSAP\_addr};$
 $\text{DLCEP.called\_addr} \leftarrow \text{req.called\_addr};$
 if $\text{req.called\_addr} = \text{UNKNOWN}$
 then $\text{send\_EC}(\text{—}, \text{DLCEP.DLCEP\_addr}, \text{DLCEP.calling\_addr},$
 $\text{!reply\_requested});$
 else $\text{send\_EC}(\text{DLCEP.called\_addr}, \text{DLCEP.DLCEP\_addr},$
 $\text{DLCEP.DLSAP\_addr}, \text{!reply\_requested});$
 $\text{DL-CONNECT.confirm};$
- 20b $\text{timeout} \wedge (\text{retry\_count} = \text{max\_retry\_count}) \wedge \text{!called\_addr.is\_group}$
 : {see generic *n0b* transition}
- 20c $\text{timeout} \wedge (\text{retry\_count} = \text{max\_retry\_count}) \wedge \text{called\_addr.is\_group}$
 : {see generic *n0c* transition}
- 20e $\text{rcv.FC} = \text{EC} \wedge \text{rcv.DA\_exists} \wedge \text{rcv.DA} = \text{DLCEP\_addr} \wedge \text{!accepted\_by\_DLE}$
 $\wedge \text{!called\_addr.is\_group}$
 : $\text{DL-DISCONNECT.indication}$
 $\text{send\_DC}(\text{rcv.SA}, \text{DLCEP\_addr}, \text{!reply\_requested});$
 $\text{start\_timer}();$
- 22a $\text{timeout} \wedge (\text{retry\_count} < \text{max\_retry\_count})$
 : $\text{send\_EC}(\text{called\_addr}, \text{DLCEP\_addr}, \text{DLSAP\_addr}, \text{reply\_requested});$
 $\text{start\_timer}(\text{send\_params.max\_confirm\_delay\_for\_EC} / (\text{V}(\text{MRC}) + 1));$
- 22b $\text{rcv.FC} = \text{EC} \wedge \text{!accepted\_by\_DLE} \wedge (\text{!rcv.DA\_exists} \vee \text{called\_addr.is\_group})$
 : $\text{send\_DC}(\text{rcv.SA}, \text{DLCEP\_addr}, \text{!reply\_requested});$


```

35      DL-CONNECT.response  $\wedge$  !responding_id.is_DLCEP_id  $\wedge$  accepted_by_DLE
       $\wedge$  class $\neq$ PEER
      : DLCEP_addr  $\leftarrow$  new_or_requested_DLCEP_addr(rsp);
        DLSAP_addr  $\leftarrow$  rsp.calling_DLSAP_addr;
        stop_timer;
        activate_addr_recognition(DLCEP_addr);
        if class=PUBLISHER
        then send_EC(remote_DLCEP_addr, DLCEP_addr, DLSAP_addr,
                     !reply_requested);
        else reset_seq_numbers_to_next_received  $\leftarrow$  TRUE;
             DL-CONNECTION-ESTABLISHED.indication;

40b      timeout
      :{see generic n0b transition}

44b      rcv.FC=EC  $\wedge$  rcv.DA_exists  $\wedge$  rcv.EC_p.reply_requested
       $\wedge$  ( rcv.DA=DLCEP_addr
       $\vee$  ( rcv.DA.is_DLSAP_addr  $\vee$  rcv.DA.is_group)
       $\wedge$   $\exists$ (DLCEP, DLCEP.remote_DLCEP_addr=rcv.SA)
       $\wedge$  DLCEP.remote_DLSAP_addr=rcv.SA2 ) )
      : send_EC(rcv.SA, DLCEP_addr, DLSAP_addr, !reply_requested);
        start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1) );

44c      timeout  $\wedge$  (retry_count < max_retry_count)
      : send_EC(called_addr, DLCEP_addr, DLSAP_addr, reply_requested);
        start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1) );

44d      rcv.FC=EC  $\wedge$  !rcv.DA_exists  $\wedge$  rcv.SA_exists  $\wedge$  rcv.SA=remote_DLCEP_addr
       $\wedge$  (rcv.publisher-DLCEP-address-reuse-discriminator
      =publisher-DLCEP-address reuse-discriminator)
      :

45a      rcv.FC=EC  $\wedge$  rcv.DA_exists  $\wedge$  rcv.DA=DLCEP_addr  $\wedge$  !rcv.EC_p.reply_requested
      : stop_timer;
        if SA_is_required
        then send_DT(rcv.SA, DLCEP_addr);
        else send_DT(rcv.SA, —);
        if required_interactions=cnf
        then DL-CONNECT.confirm;
        else DL-CONNECTION-ESTABLISHED.indication;

45b      rcv.FC=DT
      : stop_timer;
        if required_interactions=cnf
        then DL-CONNECT.confirm;
        else DL-CONNECTION-ESTABLISHED.indication;

50c      rcv.FC=EC  $\wedge$  !rcv.DA_exists  $\wedge$  rcv.SA_exists  $\wedge$  rcv.SA=remote_DLCEP_addr
       $\wedge$  (rcv.publisher-DLCEP-address-reuse-discriminator
       $\neq$ publisher-DLCEP-address reuse-discriminator)
      : DL-DISCONNECT.indication;
        start_timer();

55b      rcv.FC=EC  $\wedge$  rcv.DA_exists  $\wedge$  rcv.DA=DLCEP_addr  $\wedge$  class=PUBLISHER
       $\wedge$  accepted_by_DLE  $\wedge$  rcv.EC_p.reply_requested
      : send_EC(—, DLCEP_addr, DLSAP_addr, !reply_requested);

55c      rcv.FC=EC  $\wedge$  rcv.DA_exists  $\wedge$  rcv.DA=DLCEP_addr
       $\wedge$  class=PUBLISHER  $\wedge$  !(accepted_by_DLE  $\wedge$  rcv.EC_p.reply_requested)
      : send_DC(rcv.SA, rcv.DA, !reply_requested);

55d      rcv.FC=EC  $\wedge$  rcv.DA_exists  $\wedge$  rcv.DA=DLCEP_addr
       $\wedge$  class $\neq$ PUBLISHER
      :

55e      rcv.FC=EC  $\wedge$  !rcv.DA_exists  $\wedge$  rcv.SA_exists  $\wedge$  rcv.SA=remote_DLCEP_addr
       $\wedge$  (rcv.publisher-DLCEP-address-reuse-discriminator
      =publisher-DLCEP-address reuse-discriminator)
      :

55f      rcv.FC  $\in$  {CA,CD,ED}  $\wedge$  rcv.DA=DLCEP_addr
       $\wedge$  ( !rcv.SA_exists  $\wedge$  !SA_is_required )  $\vee$  rcv.SA=remote_DLCEP_addr )
      : ...

```

55g $rcv.FC=DT \wedge rcv.DA\_exists \wedge rcv.DA=DLCEP\_addr$
 $\wedge (!rcv.SA\_exists \wedge !SA\_is\_required) \vee rcv.SA=remote\_DLCEP\_addr)$
 : ...

55h $rcv.FC=DT \wedge !rcv.DA\_exists$
 $\wedge ((rcv.SA\_exists \wedge rcv.SA=remote\_DLCEP\_addr)$
 $\vee (rcv.SA\_was\_implied \wedge !SA\_is\_required$
 $\wedge rcv.implied\_SA=remote\_DLCEP\_addr))$
 : ...

55j $DL-DATA.request \vee DL-SUBSCRIBER-QUERY.request$
 : ...

55k $timeout \wedge (retry\_count < max\_retry\_count)$
 : ...

55m $rcv.FC=RC \wedge rcv.DA\_exists \wedge rcv.DA=DLCEP\_addr \wedge class=PUBLISHER$
 : send_DC(rcv.SA, rcv.DA, !reply_requested);

56a $rcv.FC=RC \wedge rcv.DA\_exists \wedge rcv.DA=DLCEP\_addr \wedge class=PEER$
 $\wedge rcv.SA=remote\_DLCEP\_addr$
 : DL-RESET.indication;
 flush_internal_queues;
 required_interactions $\leftarrow$ rsp_ind;
 pended_DLPDUs $\leftarrow$ RC;
 expected_DLPDUs $\leftarrow$ DT;
 rcv_params.NDS $\leftarrow$ rcv.RC_p.NDS;
 rcv_params.NDR $\leftarrow$ rcv.RC_p.NDR;
 start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1));

56b $rcv.FC=RC \wedge !rcv.DA\_exists \wedge rcv.SA=DLCEP\_addr \wedge class=SUBSCRIBER$
 : DL-RESET.indication;
 flush_internal_queues;
 required_interactions $\leftarrow$ rsp_ind;
 pended_DLPDUs $\leftarrow$ none;
 expected_DLPDUs $\leftarrow$ none;

56c DL-RESET.request
 : flush_internal_queues;
 required_interactions $\leftarrow$ cnf;
 pended_DLPDUs $\leftarrow$ none;
 if class=PEER
 then expected_DLPDUs $\leftarrow$ RC;
 send_RC(remote_DLCEP_addr, DLCEP_addr, reply_requested,
 V_C(M), V_C(L)+1);
 start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1));
 elseif class=PUBLISHER
 then expected_DLPDUs $\leftarrow$ none;
 send_RC(—, DLCEP_addr, !reply_requested, V_C(N));
 else expected_DLPDUs $\leftarrow$ none;

56d internal reset condition
 : DL-RESET.indication;
 flush_internal_queues;
 required_interactions $\leftarrow$ rsp_ind;
 pended_DLPDUs $\leftarrow$ none;
 if class=PEER
 then expected_DLPDUs $\leftarrow$ RC;
 send_RC(remote_DLCEP_addr, DLCEP_addr, reply_requested,
 V_C(M), V_C(L)+1);
 start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1));
 elseif class=PUBLISHER
 then expected_DLPDUs $\leftarrow$ none;
 send_RC(—, DLCEP_addr, !reply_requested, V_C(N))
 else expected_DLPDUs $\leftarrow$ none;

60b $timeout \wedge (required\_interactions=rsp$
 $\vee (expected\_DLPDUs \neq none \wedge retry\_count=max\_retry\_count))$
 : {see generic n0b transition}

```

65      required_interactions≠rsp_ind ∧ expected_DLPDUs=none
      : reset_seq_numbers_to_next_received ⇐ (class=SUBSCRIBER);
        flush_internal_queues;
        if required_interactions=cnf
        then DL-RESET.confirm;
        else DL-RESET-COMPLETED.indication;

66b     rcv.FC=EC ∧ rcv.DA_exists ∧ rcv.DA=DLCEP_addr ∧ class=PUBLISHER
        ∧ rcv.EC_p.reply_requested
      : expected_DLPDUs ⇐ none;
        send_EC(rcv.SA, DLCEP_addr, DLSAP_addr, !reply_requested);

66c     rcv.FC=EC ∧ rcv.DA_exists ∧ rcv.DA=DLCEP_addr
        ∧ (class≠PUBLISHER ∨ !rcv.EC_p.reply_requested)
      :

66d     rcv.FC=EC ∧ !rcv.DA_exists ∧ rcv.SA_exists ∧ rcv.SA=remote_DLCEP_addr
      :

66e     rcv.FC=RC ∧ rcv.DA_exists ∧ rcv.DA=DLCEP_addr ∧ class=PEER
        ∧ rcv.SA=remote_DLCEP_addr ∧ required_interactions≠rsp_ind
        ∧ rcv.RC_p.reply_requested
      : if required_interactions=cnf
        then expected_DLPDUs ⇐ RC;
        else expected_DLPDUs ⇐ DT;
        send_RC(rcv.SA, DLCEP_addr, !reply_requested, VC(M), VC(L)+1);
        start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1) );

66f     rcv.FC=RC ∧ rcv.DA_exists ∧ rcv.DA=DLCEP_addr ∧ class=PEER
        ∧ rcv.SA=remote_DLCEP_addr ∧ required_interactions≠rsp_ind
        ∧ expected_DLPDUs=RC ∧ !rcv.RC_p.reply_requested
      : expected_DLPDUs ⇐ none;
        if SA_is_required
        then send_DT(rcv.SA, DLCEP_addr);
        else send_DT(rcv.SA, —);

66g     rcv.FC=RC ∧ rcv.DA_exists ∧ rcv.DA=DLCEP_addr ∧ class=PEER
        ∧ rcv.SA=remote_DLCEP_addr ∧ required_interactions≠rsp_ind
      : if rcv.RC_p.reply_requested
        then pending_DLPDUs ⇐ RC;
        else pending_DLPDUs ⇐ DT;

66h     rcv.FC=RC ∧ !rcv.DA_exists ∧ rcv.SA=DLCEP_addr ∧ class=SUBSCRIBER
      :

66j     rcv.FC=DT ∧ rcv.DA_exists ∧ rcv.DA=DLCEP_addr ∧ class=PEER
        ∧ ( !rcv.SA_exists ∧ !SA_is_required ) ∨ rcv.SA=remote_DLCEP_addr
        ∧ required_interactions≠rsp_ind ∧ expected_DLPDUs=DT;
        ∧ (rcv.SD_p.NDS ≥ rcv_params.NDS)
      : expected_DLPDUs ⇐ none;

66k     DL-RESET.response ∧ required_interactions=rsp_ind
      : required_interactions=ind;
        if pending_DLPDUs=RC
        then expected_DLPDUs ⇐ DT;
          send_RC(rcv.SA, DLCEP_addr, !reply_requested, VC(M), VC(L)+1);
          start_timer(send_params.max_confirm_delay_for_EC / (V(MRC)+1) );
        elseif pending_DLPDUs=DT
        then expected_DLPDUs ⇐ none;
          if SA_is_required
          then send_DT(rcv.SA, DLCEP_addr);
          else send_DT(rcv.SA, —);

66m     timeout ∧ (expected_DLPDUs≠none) ∧ (retry_count < max_retry_count)
        ∧ required_interactions≠rsp_ind
      : if expected_DLPDUs=RC
        then send_RC(remote_DLCEP_addr, DLCEP_addr, !reply_requested,
                     VC(M), VC(L)+1);
        else send_RC(remote_DLCEP_addr, DLCEP_addr, reply_requested,
                     VC(M), VC(L)+1);

```

n0a
($2 \leq n \leq 6$) DL-DISCONNECT.request
: **if** class=PEER
 then send_DC(remote_DLCEP_addr, DLCEP_addr, reply_requested);
 elseif class=PUBLISHER
 then send_DC(—, DLCEP_addr, !reply_requested);
 start_timer();

n0b
($2 \leq n \leq 6$) internal disconnect condition when a DC DLPDU is appropriate
: **if** class=PEER
 then send_DC(remote_DLCEP_addr, DLCEP_addr, reply_requested);
 elseif class=PUBLISHER
 then send_DC(—, DLCEP_addr, !reply_requested);
 DL-DISCONNECT.indication;
 start_timer();

n0c
($2 \leq n \leq 3$) internal disconnect condition when a DC DLPDU is not appropriate
: DL-DISCONNECT.indication;
 start_timer();

n0d
($2 \leq n \leq 6$) rcv.FC=DC $\wedge$ rcv.DA=DLCEP_addr $\wedge$ rcv.SA=remote_DLCEP_addr
: **if** rcv.DC.p.reply_requested $\wedge$ class=PEER
 then send_DC(remote_DLCEP_addr, DLCEP_addr, !reply_requested);
 DL-DISCONNECT.indication;
 start_timer();

nnn
($3 \leq n \leq 6$) rcv.FC $\in$ {EC,RC,CA,CD,ED,DT} $\wedge$ rcv.DA_exists $\wedge$ rcv.DA=DLCEP_addr
(or **n**=0) $\wedge$ rcv.SA_exists $\wedge$ rcv.SA $\neq$ remote_DLCEP_addr $\wedge$ class=PEER
: send_DC(rcv.SA, DLCEP_addr, !reply_requested);

B.3 FSMs for scheduling

This is a future work item.

B.4 FSMs for bridges

B.4.1 Overview

This is a future work item.

B.4.2 The port-state FSM

See 11.4.6 and clause 6 of ISO/IEC 10038:1993.

B.4.3 The bridge-state FSM

See 11.4.6 and clause 6 of ISO/IEC 10038:1993.

B.4.4 The DLPDU forwarding FSM

This is a future work item.

B.4.5 The time distribution FSM

This is a future work item.

Annex C (informative)

Type 1: DLPDU and DL-addressing short-form summaries

NOTE This annex provides a number of compact summaries of the various types of DLPDUs and DL-addresses.

C.1 Fields used in short-form summaries

C.1.1 Subfields of the Frame Control (FC) octet

PP DLPDU and transaction priority (01=URGENT, 10=NORMAL, 11=TIME-AVAILABLE)

L length of address fields in DLPDUs which have both a long and a short form
(0=SHORT — two-octet node.selector with an implicit link=0000₁₆,
1=LONG — four-octet link.node.selector)

F final transaction in sequence, explicitly returning the token to LAS at the end of the current transaction, : (0=NOT FINAL, 1=FINAL)

on sequence-initiating (ES) and token-passing (PT) DLPDUs a third meaning applies: (0=RESTART, 1=CONTINUE) sequence or repetitively-scheduled activities

C.1.2 Nomenclature for address and other fields of DLPDUs

— symbol indicating an omitted field

d- prefix indicating a destination address

s- prefix indicating a source address

o- prefix indicating an optional field (which is always the last non-omitted field in the DLPDU)

[PP] brackets indicating implied priority field

DLSAP a DL-service-access-point address, whose explicit length is specified by the FC's L subfield

DL(SAP) either a group DL-address or a DLSAP-address, whose explicit length is specified by the FC's L subfield

DLCEP a DL-address used as a DL-connection-end-point address, whose explicit length is specified by the FC's L subfield

DLCEP/SAP either a DLCEP-address or a DLSAP-address, whose explicit length is specified by the FC's L subfield

DLCEP/(SAP) either a group DL-address or a DLSAP-address or a DLCEP-address, whose explicit length is specified by the FC's L subfield

DLSEP a DL-address used as a DL-scheduling-end-point address, whose explicit length is specified by the FC's L subfield

[d-DLCEP/SAP] brackets indicating implied d-DLCEP/SAP-address field, whose implied value is the same as that of the explicit or implied s-DLCEP/SAP-address field in the immediately-prior DLPDU, which must have been a CA, CD or ED DLPDU.

[s-DLCEP/SAP] brackets indicating implied s-DLCEP/SAP-address field, whose implied value is the same as that of the explicit d-DLCEP/SAP-address field in the immediately-prior DLPDU, which must have been a CA, CD or ED DLPDU.

C.1.3 Basic parameter and data fields of DLPDUs

link	individual link address (two octets), one component of LONG addresses; also used in time distribution (TD) DLPDUs
node	individual node address (one octet), one-component of SHORT and LONG addresses. Also used as a single octet address in some link-local DLPDUs used for inter-DLE coordination, where it is known as a node DL-address
DLSDU	complete Data Link Service Data Unit (non-null)
pDLSDU	partial or complete DLSDU (non-null)
SPDU	Support Protocol Data Unit (non-null) - a PDU used to convey scheduling and related DLS-support information between DLEs on the local or extended link.
SD-p	multi-octet SD (status and data-description) parameters of the CA, CD, ED and DT DLPDUs (see 9.4)
SR-p	parameters of the SR DLPDU (see 9.5)
EC-p	multi-octet parameters of the EC DLPDU (see 9.1)
DC-p	multi-octet parameters of the DC DLPDU (see 9.2)
RC-p	multi-octet parameters of the RC DLPDU (see 9.3)
TD-p	multi-octet parameters of the TD DLPDU (see 9.6)
RQ-p	multi-octet parameters of the RQ DLPDU (see 9.7)
RR-p	multi-octet parameters of the RR DLPDU (see 9.8)
PN-p	multi-octet parameters of the PN DLPDU (see 9.9)
DD-p	multi-octet parameters of the PT, ES and RI DLPDUs (see 9.10)

C.2 DLPDU short-form summary grouped by function

C.2.1 Connection-oriented DLPDUs

Establish Connection	1110 LF00	EC		s-DLCEP	s-DLSAP	EC-p	o-DLSDU
	1111 LF00	EC	d-DLCEP/(SAP)	s-DLCEP/SAP	s-DLSAP	EC-p	o-DLSDU
Disconnect Connection	0110 LF00	DC		s-DLCEP		DC-p	o-DLSDU
	0111 LF00	DC	d-DLCEP/(SAP)	s-DLCEP		DC-p	o-DLSDU
Reset Connection	0110 LFPP	RC		s-DLCEP		RC-p	o-DLSDU
	0111 LFPP	RC	d-DLCEP	s-DLCEP		RC-p	o-DLSDU
Compel Acknowledgment	1010 LFPP	CA	d-DLCEP	—		SD-p	o-pDLSDU
	1110 LFPP	CA	d-DLCEP	s-DLCEP/SAP		SD-p	o-pDLSDU
Compel Data	1011 LFPP	CD	d-DLCEP	—		o-SD-p	—
	1111 LFPP	CD	d-DLCEP	s-DLCEP/SAP		o-SD-p	—
Exchange Data	1000 LFPP	ED	d-DLCEP	—		SD-p	pDLSDU
	1100 LFPP	ED	d-DLCEP	s-DLCEP/SAP		SD-p	pDLSDU
Data	0101 0F00	DT [PP]		[s-DLCEP]		SD-p	o-pDLSDU
	0101 LFPP	DT		s-DLCEP		SD-p	o-pDLSDU
	1001 0F00	DT [PP]	[d-DLCEP/SAP]	—		SD-p	o-DLSDU
	1001 LFPP	DT	d-DLCEP	—		SD-p	o-pDLSDU
	1001 LFPP	DT	d-DLSAP	—		SD-p	o-DLSDU
	1101 LFPP	DT	d-DLCEP	s-DLCEP/SAP		SD-p	o-pDLSDU
Status Response	0001 0F11	SR	—	s-node		o-SR-p	—

C.2.2 Connectionless DLPDUs

Compel Acknowledgment	1110 LFPP	CA	d-DLSAP	s-DLSAP		SD-p	DLSDU
Compel Data	1111 LFPP	CD	d-DLSAP	s-DLSAP		SD-p	—
Exchange Data	1100 LFPP	ED	d-DLSAP	s-DLSAP		SD-p	DLSDU
Data	1001 0F00	DT [PP]	[d-DLSAP]	—		SD-p	o-DLSDU
	1001 LFPP	DT	d-DLSAP	—		SD-p	o-DLSDU
	1101 LFPP	DT	d-DL(SAP)	s-DLSAP		SD-p	o-DLSDU
Status Response	0001 0F11	SR	—	s-node		o-SR-p	—

NOTE The same FC values are used for connection-oriented and connectionless DLPDUs. In all cases, the structure of SD-parameters depends on characteristics of the DL-object addressed by the DLPDU's first explicit or implicit DL-address.

C.2.3 Time Support DLPDUs

Compel Time	0001 0F00	CT	—	—		—	—
Time Distribution	0001 0F01	TD	—	s-node		TD-p	—
Round-trip-delay Query	1100 0F00	RQ	d-node.0	s-node.0		RQ-p	—
Round-trip-delay Reply	1101 0F00	RR	d-node.0	s-node.0		RR-p	—

C.2.4 Link Management DLPDUs

Probe Node-address	0010 0110	PN	d-node	—	PN-p	—
Probe Response	0010 0111	PR	—	[s-node]	—	SPDU
Pass Token	0011 0FPP	PT	d-node	—	DD-p	—
Execute Sequence	1000 LF00	ES	d-DLSEP	—	DD-p	—
Return Token	0011 0100	RT	—	—	—	—
Request Interval	0010 0000	RI	—	—	DD-p	—
Claim LAS	0000 0001	CL	—	s-node	—	—
Transfer LAS	0000 0110	TL	d-node	—	TL-p	—
Status Response	0001 0F11	SR	—	s-node	o-SR-p	—
Wakeup	0000 0000	WK	d-node	—	—	—
Idle	0001 0F10	IDLE	—	—	—	o-DLSDU

C.2.5 Spare and Unusable DLPDU code points

spare	0000 0x11 0010 0x01 00xx 1xxx 0100 xxxx 101x xx00
unusable	0000 0010 0000 010x 0010 001x 0010 0100 0011 0000 0101 1x00 1001 1x00 110x 1x00

C.3 DLPDU short-form summary in alphabetic order of DLPDU names

Claim LAS	0000 0001	CL	—	s-node	—	—	
Compel Acknowledgment	1010 LFPP	CA	d-DLCEP	—	SD-p	o-pDLSDU	
	1110 LFPP	CA	d-DLCEP	s-DLCEP/SAP	SD-p	o-pDLSDU	
	1110 LFPP	CA	d-DLSAP	s-DLSAP	SD-p	DLSDU	
Compel Data	1011 LFPP	CD	d-DLCEP	—	o-SD-p	—	
	1111 LFPP	CD	d-DLCEP	s-DLCEP/SAP	o-SD-p	—	
	1111 LFPP	CD	d-DLSAP	s-DLSAP	SD-p	—	
Compel Time	0001 0F00	CT	—	—	—	—	
Disconnect Connection	0110 LF00	DC		s-DLCEP	DC-p	o-DLSDU	
	0111 LF00	DC	d-DLCEP/(SAP)	s- DLCEP	DC-p	o-DLSDU	
Data	0101 0F00	DT [PP]		[s-DLCEP]	SD-p	o-pDLSDU	
	0101 LFPP	DT		s-DLCEP	SD-p	o-pDLSDU	
	1001 0F00	DT [PP]	[d-DLCEP/SAP]	—	SD-p	o- DLSDU	
	1001 LFPP	DT	d-DLCEP	—	SD-p	o-pDLSDU	
	1001 LFPP	DT	d-DLSAP	—	SD-p	o- DLSDU	
	1101 LFPP	DT	d-DLCEP	s-DLCEP/SAP	SD-p	o-pDLSDU	
	1101 LFPP	DT	d-DL(SAP)	s-DLSAP	SD-p	o- DLSDU	
Establish Connection	1110 LF00	EC		s-DLCEP	s-DLSAP	EC-p	o-DLSDU
	1111 LF00	EC	d-DLCEP/(SAP)	s-DLCEP/SAP	s-DLSAP	EC-p	o-DLSDU
Exchange Data	1000 LFPP	ED	d-DLCEP	—	SD-p	pDLSDU	
	1100 LFPP	ED	d-DLCEP	s-DLCEP/SAP	SD-p	pDLSDU	
	1100 LFPP	ED	d-DLSAP	s-DLSAP	SD-p	DLSDU	
Execute Sequence	1000 LF00	ES	d-DLSEP	—	DD-p	—	
Idle	0001 0F10	IDLE	—	—	—	o-DLSDU	
Pass Token	0011 0FPP	PT	d-node	—	DD-p	—	
Probe Node-address	0010 0110	PN	d-node	—	PN-p	—	
Probe Response	0010 0111	PR	—	[s-node]	—	o-SPDU	
Request Interval	0010 0000	RI	—	—	DD-p	—	
Reset Connection	0110 LFPP	RC		s-DLCEP	RC-p	o-DLSDU	
	0111 LFPP	RC	d-DLCEP	s-DLCEP	RC-p	o-DLSDU	
Return Token	0011 0100	RT	—	—	—	—	
Round-trip-delay Query	1100 0F00	RQ	d-node.0	s-node.0	RQ-p	—	
Round-trip-delay Reply	1101 0F00	RR	d-node.0	s-node.0	RR-p	—	
Status Response	0001 0F11	SR	—	s-node	o-SR-p	—	
Time Distribution	0001 0F01	TD	—	s-node	TD-p	—	
Transfer LAS	0000 0110	TL	d-node	—	TL-p	—	
Wakeup	0000 0000	WK	d-node	—	—	—	

NOTE The same FC values are used for connection-oriented and connectionless DLPDUs. In all cases, the structure of SD-parameters depends on characteristics of the DL-object addressed by the DLPDU's first explicit or implicit DL-address.

C.4 DLPDU short-form summary in alphabetic order of DLPDU acronyms

Compel Acknowledgment	1010 LFPP	CA	d-DLCEP	—	SD-p	o-pDLSDU
	1110 LFPP	CA	d-DLCEP	s-DLCEP/SAP	SD-p	o-pDLSDU
	1110 LFPP	CA	d-DLSAP	s-DLSAP	SD-p	DLSDU
Compel Data	1011 LFPP	CD	d-DLCEP	—	o-SD-p	—
	1111 LFPP	CD	d-DLCEP	s-DLCEP/SAP	o-SD-p	—
	1111 LFPP	CD	d-DLSAP	s-DLSAP	SD-p	—
Claim LAS	0000 0001	CL	—	s-node	—	—
Compel Time	0001 0F00	CT	—	—	—	—
Disconnect Connection	0110 LF00	DC	—	s-DLCEP	DC-p	o-DLSDU
	0111 LF00	DC	d-DLCEP/(SAP)	s-DLCEP	DC-p	o-DLSDU
Data	0101 0F00	DT [pp]	—	[s-DLCEP]	SD-p	o-pDLSDU
	0101 LFPP	DT	—	s-DLCEP	SD-p	o-pDLSDU
	1001 0F00	DT [pp]	[d-DLCEP/SAP]	—	SD-p	o- DLSDU
	1001 LFPP	DT	d-DLCEP	—	SD-p	o-pDLSDU
	1001 LFPP	DT	d-DLSAP	—	SD-p	o- DLSDU
	1101 LFPP	DT	d-DLCEP	s-DLCEP/SAP	SD-p	o-pDLSDU
	1101 LFPP	DT	d-DL(SAP)	s-DLSAP	SD-p	o- DLSDU
	1110 LF00	EC	—	s-DLCEP	s-DLSAP	EC-p
Establish Connection	1111 LF00	EC	d-DLCEP/(SAP)	s-DLCEP/SAP	s-DLSAP	EC-p
	1111 LF00	EC	d-DLCEP/(SAP)	s-DLCEP/SAP	s-DLSAP	EC-p
Exchange Data	1000 LFPP	ED	d-DLCEP	—	SD-p	pDLSDU
	1100 LFPP	ED	d-DLCEP	s-DLCEP/SAP	SD-p	pDLSDU
	1100 LFPP	ED	d-DLSAP	s-DLSAP	SD-p	DLSDU
Execute Sequence	1000 LF00	ES	d-DLSEP	—	DD-p	—
Idle	0001 0F10	IDLE	—	—	—	o-DLSDU
Probe Node-address	0010 0110	PN	d-node	—	PN-p	—
Probe Response	0010 0111	PR	—	[s-node]	—	o-SPDU
Pass Token	0011 0FPP	PT	d-node	—	DD-p	—
Reset Connection	0110 LFPP	RC	—	s-DLCEP	RC-p	o-DLSDU
	0111 LFPP	RC	d-DLCEP	s-DLCEP	RC-p	o-DLSDU
Request Interval	0010 0000	RI	—	—	DD-p	—
Round-trip-delay Query	1100 0F00	RQ	d-node.0	s-node.0	RQ-p	—
Round-trip-delay Reply	1101 0F00	RR	d-node.0	s-node.0	RR-p	—
Return Token	0011 0100	RT	—	—	—	—
Status Response	0001 0F11	SR	—	s-node	o-SR-p	—
Time Distribution	0001 0F01	TD	—	s-node	TD-p	—
Transfer LAS	0000 0110	TL	d-node	—	TL-p	—
Wakeup	0000 0000	WK	d-node	—	—	—

NOTE The same FC values are used for connection-oriented and connectionless DLPDUs. In all cases, the structure of SD-parameters depends on characteristics of the DL-object addressed by the DLPDU's first explicit or implicit DL-address.

C.5 DLPDU FC code-point assignment matrix – overview and detail

C.5.1 DLPDU FC code-point assignment overview

The first octet of each DLPDU is a frame control (FC) octet which specifies the format of the DLPDU, including the length of all address fields in the DLPDU, and the number and specific roles of the address fields explicitly present in the DLPDU. The FC octet also specifies, where appropriate, the DLPDU's priority.

The encoding of the FC octet is largely orthogonal, as is shown in figure C.1. The exact function of each FC code point is shown in tables C.1 and C.2. In this figure and these tables, the high-order four bits of the FC octet are shown in the row heading on the left side of the figure or table, and the low-order four bits are shown in the column heading at the top of the figure or table.

The blocks shown in figure C.1 indicate the major functions of that section of the FC code point assignment matrix; a small number of unrelated FC code points are also included in each block.

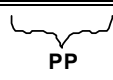
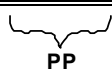
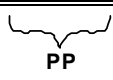
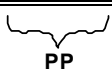
FC = xly	00__	01__	10__	11__
00__	DLPDUs with NODE DL-addresses	spare	DLPDUs with short DL-addresses	DLPDUs with long DL-addresses
01__				
10__				
11__				
priority				
token release				
DLPDU address scope				
	hort addresses		ng addresses	

Figure C.1 – Gross structure of FC code points

In Figure C.1, the PP bits take only the values 01, 10 and 11. The PP bits specify the priority of the DLPDU.

Table C.1 shows the generic assignment of the FC code points to the different DLPDUs. It also shows

- which address fields are explicitly present in the DLPDU (d for destination, s for source, and dss for the three addresses of a connection establishment DLPDU), and which are implicitly present (– for implicit source address);
- the common length of each of those d and s address fields, either long (four-octet DL-addresses), or short (one-octet NODE DL-addresses in the upper one-quarter of the table; two-octet DL-addresses in the lower three-quarters of the table, all implicitly on the local link).

Table C.1 – Generic assignment of FC code points

FC=x\y	yyyy	00_ _				01_ _				10_ _				11_ _			
xxxx		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
00_ _	00	WK d-	CL -s	don't use		don't use	don't use	TL d-									
	01	CT	TD -s	IDLE	SR [d]s	CT	TD -s	IDLE	SR [d]s								
	10	RI		don't use	don't use	don't use		PN d-	PR -[s]								
	11	don't use	PT d-			RT	PT d-										
01_ _	00																
	01	DT [s]	DT s			DT [s]	DT s			don't use	DT s			don't use	DT s		
	10	DC s	RC s			DC s	RC s			DC s	RC s			DC s	RC s		
	11	DC ds	RC ds			DC ds	RC ds			DC ds	RC ds			DC ds	RC ds		
10_ _	00	ES d-	ED d-			ES d-	ED d-			ES d-	ED d-			ES d-	ED d-		
	01	DT [d]-	DT d-			DT [d]-	DT d-			don't use	DT d-			don't use	DT d-		
	10		CA d-				CA d-				CA d-				CA d-		
	11		CD d-				CD d-				CD d-				CD d-		
11_ _	00	RQ ds	ED ds			RQ ds	ED ds			don't use	ED ds			don't use	ED ds		
	01	RR ds	DT ds			RR ds	DT ds			don't use	DT ds			don't use	DT ds		
	10	EC ss	CA ds			EC ss	CA ds			EC ss	CA ds			EC ss	CA ds		
	11	EC dss	CD ds			EC dss	CD ds			EC dss	CD ds			EC dss	CD ds		

C.5.2 DLPDU code-point assignment rationale

The structure of the frame-control code points is quite regular.

For example

- FC₃ is the L subfield, specifying the length of all addresses in the DLPDU: when FC₃ = 0 all addresses are SHORT (2 octets when FC₇₆₅₄ ≥ 0100; 1 octet otherwise); when FC₃ = 1 all addresses are LONG (4 octets when FC₇₆₅₄ ≥ 0100).
- FC₂ is the F subfield, specifying the location of the token at the end of the current transaction: In general, when FC₂ = 0 the token stays with the current token holder; when FC₂ = 1 the token returns to the LAS, or transfers to a new LAS, at the end of the current transaction.
- When FC₇₆₅₄ ≥ 0011 and FC₁₀ ≠ 00, then FC₁₀ is the PP subfield.

Table C.2 shows the individual assignment of the FC code points to the different DLPDUs.

Table C.2 – Individual assignment of FC code points

FC=x\y xxxx	yyyy	00_ _				01_ _				10_ _				11_ _			
		00	01	10	11	00	01	10	11	00	01	10	11	00	01	10	11
00_ _	00	WK	CL	don't use	spare	don't use	don't use	TL	spare	spare	spare	spare	spare	spare	spare	spare	spare
	01	CT	TD	IDLE	SR	CT	TD	IDLE	SR	spare	spare	spare	spare	spare	spare	spare	spare
	10	RI	spare	don't use	don't use	don't use	spare	PN	PR	spare	spare	spare	spare	spare	spare	spare	spare
	11	don't use	PT	PT	PT	RT	PT	PT	PT	spare	spare	spare	spare	spare	spare	spare	spare
01_ _	00	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare	spare
	01	DT	DT	DT	DT	DT	DT	DT	DT	don't use	DT	DT	DT	don't use	DT	DT	DT
	10	DC	RC	RC	RC	DC	RC	RC	RC	DC	RC	RC	RC	DC	RC	RC	RC
	11	DC	RC	RC	RC	DC	RC	RC	RC	DC	RC	RC	RC	DC	RC	RC	RC
10_ _	00	ES	ED	ED	ED	ES	ED	ED	ED	ES	ED	ED	ED	ES	ED	ED	ED
	01	DT	DT	DT	DT	DT	DT	DT	DT	don't use	DT	DT	DT	don't use	DT	DT	DT
	10	spare	CA	CA	CA	spare	CA	CA	CA	spare	CA	CA	CA	spare	CA	CA	CA
	11	spare	CD	CD	CD	spare	CD	CD	CD	spare	CD	CD	CD	spare	CD	CD	CD
11_ _	00	RQ	ED	ED	ED	RQ	ED	ED	ED	don't use	ED	ED	ED	don't use	ED	ED	ED
	01	RR	DT	DT	DT	RR	DT	DT	DT	don't use	DT	DT	DT	don't use	DT	DT	DT
	10	EC	CA	CA	CA	EC	CA	CA	CA	EC	CA	CA	CA	EC	CA	CA	CA
	11	EC	CD	CD	CD	EC	CD	CD	CD	EC	CD	CD	CD	EC	CD	CD	CD

C.6 SD-parameters (status and data-description parameters) of CA, CD, ED and DT DLPDUs

C.6.1 SD-p: Basic DLC parameters for CA, CD, ED and DT DLPDUs

Format code	Format summary	Description
A:	(null)	for use when all DLC directions are UNORDERED or not used, or when no pDLSDU or status is being conveyed
B:	(J, K, T, RR, SSS)	sent by a PEER DLC when sequence numbers are required, segmentation is never needed, and all window sizes are ≤ 3
C:	(RRRR, UUUU), (J, K, T, SSSS), (TTTT, VVVV)	sent by a PEER DLC when one of the two directions is either DISORDERED or CLASSICAL, sequence numbers are required in the other direction, and segmentation may be needed. may also be used in place of format B, D or E
D:	(RRRR, UUUU), (J, K, T, SSSS)	sent by a PEER DLC when one of the two directions is either DISORDERED or CLASSICAL, sequence numbers are required in the other direction, segmentation is never needed, and at least one window size is > 3 . may also be used in place of format E
E:	(RRRR, UUUU)	sent by a SUBSCRIBER, or the receiver of an ORDERED PEER DLC, to request a retransmission when segmentation may be needed and when the reverse direction is either UNORDERED or not used
F:	(J=0, K=0, T, SSSS), (TTTT, VVVV)	sent by a PUBLISHER or PEER DLC when sequence numbers are required, segmentation may be needed, and the reverse direction is either UNORDERED or not used
G:	(J=0, K=0, T, SSSS)	sent by a PUBLISHER or PEER DLC when sequence numbers are required, segmentation is never needed, and the reverse direction is either UNORDERED or not used
NOTE 1 Only formats A, C, F and G need be supported. Formats B, D and E are all optimizations of format C, and need not be supported.		
NOTE 2 CA, CD, DT and ED DLPDUs which provide connection-mode services use formats A to G.		

Subfield Name Subfield Description

J	Boolean indicating whether retransmission of segment UUUU of the NDR'th DLSDU is requested.
K	Boolean indicating whether all DLSDUs prior to the NDR'th DLSDU are being acknowledged.
T	Boolean indicating whether the associated DLSDU originated in a buffer with associated timeliness criteria, and that those timeliness criteria were met.
RR	2-bit residue (that is, the three low-order bits) of the sequence number NDR of a received DLSDU, or of the DLSDU before a missing DLSDU.
RRRR	4-bit residue (that is, the four low-order bits) of the sequence number NDR of a received DLSDU, or of the DLSDU before a missing DLSDU.
SSS	3-bit residue (that is, the three low-order bits) of the sequence number NDS of this DLSDU, or of the highest-numbered DLSDU sent on the DLC.
SSSSS	5-bit residue (that is, the five low-order bits) of the sequence number NDS of this DLSDU, or of the highest-numbered DLSDU sent on the DLC.

UUUU	4-bit zero-based segment number RSN of the DLSDU segment which needs to be retransmitted.
TTTT	4-bit zero-based total number TNS of DLSDU segments in the DLSDU, one segment of which is conveyed in the associated DLPDU.
VVVV	4-bit zero-based segment number ASN of the DLSDU segment conveyed in the associated DLPDU.
Z...Z	one or more bits of zero, as indicated by the number of "Z"s, reserved for future standard use.

C.6.2 SD-p: DLSDU source time stamp for CA, CD, ED and DT DLPDUs

	Low-order octets of DL-time, with a unit weight of 2 ⁻⁵ ms					
J	– (null) –					
K	– (null) –				two low-order octets of DL-time	
L	– (null) –			three low-order octets of DL-time		
M	DL-time					
	one octet	one octet	one octet	one octet	one octet	one octet

As in all OSI protocols, all multi-octet numeric fields are transmitted most-significant-octet first.

C.6.3 SD-p: Unitdata parameters and delayed-reply parameters

Format code	Format summary	Description
P:	(null)	for non-acknowledged unitdata DLPDUs with DL(SAP)s
R:	(ZZ,PP,YYYY)	for acknowledged-unitdata and exchange-unitdata request CA, CD and ED DLPDUs, all with DLSAPs
U:	(XXXX,YYYY)	for unitdata-acknowledgment and exchange-unitdata-reply DT and SR DLPDUs, all with DLSAPs
CA, CD and ED DLPDUs which provide connectionless services use only format R; DT DLPDUs with two DL-addresses which provide connectionless services use only format P; DT DLPDUs with one explicit or implicit destination DL-address and an implicit source DL-address which provide connectionless services use only format U; SR DLPDUs use format U.		

Subfield Name Subfield Description

PP	2-bit DLL priority of the DLSDU conveyed in the associated CA, CD or ED DLPDU, as previously specified (01=URGENT, 10=NORMAL, 11=TIME-AVAILABLE).
XXXX	4-bit reply status as specified in Table C.3 for SR and Table C.4 for appropriate DT DLPDUs.
YYYY	4-bit not-yet-confirmed.
ZZ	two bits of zero, reserved for future standard use.

C.6.4 SR-p: Status response parameters

Format code	Format summary	Description
X:	(null)	
Y:	(XXXX, ZZZZ)	for unitdata-acknowledgment and exchange-unitdata-reply error SR DLPDUs, all with DLSAPs

Subfield Name Subfield Description

XXXX 4-bit reply status as specified in Table C.3 for SR DLPDUs.

ZZZZ four bits of zero, reserved for future standard use.

C.6.5 Reply status for SR DLPDUs, and for unitdata-acknowledgment and exchange-unitdata-reply DT DLPDUs

Table C.3 – Reply status for SR DLPDUs

Short name	Definition	Hexadecimal coding
RR	failure — resource limitation in responder	4
RI	failure — responder DLE-class incompatible with this request	9
RTR	failure — LAS transfer rejected	B
BF	failure — fault in intermediary bridge	C
BR	temporary failure — resource limitation in intermediary bridge	D
BOK	interim success — intermediary bridge is forwarding transaction	E

**Table C.4 – Reply status for unitdata-acknowledgment
and exchange-unitdata-reply DT DLPDUs**

Short name	Definition	Hexadecimal coding
OK	success — no reply data available at responder or service does not provide this information	0
OK_U	success — URGENT priority reply data available at responder	1
OK_N	success — NORMAL priority reply data available at responder	2
OK_TA	success — TIME-AVAILABLE priority reply data available at responder	3
RR	failure — resource limitation in responder — no reply data available at responder or service does not provide this information	4
RR_U	failure — resource limitation in responder — URGENT priority reply data available at responder	5
RR_N	failure — resource limitation in responder — NORMAL priority reply data available at responder	6
RR_TA	failure — resource limitation in responder — TIME-AVAILABLE priority reply data available at responder	7
RF	failure — fault in responder	8
RI	failure — responder DL(SAP)-role incompatible with this DLPDU	9
RA	failure — response restricted to a different peer DLSAP-address	A
—	reserved for compatibility with the SR DLPDU's reply status — not available for other use	B
BF	failure — fault in intermediary bridge	C
BR	failure — resource limitation in intermediary bridge	D
BOK	reserved for interim success — intermediary bridge is forwarding transaction	E
DR	interim success — delayed reply; end station needs more time to prepare response	F

C.7 EC parameters of EC DLPDUs

EC-p:	(sender's R, NNN, Q, VVV),	-- 1 octet
	(sender's CC, PP, SS, XX),	-- 1 octet
	(sender's MCD _{CR}),	-- 2 octets
	(sender's MCD _D),	-- 2 octets
	(sender's TT _S , A _S , B _S , WWW _S),	-- 1 octet
	(sender's FFF _S , E _S , reserved, G _S , HH _S),	-- 1 octet
	(sender's M...M _S),	-- 2 octets
	(sender's TT _R , A _R , B _R , WWW _R),	-- 1 octet
	(sender's FFF _R , E _R , reserved, G _R , HH _R),	-- 1 octet
	(sender's M...M _R)	-- 2 octets

As in all OSI protocols, all multi-octet numeric fields are transmitted most-significant-octet first.

R	Boolean requiring return of a corresponding DLPDU (1=reply DLPDU requested)
NNN	publisher-DLCEP-address reuse-discriminator, or zero for non-publisher DL-addresses
Q	DLL path diversity (0=any-path, 1=this-path)
VVV	DLL protocol version number (always = 1 for this version)

CC	DLCEP class (01=PEER, 10=PUBLISHER, 11=SUBSCRIBER)
PP	DLL (and transaction) priority, as previously specified (01=urgent, 10=normal, 11=time-available)
SS	minimum DLL address size (01=LONG, 10=SHORT, 11=VERY-SHORT)
XX	DLPDU authentication (00=ordinary, 10=source, 11=maximal)
MCD _{CR}	negotiated maximum confirm delay for DL-Connect and DL-Reset requests 1 ≤ D...D ≤ 60 000, or 0xFFFF, where 0xFFFF represents the choice unlimited.
MCD _D	negotiated maximum confirm delay for DL-DATA requests 1 ≤ D...D ≤ 60 000, or 0xFFFF, where 0xFFFF represents the choice unlimited.

For the following parameters, an S subscript indicates the sender's desired attributes as a source of DLS-user data on the DLC, and an R subscript indicates the sender's desired attributes as a receiver of DLS-user data on the DLC.

TT	DLCEP features (00=unordered, 01=ordered, 10=disordered, 11=classical); =00 when M...M=00
A	Boolean requiring residual activity in the specified (sender-to-receiver or receiver-to-sender) direction (1=residual activity required)
B	Boolean local buffer/queue binding in the specified (sender-to-receiver or receiver-to-sender) direction is to a queue (=0) or to a buffer (=1)
WWW	window size (number of DLPDUs retained for possible retransmission; = 1 for BUFFER or UNORDERED; = min(K, 15) for QUEUE-K; default = 4 for implicit queue (unless overridden by local DL-management; =0 when M...M=0)
FFF	SD-p format (see C.6) in the specified direction (0=format A, 1=format B, 2=format C, 3=reserved, 4=format D, 5=format E, 6=format F, 7=format G)

NOTE Formats R and U (see C.6) are not used with the connection-oriented DLS.

E	Boolean indicating desire/ability to have sent/received ED DLPDUs contain a pDLSDU (1=pDLSDU desired/permitted in ED DLPDUs on this DLC)
reserved	one or more bits of zero, reserved for future standard use
G	Boolean indicating the presence (=1) or absence (=0) of a timeliness parameter in transmitted DLPDUs in the specified direction
HH	time-stamp-format in the specified direction (0=format J, 1=format K, 2=format L, 3=format M)
M...M	maximum DLSDU size $0 \leq M...M \leq 16 \times (M...M_{pp})$, based on the DLL priority PP, where $M...M_{Urgent}=64$, $M...M_{Normal}=128$ and $M...M_{Time-Available}=256$.

C.8 Parameters of DC and RC DLPDUs

C.8.1 DC parameters of DC DLPDUs

DC-p: (R, reserved, VVV), -- 1 octet
(RRRRRRRR), -- 1 octet

R Boolean requiring return of a corresponding DLPDU (1=return DLPDU requested)

reserved one or more bits of zero, reserved for future standard use

VVV DLL protocol version number (always = 1 for this version)

RRRRRRRR reason for disconnect (see C.8.3)

C.8.2 RC parameters of RC DLPDUs

RC-p: (R, reserved, VVV), -- 1 octet
(RRRRRRRR), -- 1 octet
(XXXSSSSS), -- 1 octet

R Boolean requiring return of a corresponding DLPDU (1=return DLPDU requested)

reserved one or more bits of zero, reserved for future standard use

VVV DLL protocol version number (always = 1 for this version)

RRRRRRRR reason for reset (see C.8.3)

XXXSSSSS 5-bit residue of the sequence number N_S preceding the next sequence number to be sent on the DLC, if any, with an arbitrary value for the three high-order bits of the octet.

C.8.3 DC and RC reason coding

Code	Reason for disconnect or reset	Reason class
00	user-originated disconnection — normal condition	user-originated disconnection (00..1F)
02	user-originated disconnection — abnormal condition	
1E	user-originated disconnection or connection rejection — reason unspecified	
20	user-originated connection rejection — connection not authorized, permanent condition	user-originated connection rejection (20..3F)
21	user-originated connection rejection — unacceptable QoS, permanent condition	
22	user-originated connection rejection — non-QoS reason, permanent condition	
24	user-originated connection rejection — transient condition	
40	provider-originated disconnection — incorrect DLCEP pairing, permanent condition	provider-originated disconnection (40..5F)
41	provider-originated disconnection — wrong publisher-DLCEP-address reuse-discriminator, permanent condition	
42	provider-originated disconnection — other permanent condition	
43	provider-originated disconnection — wrong DLPDU format or parameters, permanent condition	
44	provider-originated disconnection — wrong DLSDU size, permanent condition	
45	provider-originated disconnection — transient condition	
46	provider-originated disconnection — timeout	
5E	provider-originated disconnection or connection rejection — reason unspecified	
60	provider-originated connection rejection — DL(SAP) address unknown	provider-originated connection rejection (60..7D)
62	provider-originated connection rejection — DLSAP unreachable, permanent condition	
64	provider-originated connection rejection — DLSAP unreachable, transient condition	
65	provider-originated connection rejection — inconsistent DLCEP state, permanent condition	
66	provider-originated connection rejection — QoS unavailable, permanent condition	
68	provider-originated connection rejection — QoS unavailable, transient condition	
7E	disconnection or connection rejection, unknown origin — reason unspecified	unknown origin disconnect (7E..7F)
80	user-originated reset — resynchronization after user timeout	user-originated reset (80..BF)
82	user-originated reset — resynchronization after user-detected user-state inconsistencies	
9E	user-originated reset — reason unspecified	
C0	provider-originated reset — resynchronization after activation of a DL-management-established DLCEP	provider-originated reset (C0..FD)
C2	provider-originated reset — resynchronization after timeout	
C4	provider-originated reset — resynchronization after maximum number of retransmission requests or attempts	
C6	provider-originated reset — resynchronization after detected sequence number error	
C8	provider-originated reset — resynchronization after other detected DLCEP state inconsistencies	
FC	provider-originated reset — reason unspecified	
FE	reset, unknown origin — reason unspecified	unknown origin reset (FE..FF)

C.9 Parameters of TD, RQ and RR DLPDUs

C.9.1 General

C(NT) is the current value of the nodetimer, local to each DLE, zeroed at DLE startup.

NOTE The three low-order octets of C(NT), or of a value derived from C(NT),

- a) are appended to the TD, RQ and RR DLPDUs at the moment of transmission, with an approximately fixed delay relative to transmission of the last octet of the DLPDU;
- b) are appended to the DLPDU at the moment of reception, with an approximately fixed delay relative to receipt of the last octet of the DLPDU.

Thus the TD, RQ and RR DLPDUs can be considered to have three distinct stages in their formation:

- i) the DLPDU header as it exists in a transmit queue awaiting the opportunity to transmit, without either of its last two C(NT)-based fields appended;
- ii) the DLPDU as it is presented to and reported by the PhL, without its last C(NT)-based field appended;
- iii) the DLPDU as it exists in a receive queue awaiting post-reception processing by upper-DLL software, with all C(NT)-based fields appended.

N(NT) is a previously recorded value of some DLE's C(NT).

V(DLTO) (see 6.7.1.21) is the DL-time-offset, which equals the current value of DL-time minus (V(LSTO) + C(NT)).

As in all OSI protocols, all multi-octet numeric fields are transmitted most-significant-octet first.

C.9.2 TD-parameters of TD DLPDUs

TD-p:	(L...L),	-- 2 octets
	(TTT, LLL, SS),	-- 1 octet
	VS(DLTO),	-- 7 octets
	(VS(DLTO) + VS(LSTO)+CS(NT)),	-- 7 octets
	(A...A),	-- 3 octets, appended at moment of transmission
	(CR(NT)23..0)	-- 3 octets, appended at moment of reception

V_S(TSL) 16-bit link-id of the root link of the spanning (sub)tree from which the DL-time sense originated, that is, the link-id of the time-source, V_S(TSL)

L...L 16-bit link-id of the root link of the spanning (sub)tree from which the DL-time sense originated, that is, the link-id of the time-source, V_S(TSL)

TTT limiting time-synchronism class between time-source and sender, N_S(TQ)₇₋₅ (000=NONE, 001=1 s, 010=100 ms, 011=10 ms, 100=1 ms, 101=100 μs, 110=10 μs, 111=1 μs)

LLL number of intervening links on the DL-time propagation path from the DL-time source DLE to the sending DLE, N_S(TQ)₄₋₂,

- 0 indicates that the DL-time originates with the DLE itself,
 - 1 indicates that the DL-time originates with another DLE on the local link,
 - 2 indicates that the DL-time originates with a DLE on a link one bridge removed,
- and so forth;

SS time-source-type, expressing the method of synchronization of the DL-time source DLE with Universal Coordinated Time (UTC), the worldwide time standard, $N_S(TQ)_{1-0}$

00 some DLE's node-time, locally generated and not received from an extra-DLS-provider source,

01 local time (not UTC) received (directly or indirectly) from a human source,

10 UTC received (directly or indirectly) from a human source or an unreliable electronic source, or previously (but no longer) received (directly or indirectly) from a reliable electronic source,

11 UTC continuously received (directly or indirectly) from a reliable electronic source.

$V_S(DLTO)$ the DL-time-offset of the sending DLE during DLPDU formation shall be expressed as a signed seven-octet integer encoded most-significant-octet first, where the least significant bit represents a time granularity of approximately 2^{-13} ms (see Table C.5)

$V_S(DLTO) + V_S(LSTO) + C_S(NT)_1$ the DL-time of the sending DLE at a moment during DLPDU formation, less than 1,1 s prior to transmission, where the least significant bit represents a time granularity of approximately 2^{-13} ms

A...A a small adjustment to account for systemic offsets in the previous subfield, where the least significant bit represents a time granularity of approximately 2^{-13} ms, computed as $C_S(NT)_2 - C_S(NT)_1$, where $C_S(NT)_2$ represents the moment of end-of-transmission

$C_R(NT)_{23..0}$ low-order three octets of TD receiver's nodetimer at the moment of reception. This subfield does not appear on the medium (see Table C.6)

Table C.5 – Approximate numeric significance of the bits of seven-octet DL-time

Octet of DL-time in transmission order	Symbolic contents	Approximate weight of the low-order bit of the octet
1	0YYYYYYY	1,09 year
2	DDDDDDDD	1,55 day
3	HHHHHMMM	8,74 min
4	MMMSSSSS	2,05 s
5	Smmmmmmm	8,00 ms
6	mmm•µµµµµµ	31,25 µs
7	µµµµµnnn	122 ns

Table C.6 – Approximate numeric significance of the bits of N(NT), A...A, and three-octet C(NT)

Octet of short time in transmission order	Symbolic contents	Approximate weight of the low-order bit of the octet
1	Smmmmmmm	8,00 ms
2	mmm•µµµµµµ	31,25 µs
3	µµµµµnnn	122 ns

C.9.3 RQ-parameters of RQ DLPDUs

RQ-p:	(CS(NT)23..0)	-- 3 octets, appended at the moment of transmission
	(CR(NT)23..0)	-- 3 octets, appended at the moment of reception
	CS(NT)23..0	low-order three octets of RQ sender's nodetimer at the moment of RQ DLPDU formation just prior to (or during) transmission, representing the low-order three octets of the DLS-provider's current node-time in units of 2 ⁻¹³ ms
	CR(NT)23..0	low-order three octets of RQ receiver's nodetimer at moment of reception. This subfield does not appear on the medium

C.9.4 RR-parameters of RR DLPDUs

RR-p:	(TTT, E, reserved),	-- 1 octet
	(NR(NT)23..0),	-- 3 octets
	(NS(NT)23..0)	-- 3 octets
	(CS(NT)23..0),	-- 3 octets, appended at moment of transmission
	(CR(NT)23..0)	-- 3 octets, appended at moment of reception

TTT time-synchronism class of the replying DLE, N_S(TQ)₇₋₅, encoded as in C.9.2

E Boolean indicating whether a DLE maintains a real or estimated C(NT); true unless N_S(TQ)₇₋₅ specifies NONE

reserved one or more bits of zero, reserved for future standard use

N_R(NT)_{23..0} low-order three octets of RR receiver's (RQ sender's) nodetimer at the moment of transmission of the interrogating RQ DLPDU

N_S(NT)_{23..0} low-order three octets of RR sender's (RQ receiver's) nodetimer at the moment of reception of the interrogating RQ DLPDU, adjusted for any systemic offset in the value of the this and the next subfield

C_S(NT)_{23..0} low-order three octets of RR sender's nodetimer at the moment of RR DLPDU formation just prior to (or during) transmission, representing the low-order three octets of the DLS-provider's current node-time in units of 2⁻¹³ ms

C_R(NT)_{23..0} low-order three octets of RR receiver's nodetimer at the moment of reception. This subfield does not appear on the medium.

C.10 Parameters of PN, PT, ES and RI DLPDUs**C.10.1 General**

As in all OSI protocols, all multi-octet numeric fields are transmitted most-significant-octet first.

C.10.2 PN-parameters of PN DLPDUs

PN-p: (SSSS, reserved, VVV), -- 1 octet
 (GGGG, PPPP), -- 1 octet
 (TTTTTTTT TTTTTTTT), -- 2 octets
 (DDDDDDDD), -- 1 octet
 (MMMMMMMM) -- 1 octet

SSSS PhL maximum inter-channel signal skew

reserved one or more bits of zero, reserved for future standard use

VVV DLL protocol version number (always = 001 for this version)

GGGG PhL post-transmission-gap extension units

PPPP PhL preamble extension units

T...T slot-time, V(ST), $1 \leq T...T \leq 4\ 095$ octet-durations

D...D maximum-response-delay, V(MRD), $0 \leq D...D \leq 11$ slot-times

M...M minimum-inter-PDU-delay, V(MID), $0 \leq M...M \leq \min(120, (D...D - 1) \times T...T)$

C.10.3 DD-parameters (delegation-duration parameters) of PT, ES and RI DLPDUs

DD-p: (DDDDDDDD DDDDDDDD) -- 2 octets

D...D requested or delegated duration of token usage, V(RD), measured in octet-durations

C.11 Addressing summary extracted from figures and tables of 6.3

The figures and tables in this annex are reproduced from 6.3. They provide a convenient summary of the structure of DL-addresses and the preassigned ranges and uses of those DL-addresses.

Link designator	Sublink selector
2 octets	2 octets

**Figure C.2 [Figure 4] –
Basic structure of a DL-address**

Implied link designator (0000)	Node designator	Sub-node selector
	1 octet	1 octet

**Figure C.3 [Figure 5] –
Basic structure of a sublink selector**

Universally-administered MAC-addresses			
I/G	0	Globally-administered	Vendor-administered
1	1	22 bits	24 bits

Locally-administered MAC-addresses			
I/G	1	User-administered	
1	1	46 bits	

**Figure C.5 [Figure 7] –
Basic structure of MAC-addresses**

Link	Node	Selector
2 octets	1 octet	1 octet

or

Link	Node selector
2 octets	2 octets

or

Link Node Selector
4 octets

**Figure C.4 [Figure 6] –
DL-address alternative structures**

A DL-address as a locally-administered MAC-address			
I/G	1	Set by DL-management	Extended DL-address
1	1	14 bits	32 bits

**Figure C.6 [Figure 8] – Representation
of a DL-address as a MAC-address**

**Table C.7 [Table 2] –
Link || node || selector addressing**

Link	N S	Assigned use for specified DL-address range
0000 ... FFFF	0000 ... FFFF	local link DL-addresses as specified in Table 3.
0001 ... 0007	0000 ... 0007	flat non-local group DL-addresses reserved for DLL use
0001 ... 003F	0008 ... 003F	flat non-local group DL-addresses reserved for standard AEs
0001 ... 003F	0040 ... FFFF	flat non-local group DL-addresses available for vendor, user and dynamic assignment
0040 ... 007F	0000 ... FFFF	flat non-local globally-administered (see note) group DL-addresses — potentially one per vendor
0080 ... 0007	0000 ... 0007	flat non-local individual “DLSAP”-addresses reserved for DLL use
0080 ... 003F	0008 ... 003F	flat non-local individual DLSAP-addresses reserved for standard AEs
0080 ... 00FF	0040 ... FFFF	flat non-local individual DLSAP-addresses available for vendor, user and dynamic assignment
0100 ... 0FFF	0000 ... FFBF	flat non-local individual DL-addresses available for vendor, user and dynamic assignment as DLCEP-addresses and DLSEP-addresses
0FFF ... FFF7	FFC0 ... FFF7	flat non-local individual DL-addresses reserved for standard AE DLCEP-addresses and DLSEP-addresses
0FFF ... FFFF	FFF8 ... FFFF	flat non-local individual DL-addresses reserved for DLL use as DLCEP-addresses and DLSEP-addresses
1000 ... FEFF	0000 ... FFFF	Individual links of DL-addresses as specified in Table 3.
NOTE These group DL-addresses are based on the Organizationally Unique Identifiers specified in the ISO/IEC 8802 family of Local Area Network standards.		

**Table C.8 [Table 3] –
Link-local node || selector addressing**

N	S	Assigned use for specified DL-address range
00 ... FF	00 ... FF	local node DL-addresses as specified in Table 5.
01 ... 07	00 ... 07	flat link-local group DL-addresses reserved for DLL use
01 ... 3F	08 ... 3F	flat link-local group DL-addresses reserved for standard AEs
01 ... 03	40 ... FF	flat link-local group DL-addresses available for vendor, user and dynamic assignment
04 ... 07	00 ... 07	flat link-local individual “DLSAP”-addresses reserved for DLL use
04 ... 3F	08 ... 3F	flat link-local individual DLSAP-addresses reserved for standard AEs
04 ... FF	40 ... FF	flat link-local individual DLSAP-addresses available for vendor, user and dynamic assignment
05 ... 0F	00 ... BF	flat link-local individual DL-addresses available for vendor, user and dynamic assignment as DLCEP-addresses and DLSEP-addresses
0F ... F7	C0 ... F7	flat link-local individual DL-addresses reserved for standard AE DLCEP-addresses and DLSEP-addresses
0F ... FF	F8 ... FF	flat link-local individual DL-addresses reserved for DLL use as DLCEP-addresses and DLSEP-addresses
10 ... FF	00 ... FF	Individual nodes of DL-addresses as specified in Table 5.

**Table C.9 [Table 6] –
Predefined flat non-local DL-addresses**

link N S	Assigned use for specified DL-address
0001 0000	the DL-support functions of “all” (see note) DLEs on the extended link
0001 0001	the DL-support functions of “all” (see note) LM DLEs on the extended link
0001 0002	the DL-support functions of “all” (see note) bridge DLEs on the extended link
0001 0003	the DL-bridge functions of “all” (see note) bridge DLEs on the extended link
0001 0008	the SMAEs of “all” (see note) unconfigured DLEs on the extended link
0001 0009	the SMAEs of “all” (see note) DLEs on the extended link
0001 000A	the SMAEs of “all” (see note) LM DLEs on the extended link
0001 000B	the SMAEs of “all” (see note) bridge DLEs on the extended link
0001 000C	the SLAEs of “all” (see note) LoadServers on the extended link
0001 000D	the SLAEs of “all” (see note) Loadable Devices on the extended link
0080 0000	reserved for DLL use for a DL-support “DLSAP”-address (see note)
0080 0004	the “DLSAP”-address for the DL-bridge functions of the bridge DLE on the extended link which is the root bridge of the spanning tree
0FFF FFFE	the DLCEP-address for the DLC from the bridge functions of the bridge DLE on the extended link which is the root bridge of the spanning tree to the bridge functions in all other bridge DLEs on the extended link
0FFF FFFF	reserved for DLL use for a DL-support DLCEP-address or DLSEP-address (see note)

NOTE DLEs which do not recognize LONG DL-addresses are necessarily excluded from these sets

**Table C.10 [Table 7] –
Predefined flat link-local DL-addresses**

node selector	assigned use for specified DL-address
01 00	the DL-support functions of all DLEs on the link
01 01	the DL-support functions of all LM DLEs on the link
01 02	the DL-support functions of all bridge DLEs on the link
01 03	the DL-bridge functions of all bridge DLEs on the link
01 08	the SMAEs of all unconfigured DLEs on the link
01 09	the SMAEs of all DLEs on the link
01 0A	the SMAEs of all LM DLEs on the link
01 0B	the SMAEs of all bridge DLEs on the link
01 0C	the SLAEs of all LoadServers on the link
01 0D	the SLAEs of all LoadableDevices on the link
04 00	the “DLSAP”-address for the DL-support functions of the DLE on the link which is serving as LAS
04 04	the “DLSAP”-address for the DL-bridge functions of the bridge DLE on the link which is dominant (closest to the root) in the bridge spanning tree
0F FE	the DLCEP-address for the DLC from the bridge functions of the bridge DLE on the link which is dominant (closest to the root) in the bridge spanning tree to the bridge functions in all other bridge DLEs on the link
0F FF	the DLCEP-address for the DLC from the DL-support functions of the DLE on the link which is serving as LAS to the DL-support functions of all of the other LM DLEs on the link

**Table C.11 [Table 4] –
Link-local node designators**

Node	Assigned use for specified DL-address range
00	local DLE
01 ... 0F	unusable
10 ... 13	bridge-class DLEs
14 ...	link-master-class DLEs
...	unused
... F7	basic-class DLEs
F8 ... FB	non-"visitor" DLEs awaiting proper node designator assignment
FC ... FF	"visitor" DLEs

**Table C.12 [Table 5] –
Node-local selector addressing**

Selector	Assigned use for specified DL-address range
00 ... 01	node-local individual "DLSAP"-addresses reserved for DLL use
02 ... 07	node-local individual DLSAP-addresses reserved for standard AEs
08 ... 1F	node-local individual DLSAP-addresses available for vendor, user and dynamic assignment
20 ... F7	node-local individual DL-addresses available for vendor, user and dynamic assignment as DLCEP-addresses and DLSEP-addresses
F8 ... FF	node-local individual DL-addresses reserved for use as standard AE DLCEP-addresses and DLSEP-addresses

**Table C.13 [Table 8]–
Predefined node-local DL-addresses**

Selector	Assigned use for specified DL-address
00	the "DLSAP"-address for the DL-support functions of the node's DLE
01	the "DLSAP"-address for the DL-bridge functions of the node's DLE
02	the DLSAP-address for the node's SMAE
03	the DLSAP-address for the node's SLAE

Annex D **(informative)**

–

Type 2: Indicators and Switches

D.1 Purpose

A common approach to status indicators and switch behavior is recommended to provide a consistent human user interface for all devices conforming to these specifications. These indicators, typically LEDs, assist maintenance and diagnosis of problems with the Media, Physical and Data Link Layers.

NOTE Throughout this annex the term “network” is used to refer to that portion of the Ph-subnetwork and DL-subnetwork that is connected to the device making the observation.

D.2 General indicator requirements

If a product does support status indicators, they shall follow the specifications described in this clause.

If indicators are supported by a product, then two categories of status indicators shall be provided:

- One module status indicator;
- Two network status indicators.

Additional indicators may be present; however, the naming and symbol conventions of the standard indicators shall not be employed for other indicators.

NOTE Indicators, typically implemented as LEDs, help maintenance personnel to quickly identify a faulty unit or media. For this reason, it is strongly recommended to implement these indicators in any product for which no practical constraint prevents it.

D.3 Common indicator requirements

D.3.1 Applicability of common requirements

The common indicator requirement shall only apply to indicators for which requirements are specified in This specification.

D.3.2 Visibility of indicators

Indicators shall be visible without removing covers or parts from the equipment. Indicators shall be easily seen in normal lighting. Any related labels and icons shall be visible whether or not the indicator is illuminated.

D.3.3 Indicator flash rate

A two state indicator shall have a flash rate of $(1,0 \pm 0,1)$ Hz. The indicator shall have a duty cycle of $(50 \pm 5)\%$.

NOTE The indicator is in the first state for approximately 0,5 seconds and in the second state for approximately 0,5 seconds.

D.3.4 Indicators at power up

During power up, indicators shall turn to a fault or reset state. If turned on (illuminated), indicators shall remain so for at least 1 second. After completion of power up, indicators shall return to a normal operational state.

D.4 Module status indicator

D.4.1 Behavior

This bicolor (red/green) indicator shall provide device status. It shall indicate whether or not the device is powered, configured, and operating properly.

D.4.2 Labeling

The module status indicator shall be labeled with one of the following:

- “MS”;
- “OK”;
- “status”;
- “mod status”;
- “module status”.

D.5 Network status indicators

D.5.1 Description

The indication of network status shall require two bicolor (red/green) indicators. If the device has redundant PhLs (channels), each of the indicators shall be associated with one of the channels. If the device does not support channel redundancy, one of the indicators shall be associated with the single channel; however, two indicators shall still be required.

NOTE These indicators signify a number of conditions within the PhL; and when viewed together, they can reflect the state of the Data Link Layer.

D.5.2 Definitions

The following definitions shall be used to describe the indicator behavior:

- steady** – indicator shall be illuminated continuously;
- alternating** – both indicators, viewed together, shall alternate between two defined states, and the two indicators shall always be in opposite states, out of phase;
- flashing** – each indicator, viewed independently, shall alternate between two states, and if the indicators are flashing, they shall flash together, in phase.

D.5.3 Priority

The different states of the network status indicators shall follow a strict priority scheme. If more than one condition exists, the network status indicators shall display the state with the highest priority as specified in Table D.1.

Table D.1 – Network status indicators

Priority	Indicator state	How to view	Cause
highest (1)	both steady off	viewed together	reset or no power
2	both steady red		failed link interface
3	alternating red / green		self test
4	alternating red / off		bad node configuration (such as duplicate MAC ID)
5	steady off	viewed independently	channel disabled or channel not supported
6	flashing red / green		invalid link configuration
7	flashing red / off		link fault or no DLPDUs received
8	flashing green / off		temporary channel error, or listen-only
lowest (9)	steady green		normal operation

D.5.4 States

D.5.4.1 Steady green

During normal operation, when DLPDUs are being received without detected errors, the network status indicator for any enabled channel shall be steady green.

D.5.4.2 Steady red

If the link interface has faulted, both indicators shall be steady red.

D.5.4.3 Steady off

If one of the PhL entities is disabled, the indicator for the disabled channel shall be off. A channel can be disabled either because the device has only a single PhL or because it has two PhLs but exists on a non-redundant link. Only deactivating the entire network interface shall cause both indicators to be extinguished (off).

D.5.4.4 Flashing green / off

A flashing green / off pattern shall signify a temporary channel error. The two channels shall be treated independently with respect to these errors. An error on channel A shall be indicated by any of three events:

`Ph_status_indication` from channel A does not equal Normal;

`DLL_badFCS_indication(CHA)` is asserted;

`DLL_event_indication(DLL_EV_errA)` is asserted.

Likewise, an error on channel B shall be indicated by any of three events:

`Ph_status_indication` from channel B does not equal Normal;

`DLL_badFCS_indication(CHB)` is asserted;

`DLL_event_indication(DLL_EV_errB)` is asserted.

If more than one error occurs in a single DLPDU, only one channel error shall be declared. There shall be at most one error event per DLPDU per channel. Each PhL shall have an independent `Ph_status_indication`. The Data Link Layer shall report errors on at least the channel selected for presentation to higher levels. The DLL may also report errors on the other channel.

The flashing green state shall be entered whenever a channel error is detected. The flashing green state shall transition to the steady green state if no errors are detected for 4 ± 1 seconds on a given channel.

When the Data Link Layer is in a listen-only state, the indicator for any enabled channels shall flash green.

NOTE The ControlNet Object can be used to force the Data Link Layer to enter the listen-only state.

D.5.4.5 Flashing red / off

A flashing red / off pattern shall signify a severe link error. If the `Ph_frame_indication` from one of channels remains false for a $1,0 \pm 0,1$ second period, the respective indicator shall assume the flashing red state for the next $1,0 \pm 0,1$ second period.

For devices with redundant channels enabled, if the error rate for one of the channels exceeds that of the other channel by more than one error per N DLPDUs received, then the channel with the higher rate shall flash red. N shall be in the range 10^6 through 2×10^6 . The interval used to measure channel error rate shall be 5×10^6 DLPDUs.

D.5.4.6 Flashing red / green

A flashing red / green pattern shall signify a bad link configuration. The network status indicators shall enter the flashing red / green state if the network attachment monitor (NAM) indicates that the node has been asked to use invalid or unsupported link parameters.

NOTE To optionally indicate a receive queue overflow or a transmit queue underflow, the network status indicators should enter the flashing red / green state. The network status indicators should remain in this state at least 3 seconds even if the overflow condition has cleared.

D.5.4.7 Alternating red / off

An alternating red / off pattern shall signify an invalid node configuration. This state shall be entered if the `DLL_event_indication` has either of the following values:

`DLL_EV_dupNode;`

`DLL_EV_rogue.`

This state shall be left immediately when the error condition clears.

D.5.4.8 Alternating red / green

An alternating red / green pattern shall signify that the Data Link Layer is in a self-test mode. The network status indicators shall remain in this state for at least 3 seconds even if the self-test has completed.

D.5.5 NAP considerations

The network status indicators do not usually reflect any errors received on the NAP; however, nodes which have only a NAP (no other PhL variants) may use the network status indicators to represent the status of the NAP. If this optional behavior is implemented, the state of both the indicators shall reflect the state of NAP communication, instead of the A and B channels.

D.5.6 Labeling

The network status indicators shall be labeled using the symbols drawn in Figure D.1 and Figure D.2. The first channel shall be labeled with the outline of a channel icon. The second channel shall be labeled with a filled in channel icon. The first and second channels may be labeled with an uppercase A and B, respectively.

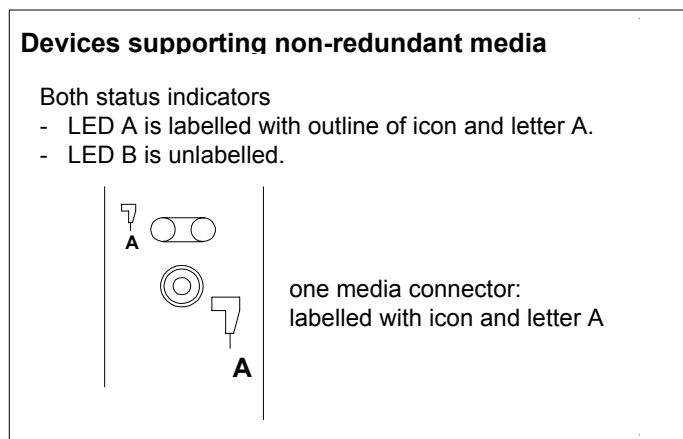


Figure D.1 – Non redundant network status indicator labeling

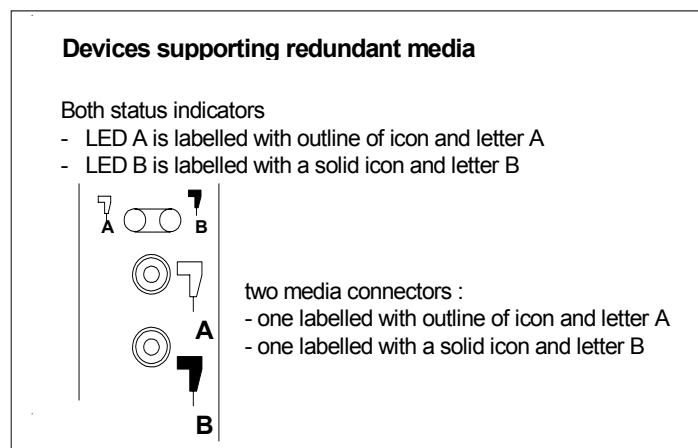


Figure D.2 – Redundant network status indicator labeling

D.6 Switches

D.6.1 General

All switches shall behave in the same way for the same function. For example, all network address switches for devices shall behave in the same way.

D.6.2 Standardizing switch behavior

D.6.2.1 Software override of switches

If a user accessible switch can be overwritten by software, then an indicator shall be present that indicates whether or not it has been overwritten.

D.6.2.2 Software override of switches

If an attribute is configurable with a switch and it cannot be overwritten with software, then the device shall respond with an error to a service which attempts to set the attribute.

D.6.2.3 Times that switches are valid

Switches shall comply with either of the following rules:

- the switch shall be live at all times;
- the switch shall be live only at power turn on (PTO).

D.6.3 Network address switches

D.6.3.1 Appearance

If the MAC ID is able to be set using switches, they shall be indicated in decimal format. This may be accomplished by using a rotary, thumbwheel, pushwheel, or other type of switch. The most significant digit shall be to the left or to the top of the least significant digit.

D.6.3.2 When to read switch

Network address switches shall only be read at power turn on (PTO). If the network address switches are changed after power turn on, a minor fault shall be detected and the module status indicator shall indicate a non-critical fault (flashing red). The MAC ID of the node used by the Data Link Layer shall remain at the value read during PTO. The current setting of the network address switches shall be stored in the ControlNet Object so that another node may remotely diagnose the non-critical fault.

D.6.3.3 Labeling

The network address switches shall be labeled with one of the following:

- “NA”;
- “address”;
- “net address”.

Annex E (normative)

–

Type 3: DL-Protocol state machines

NOTE 1 This annex specifies a number of finite state machines used by the DLE to provide its low-level and high-level protocol functions. This specification is complementary to the textual specification in the body of this standard; in case of conflict the requirements of this Annex take precedence.

NOTE 2 The finite state machine descriptions given here are necessarily less than a complete description of an implementation. Additional requirements and considerations are found in the textual specification.

E.1 Overall structure

The DL protocol of Type 3 consist of three portions:

Handling of the services invoked by Application Layer (DL- and DLM-services)

Media Access Control (Token handling and service interactions)

Interface to physical layer with PDU assembling/disassembling

The Interface between the service handler (FLC/DLM) and Media Access Control (MAC) consists of a set of Queues, a SAP-list and a DL-Data-Resource (This will be used also by the Send-Receive Unit –SRU).

The Interface between MAC and SRU consist of a set of services (see E.6.5).

The protocol sequences are described by means of State Machines.

In state diagrams states are represented as boxes state transitions are shown as arrows. Names of states and transitions of the state diagram correspond to the names in the textual listing of the state transitions.

The textual listing of the state transitions is structured as follows:

The first row contains the name of the transition.

The second row in define the current state.

The third row contains an optional event followed by Conditions starting with a “/” as first line character and finally followed by the Actions starting with a “=>” as first line character.

The last row contains the next state.

If the event occurs and the conditions are fulfilled the transition fires, that is, the actions are executed and the next state is entered.

Further description see IEC 61158-6 Conventions for Statemachines (Type 3).

Figure E.1 illustrates the structuring of the Protocol Machines.

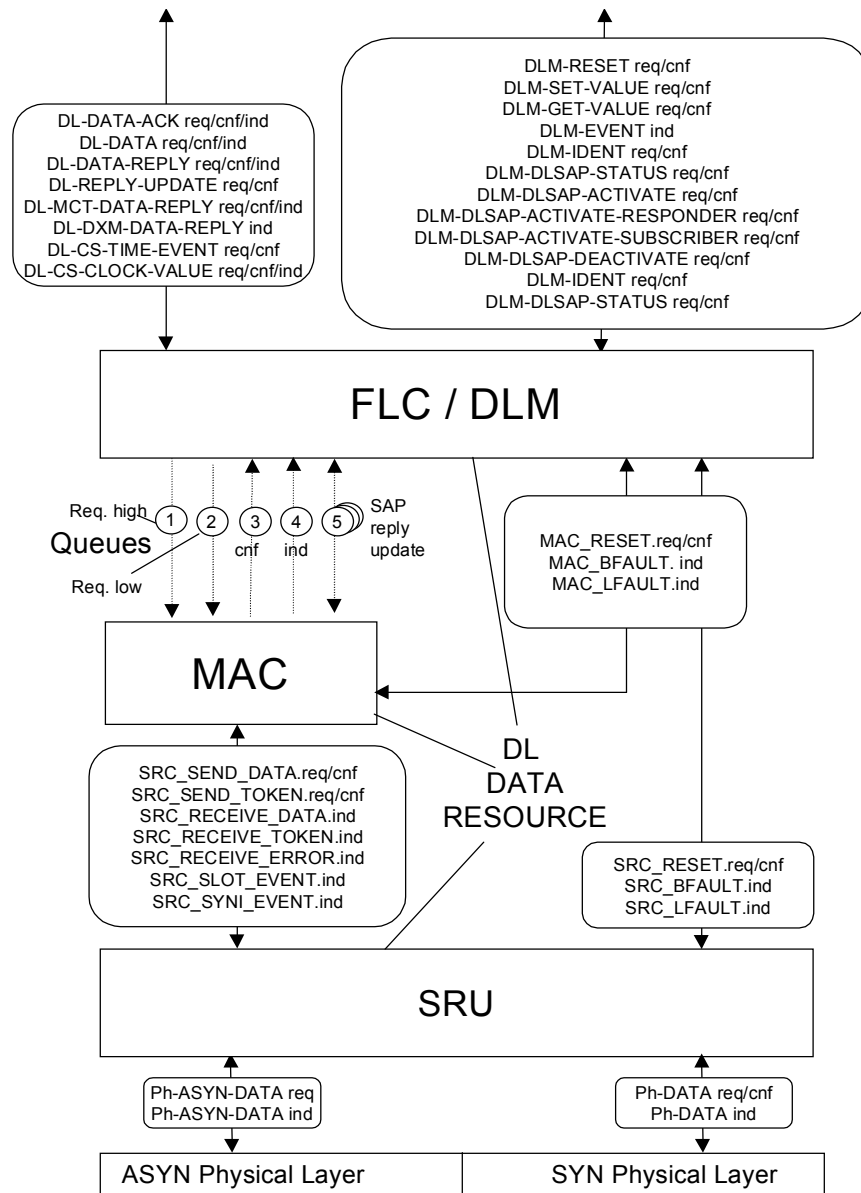


Figure E.1 – Structuring of the protocol machines

E.2 Variation of state machines in different devices

Table E.1 shows the assignment of State Machines parts to devices. The Layout of the State Machine Tables is defined in IEC 61158-6, State Machine Conventions.

Table E.1 – Assignment of state machines

	Machine	Options	Remarks
Slave	FLC	SDN/SDA/SRD.ind, Reply_Update.req/ind	support for Data rate- detection
	DLM	Limited set of FDL-variables No DLSAP-ACTIVATE-service	
	MAC	Only OFFLINE- and PASSIVE-IDLE- State	
	SRC	No Token No Request (sending) No Response (receiving)	
Master	FLC	Full set of services	
	DLM	Full set of services	
	MAC	Full set of states	
	SRC	Full set of services/states	
Only Master	MAC	Passive idle omitted No active/passive switching	Other option: Passive but no switching

E.3 DL Data Resource

The DL Data Resource models a data interface for all state machines of the Data Link Layer. It contains queues and buffers for exchanging data between the sublayers of the DL. Moreover it contains management information which have to be accessed by various sublayers of the DL. Table E.2 illustrates all Data Resources of the Data Link Layer.

Table E.2 – Data resource

Name	Struct element	Type	Range	Remark
FDL-Variables		S0		
SAP_List		[0..63, CS, NIL] of S1		
H_List		S2		
L_List		S2		
C_List		S3		
I_List		S3		
Ident_List		S12		
S0	Variables			
	TS	U8	0 to 126	DL-address of this station
	Data_rate		9,6; 19,2; 31,25; 45,45; 93,75; 187,5; 500; 1500, 3000; 6000; 12000 kBit/s and others	Data rate of this fieldbus
	Medium_ redundancy		single; redundant	Availability of redundant media
	HW-Release		LE_HR; hardware release identification	Hardware release number
	SW-Release		LE_SR; software release identification	Software release number
	SYNCHT		DSAP SSAP LSDU	Contents of the SYNCH DLPDU
	Tct	U32	1 to 2 ²⁴ -1 (bit times)	Isochronous cycle time
	Isochronous_ mode		0 1 2 3	non Isochronous with cycle correction with cycle drop TTR timer stop
	maxTsh	U8	1 to 256 (bit times)	maximal time shift
	Tcsi	U32	1 to 2 ³² -1 (bit times)	Clock Synchronization Interval Time
	Preamble_length	U8	8 to 64 (bit)	Length of the preamble of physical frames at synchronous transmission
	Tsyn	U8	4 to 32 (bit times)	post transmission gap time of synchronous transmission
	Tsl	U16	52 to 2 ¹⁶ -1 (bit times)	Slot time
	minTsdr	U16	2 ⁰ to 2 ¹⁶ -1 (bit times)	Smallest station delay time
	maxTsdr	U16	2 ⁰ to 2 ¹⁶ -1 (bit times)	Largest station delay time
	Tqui	U8	0 to 2 ⁸ -1 (bit times)	Transmitter fall time (line state uncertain time) or repeater switch time
	Tset	U8	2 ⁰ to 2 ⁸ -1 (bit times)	Setup time
	Ttr	U32	2 ⁰ to 2 ²⁴ -1 (bit times)	Target rotation time
	G	U8	1 to 100	GAP update factor
	in_ring_desired	Bool	true; false	Request entry into or exit out of the logical token ring

Table E.2 (continued)

Name	Struct element	Type	Range	Remark
	HSA	U8	0 to 126	Highest station address on this fieldbus
	max_retry_limit	U8	0 to 15 (preferably 0)	Maximum number of retries
	DLPDU_sent_count	U32	0 to 2 ³² -1	Number of DLPDUs sent
	Retry_count	U16	0 to 2 ¹⁶ -1	Number of DLPDU repeats
	DLPDU_sent_count_sr	[0..126] of U32	max. 126 entries of 0 to 2 ³² -1	List of numbers of DLPDUs sent per station
	Error_count	[0..126] of U16	max. 126 entries of 0 to 2 ¹⁶ -1	List of numbers of no or erroneous responses per station
	SD_count	U32	0 to 2 ³² -1	Number of correct start delimiters
	SD_error_count	U16	0 to 2 ¹⁶ -1	Number of defective start delimiters
	Trr	U32	2 ⁰ to 2 ²⁴ -1 (bit times)	Real rotation time
	LMS	[0..126] of U8	up to 127 DL-addresses	List of Master stations in the logical ring
	GAPL	[0..126] of DLE status	max. 126 DL-addresses (0 to 126) inclusive DLE status	List all of stations in the own GAP
	Cycle_violation_count	U32		Number of cycle violations which occurred since the start of isochronous mode
	Tid1	U16	33 bit + 2 bit + 2 × Tset + Tqui	Deduced Variables
	Tid2	U16	max (Tid1, maxTsdr)	
S1	SAP_List			
	Access	U8	0..126, NIL	
	Function_List_R	List of		SDN_H/L, SDA_H/L, SRD_H/L, MSR_D_H/L, CS(TE/CV)
	Function_List_I	List of		SDN_H/L, SDA_H/L, SRD_H/L, MSR_D_H/L, CS(TE/CV)
	LenList	[0..16] of U8	0..246	List of permitted DLSDU lengths depending on FC.Function
	Indication_mode		ALL/DATA/ UNCHANGED	
	Publisher_enabled		TRUE/FALSE	
	Ibuffer	S7		Indication Buffer
	Ubuffer	S8		Update Buffer
	Sbuffer	S7		Subscriber Buffer
S2	H_List/L_List			
	Num_entry	U16		
	First_entry	Ref to S9		
	Last_entry	Ref to S9		
	Insert()			
	Remove()			

Table E.2 (continued)

Name	Struct element	Type	Range	Remark
S3	C_List/I_List			
	Num_entry	U16		
	First_entry	Ref to S10		
	Last_entry	Ref to S10		
	Insert()			
	Remove()			
S4	REQM			
	DA	U8	0..126	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	Serv_class		high/low	
	Conf	Bool		
S5	RESM			
	DA	U8	0..126	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
S6	FC			
	Function		req rsp	SDN_H/L, SDA_H/L, SRD_H/L, MSRD_H/L, Ident, LSAP_Status, CS(TE/CV)
	Frame_type		req/rsp	
	FCB	U8	0,1	
	FCV	U8	0,1	
	Stn-type		Slave, M_n_rdy, M_rdy, M_in_ring	
S7	Ibuffer			
	Num_entry	U16		
	First_entry	Ref to S10		
	Last_entry	Ref to S10		
	Insert()			
	Remove()			
S8	Ubuffer			
	High_reference	U32		
	High_buffer	S11		
	High_transmit		SINGLE/MULTIPLE	
	Low_reference	U32		
	Low_buffer	S11		

	Low_transmit		SINGLE/MULTIPLE	
S9	H/L_List entry			
	Next entry	Ref to S9		
	DA	U8	0..127	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	conf	Bool		
S10	C/I_List entry			
	Next entry	Ref to S10		
	DA	U8	0..127	
	SA	U8	0..127	
	DSAP	U8	0..63, CS, NIL	
	SSAP	U8	0..63, CS, NIL	
	FC	S6		
	DLSDU	S11		
	conf	Bool		
	R_Status	NO,DL,DH,OK,RS, RR,UE,NR,RDH, RDL,NA,DS		
	Reference	U32		
S11	DLSDU			
	Len	U8	0..246	
	Data	[0..246] of U8		
S12	Ident_List			
	Vendor_Name	[0..32] of U8		
	Model_Name	[0..32] of U8		
	HW_Release	[0..32] of U8		
	SW_Release	[0..32] of U8		

E.4 FLC / DLM

E.4.1 Primitive definitions

E.4.1.1 Primitive exchanged between DL-User and FLC

Table E.3 shows all primitives issued by DL-User to the FLC. See IEC 61158-3, Type 3 for a description of the functions.

Table E.3 – Primitives issued by DL-User to FLC

Primitive name	Associated parameters
DL-DATA-ACK request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-DATA request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-DATA-REPLY request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU
DL-MCT-DATA-REPLY request	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU,
DL-REPLY-UPDATE request	Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference
DL-CS-TIME-EVENT request	D_addr, D_SAP_index, S_SAP_index
DL-CS-CLOCK-VALUE request	D_addr, D_SAP_index, S_SAP_index, DLSDU

Table E.4 shows all primitives issued by the FLC to the DL-User. See IEC 61158-3, Type 3 for a description of the functions.

Table E.4 – Primitives issued by FLC to DL-User

Primitive name	Associated parameters
DL-DATA-ACK confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA-REPLY confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status
DL-REPLY-UPDATE confirm	Service_class, S_SAP_index, DL_status
DL-MCT-DATA-REPLY confirm	Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status
DL-CS-TIME-EVENT confirm	D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status
DL-CS-CLOCK-VALUE confirm	D_addr, D_SAP_index, S_SAP_index, DL_status
DL-DATA-ACK indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-DATA indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-DATA-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference
DL-MCT-DATA-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference

Primitive name	Associated parameters
DL-DXM-REPLY indication	Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU
DL-CS-CLOCK-VALUE indication	D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Receive_delay_time CS_status

E.4.1.2 Primitive exchanged between DL-User and DLM

Table E.5 shows all primitives issued by the DL-User to the DLM. See IEC 61158-3, Type 3 for a description of the functions.

Table E.5 – Primitives issued by DL-User to DLM

Primitive name	Associated parameters
DLM-RESET request	(none)
DLM-SET-VALUE request	Variable_name(1 to n), Index(1 to k), Desired_value (1 to n)
DLM-GET-VALUE request	Variable_name(1 to n), Index(1 to k)
DLM-DLSAP-ACTIVATE request	S_SAP_index, Access, Service_list
DLM-DLSAP-ACTIVATE-RESPONDER request	S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled
DLM-DLSAP-ACTIVATE-SUBSCRIBER request	S_SAP_index, DLSDU_length_list
DLM-DLSAP-DEACTIVATE request	S_SAP_index
DLM-IDENT request	DL_addr
DLM-DLSAP-STATUS request	D_SAP_index, DL_addr

Table E.6 shows all primitives issued by the DLM to the DL-User. See IEC 61158-3, Type 3 for a description of the functions.

Table E.6 – Primitives issued by DLM to DL-User

Primitive name	Associated parameters
DLM-RESET confirm	DLM_Status
DLM-SET-VALUE confirm	Current_value(1 to n), DLM_status(1 to n)
DLM-GET-VALUE confirm	Desired_value(1 to n), DLM_status(1 to n)
DLM-EVENT indicate	Event/Fault TSH
DLM-DLSAP-ACTIVATE confirm	S_SAP_index, DLM_status
DLM-DLSAP-ACTIVATE-RESPONDER confirm	S_SAP_index, DLM_status
DLM-DLSAP-ACTIVATE-SUBSCRIBER confirm	S_SAP_index, DLM_status
DLM-DLSAP-DEACTIVATE confirm	S_SAP_index, DLM_status
DLM-IDENT confirm	DL-addr, Ident_list, DLM_status
DLM-DLSAP-STATUS confirm	D_SAP_Index, DL-addr, Access, Service_type(1 to n), Role_in_service_list(1 to n), DLM_status

E.4.1.3 Parameters of FLC Primitives

Table E.7 shows all parameters used with primitives between the DL-User and the FLC.

Table E.7 – Parameters used with primitives exchanged between DL-User and FLC

Parameter name	Description
S_SAP_index	Identifier of the local Service Access Point
D_SAP_index	Identifier of a remote Service Access Point
D_addr	Station address of the receiving DLE
S_addr	Station address of the sending DLE
Service_class	Indicates priority of the service
DL_status	Status of the service execution
DLSDU	Data Unit from or to the DLS-user
Transmit_strategy	Operation mode of the update buffer (single = buffer is transmitted once, multiple = buffer is transmitted repeated)
Reference	Reference to identify the DLSDU passed to the local DLE
Update_status	Indicates the presence of a DLSDU from a remote DLE in the update buffer
Send_delay_time	Protocol delay time between the invocation of the Clock Synchronization service request primitive and the transmission of the DLPDU
Receive_delay_time	Protocol delay time between the reception of the previous and the current DLPDU of Clock Synchronization
CS_status	Status of the Clock Synchronization sequence

E.4.1.4 Parameters of DLM primitives

Table E.8 shows all parameters used with primitives between the DL-User and the DLM.

Table E.8 – Parameters used with primitives exchanged between DL-User and DLM

Parameter name	Description
S_SAP_index	Identifier of the local Service Access Point
D_SAP_index	Identifier of a remote Service Access Point
Variable_name(1 to n)	List of DL variable names
Desired_value (1 to n)	List of DL variable values
Current_value (1 to n)	List of DL variable values
Access	Permission to use of a local SAP by one or several remote DLEs
Service_list	Type of services accepted at a local SAP
DLSDU_length_list	Maximum length of incoming or outgoing DLSDU at a local SAP
Indication_mode	Mode for Indication of received DLPDUs without user data
DLM_status	Status of the service execution
Event/Fault	Indicates the cause of an event or a fault
DL-addr	Station address of the local or a remote DLE
Ident_list	Specifies Vendor Name, Controller Type; HW/SW-Release
Service_type(1 to n)	Specifies the DL-services activated
Role_in_service_list(1 to n)	Specifies the role in service (initiator or responder or both)

E.4.2 State machine description

The FLC forms the interface between DL Protocol Machines and DL-User for SDA, SDN, SRD, MSRD and CS services.

The DLM forms the interface between DLM Protocol Machines and DLM-User for management services.

The FLC services are processed in the same way as the DLM services. For this reason the DLM module is described in the same state machine as the FLC module and is contained in the following description.

The services are all handled in the READY-State.

The service request will be validated first. A negative validation will result in a negative confirmation. Otherwise the service will be put into the high or low priority service queue according to the parameter service class. Services executed by MAC will be moved from the request queues to the confirmation queues. All valid incoming services will be put into the indication queue.

The set of SAP (de-)activate services are used to organise the SAP list. The Reply Update services are used to put reply data in the service list.

Local Variables

The FLC / DLM does not use local variables. All variables which have to be accessed by the FLC / DLM are contained in the DL Data Resource.

State Table Nomenclature

The standard suffixes “.req”, “.cnf” and “.ind” are used to indicate the request, confirm and indication primitives, respectively. Service primitive names are entirely upper-case.

E.4.3 FLC / DLM Table

The FLC and DLM State Table is shown in Table E.9.

Table E.9 – FLC / DLM state table

#	Current state	Event /condition ⇒action	Next state
1	READY	DLM-RESET.req ⇒ MAC_reset := FALSE SRC_reset := FALSE MAC_RESET.req SRC_RESET.req	WAIT_RES ET_CNF
2	WAIT_RES ET_CNF	MAC_RESET.cnf /SRC_reset ⇒ RESET_LIST DLM-RESET.cnf	READY
3	WAIT_RES ET_CNF	MAC_RESET.cnf /!SRC_reset ⇒ MAC_reset := TRUE	WAIT_RES ET_CNF
4	WAIT_RES ET_CNF	SRC_RESET.cnf /MAC_reset ⇒ RESET_LIST DLM-RESET.cnf	READY
5	WAIT_RES ET_CNF	SRC_RESET.cnf /!MAC_reset ⇒ SRC_reset := TRUE	WAIT_RES ET_CNF
6	READY	DLM-GET-VALUE.req (Variable_name((1 to n), Index(1 to k))) /!CHECK_PAR_READVALUE (Variable_name((1 to n), Index(1 to k))) ⇒ DLM_status (1 to n) := IV Current_value (1 to n) := NIL DLM-GET-VALUE.cnf (Current_value(1 to n),DLM_status(1 to n))	READY
7	READY	DLM-GET-VALUE.req (Variable_name((1 to n), Index(1 to k))) /CHECK_PAR_READVALUE (Variable_name((1 to n), Index(1 to k))) ⇒ Check_variable_names (Variable_name((1 to n), Index(1 to k))) Set_current_values (Variable_name((1 to n), Index(1 to k))) DLM-GET-VALUE.cnf (Current_value(1 to n),DLM_status(1 to n))	READY
8	READY	DLM-SET-VALUE.req (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) /!CHECK_PAR_SETVALUE (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) ⇒ DLM_status (1 to n) := IV DLM-SET-VALUE.cnf (DLM_status (1 to n))	READY
9	READY	DLM-SET-VALUE.req (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) /CHECK_PAR_SETVALUE (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) ⇒ Set_variable_list (Variable_name((1 to n), Index(1 to k)), Desired_value(1 to n)) DLM-SET-VALUE.cnf (DLM_status (1 to n))	READY
10	READY	MAC_BFAULT.ind (Fault_type, Tsh) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault, Tsh)	READY
11	READY	MAC_BFAULT.ind (Fault_type) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
12	READY	MAC_LFAULT.ind (Fault_type) ⇒ Event/Fault := Fault_type DLM-EVENT.ind (Event/Fault)	READY
13	READY	SRC_BFAULT.ind (Event_type) ⇒ Event/Fault := Event_type DLM-EVENT.ind (Event/Fault)	READY
14	READY	SRC_LFAULT.ind (Event_type) ⇒ Event/Fault := Event_type DLM-EVENT.ind (Event/Fault)	READY
15	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP ⇒ DLM_status := IV DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
16	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP && ACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
17	READY	DLM-DLSAP-ACTIVATE.req (S_SAP_index, Access, Service_list) /CHECK_PAR_DLSAP && !ACTIVATED (S_SAP_index) ⇒ SET_SAP_LIST (S_SAP_index, Access, Service_list) DLM_status := OK DLM-DLSAP-ACTIVATE.cnf (S_SAP_index, DLM_status)	READY
18	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES ⇒ DLM_status := IV DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
19	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !ACTIVATED (S_SAP_index) && Indication_mode≠UNCHANGED ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
20	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !ACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
21	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && !ACTIVATED (S_SAP_index) && Indication_mode≠UNCHANGED ⇒ SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) DLM_status := OK DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
22	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && /ACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED && CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list) ⇒ SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) DLM_status := OK DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
23	READY	DLM-DLSAP-ACTIVATE-RESPONDER.req (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled) /CHECK_PAR_DLSAP_RES && /ACTIVATED (S_SAP_index) && Indication_mode=UNCHANGED && !CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list) ⇒ DLM_status := NO DLM-DLSAP-ACTIVATE-RESPONDER.cnf (S_SAP_index, DLM_status)	READY
24	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB ⇒ DLM_status := IV DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
25	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB && SACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
26	READY	DLM-DLSAP-ACTIVATE-SUBSCRIBER.req (S_SAP_index, DLSDU_length_list) /CHECK_PAR_DLSAP_SUB && !SACTIVATED (S_SAP_index) ⇒ SET_SSAP_LIST (S_SAP_index, DLSDU_length_list) DLM_status := OK DLM-DLSAP-SUBSCRIBER.cnf (S_SAP_index, DLM_status)	READY
27	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT ⇒ DLM_status := IV DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
28	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT && !ACTIVATED (S_SAP_index) ⇒ DLM_status := NO DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
29	READY	DLM-DLSAP-DEACTIVATE.req (S_SAP_index) /CHECK_PAR_DLSAP_DEACT && ACTIVATED (S_SAP_index) ⇒ RESET_SAP_LIST (S_SAP_index) DLM_status := OK DLM-DLSAP-DEACTIVATE.cnf (S_SAP_index, DLM_status)	READY
30	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS ⇒ Access := NIL Service_list (1 to n) := NIL Role_in_service_list (1 to n) := NIL DLM_status := IV DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
31	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS && D_addr = Variables.TS && SAP_List[D_SAP_index] = NIL ⇒ Access := NIL Service_list (1 to n) := NIL Role_in_service_list (1 to n) := NIL DLM_status := LR DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
32	READY	DLM-DLSAP-STATUS.req (D_SAP_index, D_addr) /CHECK_PAR_STATUS && D_addr = Variables.TS && SAP_List ≠ NIL[D_SAP_index] ⇒ Write_SAPstatus_list() DLM_status := OK DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
33	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT ⇒ Ident_list := NIL DLM_status := IV DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
34	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && D_addr = Variables.TS && Ident_List = NIL ⇒ Ident_list := NIL DLM_status := LR DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
35	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && D_addr = Variables.TS && Ident_List ≠ NIL ⇒ Write_ident_list() DLM_status := OK DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
36	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && Ident_Pending = TRUE && D_addr ≠ Variables.TS ⇒ Ident_list := NIL DLM_status := LR DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
37	READY	DLM-IDENT.req (D_addr) /CHECK_PAR_IDENT && Ident_Pending = FALSE && D_addr ≠ Variables.TS ⇒ Ident_Pending := TRUE PUT_LREQ (IDENT)	READY
38	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN ⇒ DL_status := IV DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
39	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && !CHECK_SAP(Service_class, S_SAP_index, SDN) ⇒ DL_status := LS DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
40	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && !RESOURCE(S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
41	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (SDN_H)	READY
42	READY	DL-DATA.req (Service_class, D_Addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDN && CHECK_SAP(Service_class, S_SAP_index, SDN) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_LREQ (SDN_L)	READY
43	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA ⇒ DL_status := IV DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
44	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && !CHECK_SAP(Service_class, S_SAP_index, SDA) ⇒ DL_status := LS DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
45	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && !RESOURCE(S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
46	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class = high ⇒ PUT_HREQ (SDA_H)	READY
47	READY	DL-DATA-ACK.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_SDA && CHECK_SAP(Service_class, S_SAP_index, SDA) && RESOURCE(S_SAP_index, DLSDU.Len) && Service_class = low ⇒ PUT_LREQ (SDA_L)	READY
48	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY ⇒ DL_status := IV DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
49	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
50	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
51	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (SRD_H)	READY
52	READY	DL-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_LREQ (SRD_L)	READY
53	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY ⇒ DL_status := IV DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
54	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
55	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
56	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ PUT_HREQ (MSRD_H)	READY
57	READY	DL-MCT-DATA-REPLY.req (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU) /CHECK_PAR_REPLY && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ PUT_HREQ (MSRD_L)	READY
58	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /!CHECK_PAR_UPDATE ⇒ DL_status := IV DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
59	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && !CHECK_SAP (Service_class, S_SAP_index, SRD) ⇒ DL_status := LS DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
60	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) / CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && !RESOURCE (S_SAP_index, DLSDU.Len) ⇒ DL_status := LR DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
61	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=high ⇒ DL_status = OK PUT_HUBUFFER (S_SAP_index) DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
62	READY	DL-REPLY-UPDATE.req (Service_class, S_SAP_index, DLSDU, Transmit_strategy, Reference) /CHECK_PAR_UPDATE && CHECK_SAP (Service_class, S_SAP_index, SRD) && RESOURCE (S_SAP_index, DLSDU.Len) && Service_class=low ⇒ DL_status = OK PUT_LUBUFFER (S_SAP_index) DL-REPLY-UPDATE.cnf (S_SAP_index, DL_status)	READY
63	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = IDENT ⇒ Ident_pending := FALSE GET_IDENT_CNF() DLM-IDENT.cnf (D_addr, Ident_list, DLM_status)	READY
64	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = DLSAP-STATUS ⇒ Status_pending := FALSE GET_DLSAPSTATUS_CNF() DLM-DLSAP-STATUS.cnf (D_SAP_index, D_addr, Access, Service_type (1 to n), Role_in_service_list (1 to n), DLM_status)	READY
65	READY	/C_List.Num_entry≠0 && C_List.First_entry.Function = SDA_H C_List.First_entry.Function = SDA_L ⇒ GET_SDA/SDN_CNF() DL-DATA-ACK.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
66	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = SDN ⇒ GET_SDA/SDN_CNF() DL-DATA.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DL_status)	READY
67	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = SRD ⇒ GET_SRD_CNF() DL-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
68	READY	/C_List.Num_entry≠0 && C_List.First_entry.Service = MSRD ⇒ GET_SRD_CNF() DL-MCT-DATA-REPLY.cnf (Service_class, D_addr, D_SAP_index, S_SAP_index, DLSDU, DL_status)	READY
69	READY	/I_List.Num_entry≠0 && I_List.First_entry.FC.Function = SDA_H I_List.First_entry.FC.Function = SDA_L ⇒ GET_SDA/SDN_IND() DL-DATA-ACK.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU)	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
70	READY	/I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function= SDN_H I_List.First_entry.FC.Function = SDN_L ⇒ GET_SDA/SDN_IND() DL-DATA.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU)	READY
71	READY	/I_List.Num_entry≠0 && I_List.First_entry.DA ≠ 127 && I_List_entry.First_entry.FC.Function= SRD_H I_List.First_entry.FC.Function = SRD_L ⇒ GET_SRD_IND() DL-DATA-REPLY.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference)	READY
72	READY	/I_List.Num_entry≠0 && I_List.First_entry.DA = 127&& I_List_entry.First_entry.FC.Function= SRD_H I_List.First_entry.FC.Function = SRD_L ⇒ GET_DXM_IND() DL-DXM-REPLY.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU)	READY
73	READY	/I_List.Num_entry≠0 && I_List.First_entry.DA ≠ 127 && I_List_entry.First_entry.FC.Function= MSRD_H I_List.First_entry.FC.Function = MSRD_L ⇒ GET_SRD_IND() DL-MCT-DATA-REPLY.ind (Service_class, D_addr, D_SAP_index, S_addr, S_SAP_index, DLSDU, Update_status, Reference)	READY
101	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &!CHECK_PAR_TE ⇒ DL_status:=IV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
102	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &CHECK_PAR_TE && !CHECK_SAP(High, CS, TE/CV) ⇒ DL_status:=LS DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
103	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &CHECK_PAR_TE && CHECK_SAP(High, CS, TE/CV) && !RESOURCE(CS) ⇒ DL_status:=LR DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
104	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==STE &CHECK_PAR_TE && CHECK_SAP(High, CS, TE/CV) && RESOURCE(CS) ⇒ TSDT.start(2*Tcsi); PUT_HREQ (TE) Tm_State:=W_STE	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
105	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==STE ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
106	READY	/Tm_State==STE &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=CONFLICT	READY
107	READY	/Tm_State==W_STE &C_List.Num_entry≠0 && C_List.First_entry.Service=TE && C_List.First_entry.R_status=OK ⇒ Send_delay_time := 2*Tcsi-TSDT.cv; TSDT.stop; GET_TE_CNF(); Status:=OK DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=SCV	READY
108	READY	/Tm_State==W_STE &C_List.Num_entry≠0 && C_List.First_entry.Service=TE && C_List.First_entry.R_status≠OK ⇒ Send_delay_time := 2*Tcsi-TSDT.cv; TSDT.stop; GET_TE_CNF(); Status:=DS DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=STE	READY
109	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==W_STE ⇒ Send_delay_time := 0 DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=W_STE	READY
110	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==W_STE ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=W_STE	READY
111	READY	/Tm_State==W_STE &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=W_CC	READY
112	READY	/Tm_State==W_STE &TSDT expired ⇒ TSDT.start(4*Tcsi); Tm_State:=W_DS	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
113	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &!CHECK_PAR_CV ⇒ DL_status:=IV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=SCV	READY
114	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &CHECK_PAR_CV && !CHECK_SAP(High, CS, TE/CV) ⇒ DL_status:=LS DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=SCV	READY
115	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &CHECK_PAR_CV && CHECK_SAP(High, CS, TE/CV) && !RESOURCE(CS) ⇒ DL_status:=LR DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=SCV	READY
116	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==SCV &CHECK_PAR_CV && CHECK_SAP(High, CS, TE/CV) && RESOURCE(CS) ⇒ DLSDU:=CS_list; TSDT.start(2*Tcsi); PUT_HREQ (CV) Tm_State:=W_SCV	READY
117	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==SCV ⇒ Send_delay_time := 0; DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=SCV	READY
118	READY	/Tm_State==SCV &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=CONFLICT	READY
119	READY	/Tm_State==W_SCV &C_List.Num_entry≠0 && C_List.First_entry.Service=CV && C_List.First_entry.R_status=OK ⇒ GET_CV_CNF(); Status:=OK DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
120	READY	/Tm_State==W_SCV &C_List.Num_entry≠0 && C_List.First_entry.Service=CV && C_List.First_entry.R_status≠OK ⇒ GET_CV_CNF(); Status:=DS DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
121	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==W_SCV ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=W_SCV	READY
122	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==W_SCV ⇒ Send_delay_time := 0; DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=W_SCV	READY
123	READY	/Tm_State==W_SCV &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi); Tm_State:=W_CC	READY
124	READY	/Tm_State==W_STE &TSDT expired ⇒ TSDT.start(4*Tcsi); Tm_State:=W_DS	READY
125	READY	DL-CS-TIME-EVENT.req(D_addr, D_SAP_index, S_SAP_index) /Tm_State==CONFLICT ⇒ Send_delay_time := 0; DL_status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=CONFLICT	READY
126	READY	DL-CS-CLOCK-VALUE.req (D_addr, D_SAP_index, S_SAP_index, CS_list) /Tm_State==CONFLICT ⇒ DL_status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=CONFLICT	READY
127	READY	/Tm_State==CONFLICT &Tr_State ≠ W_TE && TM < TS ⇒ TSDT.start(4*Tcsi) Tm_State:=CONFLICT	READY
128	READY	/Tm_State==CONFLICT &TSDT expired ⇒ Tm_State:=STE	READY
129	READY	/Tm_State==W_CC &C_List.Num_entry≠0 && C_List.First_entry.Service=TE ⇒ Send_delay_time := 0; GET_TE_CNF(); Status:=SV DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, Send_delay_time, DL_status) Tm_State:=CONFLICT	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
130	READY	/Tm_State==W_CC &C_List.Num_entry≠0 && C_List.First_entry.Service=CV ⇒ GET_CV_CNF(); Status:=SV DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=CONFLICT	READY
131	READY	/Tm_State==W_DS &C_List.Num_entry≠0 && C_List.First_entry.Service=TE ⇒ GET_TE_CNF(); Status:=DS DL-CS-TIME-EVENT.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
132	READY	/Tm_State==W_DS &C_List.Num_entry≠0 && C_List.First_entry.Service=CV ⇒ GET_CV_CNF(); Status:=DS DL-CS-CLOCK-VALUE.cnf (D_addr, D_SAP_index, S_SAP_index, DL_status) Tm_State:=STE	READY
201	READY	/Tr_State==W_TE &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = TE ⇒ TRDT.start(2*Tcsi); GET_TE/CV_IND(); TM:=S_addr Tr_State:=W_CV	READY
202	READY	/Tr_State==W_TE &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = CV ⇒ GET_TE/CV_IND() Tr_State:=W_TE	READY
203	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = TE && I_List_entry.First_entry.SA ≠ TM ⇒ GET_TE/CV_IND() Tr_State:=W_CV	READY
204	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = TE && I_List_entry.First_entry.SA = TM ⇒ TRDT.stop; Receive_delay_time := 2*Tcsi – TRDT.cv; CS_list := DLSDU; CS_status:=SV; GET_TE/CV_IND() DL-CS-CLOCK-VALUE.ind (D_addr, D_SAP_index, S_addr, S_SAP_index, CS_list, CS_status, Receive_delay_time) Tr_State:=W_TE	READY
205	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = CV && I_List_entry.First_entry.SA ≠ TM ⇒ GET_TE/CV_IND() Tr_State:=W_CV	READY

Table E.9 (continued)

#	Current state	Event /condition ⇒action	Next state
206	READY	/Tr_State==W_CV &I_List.Num_entry≠0 && I_List_entry.First_entry.FC.Function = CV && I_List_entry.First_entry.SA = TM ⇒ TRDT.stop; Receive_delay_time := 2*Tcsi – TRDT.cv; CS_list := DLSDU; CS_status:=SV; GET_TE/CV_IND() DL-CS-CLOCK-VALUE.ind (D_addr, D_SAP_index, S_addr, S_SAP_index, CS_list, CS_status, Receive_delay_time) Tr_State:=W_TE	READY
207	READY	/Tr_State==W_CV &TCSI expired ⇒ Receive_delay_time := 0; CS_list := NULL; CS_status:=SV DL-CS-CLOCK-VALUE.ind (D_addr, D_SAP_index, S_addr, S_SAP_index, CS_list, CS_status, Receive_delay_time) Tr_State:=W_TE	READY

E.4.4 Functions

The FLC and DLM Functions are summarized in Table E.10.

Table E.10 – FLC / DLM function table

Function name	Operations
ACTIVATED (S_SAP_index)	Check that SAP is activated by setting its entry to valid value. SAP_List(S_SAP_index) ≠ NIL
RACTIVATED (S_SAP_index)	Check that SAP is activated as Responder.
SACTIVATED (S_SAP_index)	Check that SAP is activated as Subscriber.
CHECK_PAR_DLSAP	Check that all parameters of Service DLM-DLSAP-ACTIVATE.req are valid. Number of paramters must be 3. S_SAP_index: 0..63, NIL Access: 1..127 Service_list-Service_list_length: 1..(1 + 4 × 3) Service_list.n-th service_activate: “SDA” or “SDN” or “SRD” Service_list.n-th role_in_service: “INITIATOR” or “RESPONDER” (SDA, SDN only) or “BOTH” (SDA, SDN only) Service_list. nth DLSDU_length_list (1 to 4): 0..246
CHECK_PAR_DLSAP_DEACT	Check that all parameters of Service DLM-DLSAP-ACTIVATE-RESPONDER.req are valid. Number of paramters must be 1. S_SAP_index: 0..63, NIL
CHECK_PAR_DLSAP_SUB	Check that all parameters of Service DLM-DLSAP-ACTIVATE-RESPONDER.req are valid. Number of paramters must be 2. S_SAP_index: 0..62, NIL DLSDU_length_list (1 to 2): 0..246
CHECK_PAR_DLSAP_RES	Check that all parameters of Service DLM-DLSAP-ACTIVATE-RESPONDER.req are valid. Number of paramters must be 4. S_SAP_index: 0..62, NIL Access: 1..127 DLSDU_length_list (1 to 4): 0..246 Indication_mode: “ALL” or “DATA” or “UNCHANGED”
CHECK_PAR_IDENT	Check that all parameters of Service DLM-IDENT.req are valid. Number of paramters must be 1. D_addr: 0..126

Table E.10 (continued)

Function name	Operations
CHECK_PAR_READVALUE (Variable_name((1 to n), Index(1 to k)))	Check that the requested management variable names are valid. Possible values for MAC sublayer are: - in_ring_desired - Data_rate - TS - Ttr - G - HSA - max_retry_limit - SYNCHT - Tct - Isochronous_mode - maxTsh - Tcsi - HW_Release - SW_Release - Trr - LMS - GAPL Possible values for SRC sublayer are: - minTsdr - maxTsdr - Tsl - Tqui - Tset - Medium_redundancy - DLPDU_sent_count - Retry_count - DLPDU_sent_count_sr(1 to n) - Error_count(1 to n) - SD_count - SD_error_count
CHECK_PAR_SETVALUE (Variable_name((1 to n), Index(1 to k)))	Check that the requested management variable names are valid. Possible values for MAC sublayer are: - in_ring_desired - Data_rate - TS - Ttr - G - HSA - max_retry_limit - SYNCHT - Tct - Isochronous_mode - maxTsh - Tcsi - HW_Release - SW_Release Possible values for SRC sublayer are: - minTsdr - maxTsdr - Tsl - Tqui - Tset - Medium_redundancy - DLPDU_sent_count - Retry_count - DLPDU_sent_count_sr(1 to n) - Error_count(1 to n) - SD_count - SD_error_count
CHECK_PAR_SDA	Check that all parameters of Service DL-DATA-ACK.req are valid. Number of paramters must be 5. Service_class: "high" or "low" D_addr: 0..126 D_SAP_index: 0..62, NIL S_SAP_index: 0..62, NIL DLSDU.Len: 1..246 DLSDU.Data: according to DLSDU.Len

Table E.10 (continued)

Function name	Operations
CHECK_PAR_SDN	Check that all parameters of Service DL-DATA.req are valid. Number of paramters must be 5. Service_class: "high" or "low" D_addr: 0..127 D_SAP_index: 0..63, NIL S_SAP_index: 0..62, NIL DLSDU.Len: 1..246 DLSDU.Data: according to DLSDU.Len
CHECK_PAR_STATUS	Check that all parameters of Service DLM-DLSAP-STATUS.req are valid. Number of paramters must be 2. DLSAP: 0..63, NIL, CS D_addr: 0..126
CHECK_PAR_REPLY	Check that all parameters of Service DL-DATA-REPLY.req are valid. Number of paramters must be 5. Service_class: "high" or "low" D_addr: 0..126 D_SAP_index: 0..62, NIL S_SAP_index: 0..62, NIL DLSDU.Len: 0..246 DLSDU.Data: according to DLSDU.Len
CHECK_PAR_UPDATE	Check that all parameters of Service DL-REPLY-UPDATE.req are valid. Number of paramters must be 4. Service_class: "high" or "low" S_SAP_index: 0..62, NIL DLSDU.Len: 0..246 DLSDU.Data: according to DLSDU.Len Transmit_strategy: "SINGLE" or "MULTIPLE"
CHECK_PAR_TE	Check that all parameters of Service DL-CS-TIME-EVENT.req are valid. Number of paramters must be 3. D_addr = 127 D_SAP_index = CS S_SAP_index = CS
CHECK_PAR_CV	Check that all parameters of Service DL-CS-CLOCK-VALUE.req are valid. Number of paramters must be 4. D_addr = 127 D_SAP_index = CS S_SAP_index = CS Length of DLSDU: 18
CHECK_SAP (Service_class, S_SAP_index, Function)	Check that the SAP is activated to perform the current service request. SAP_List(S_SAP_index) ≠ NIL SAP_List(S_SAP_index).Function_List_I contains combination (Service_class, Function)
CHECK_SAP_LIST (S_SAP_index, DLSDU_length_list)	Check that the parameters of the service primitive conform to the settings of the SAP. DLSDU_length_list.Max_DLSDU_length_req_low = SAP_List(S_SAP_index).Ubuffer.Low_buffer.DLSDU.Len) DLSDU_length_list.Max_DLSDU_length_req_high = SAP_List(S_SAP_index).Ubuffer.High_buffer.DLSDU.Len) DLSDU_length_list.Max_DLSDU_length_ind_low = SAP_List(S_SAP_index).Ibuffer.Low_len) DLSDU_length_list.Max_DLSDU_length_ind_high = SAP_List(S_SAP_index).Ibuffer.High_len)
Check_variable_names (Variable_name((1 to n), Index(1 to k)))	Check that the management variables are set to valid values. LOOP for i from 1 to n for all variable names: if Variables.Variable_name (i) = NIL DLM_status (i) := NO else DLM_status (i) := OK

Table E.10 (continued)

Function name	Operations
GET_DLSAPSTATUS_CNF ()	Get a management service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP retrieve Access from C_List.First_Entry.DLSDU.Data retrieve Service_type (1 to n) from C_List.First_Entry.DLSDU.Data retrieve Role_in_service_list (1 to n) from C_List.First_Entry.DLSDU.Data DLM_status := C_List.First_Entry.R_Status C_List.Remove()
GET_IDENT_CNF ()	Get a management service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA Ident_list := C_List.First_Entry.DLSDU.Data DLM_status := C_List.First_Entry.R_Status C_List.Remove()
GET_SDA/SDN_CNF ()	Get a service confirmation from the confirmation queue to MAC. C_List.Num_entry-- if (C_List.First_Entry.FC.Function = "SDA_H" or "SDN_H") Service_class := high if (C_List.First_Entry.FC.Function = "SDA_L" or "SDN_L") Service_class := low D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP S_SAP_index := C_List.First_Entry.SSAP DL_status := C_List.First_Entry.R_Status C_List.Remove()
GET_SDA/SDN_IND ()	Get a service indication from the indication queue to MAC. I_List.Num_entry-- if (I_List.First_Entry.FC.Function = "SDA_H" or "SDN_H") Service_class := high if (I_List.First_Entry.FC.Function = "SDA_L" or "SDN_L") Service_class := low D_addr := I_List.First_Entry.DA D_SAP_index := I_List.First_Entry.DSAP S_addr := I_List.First_Entry.SA S_SAP_index := I_List.First_Entry.SSAP DLSDU := I_List.First_Entry.DLSDU I_List.Remove() SAP_List(S_SAP_index).Ibuffer.Insert() SAP_List(S_SAP_index).Ibuffer.Num_entry++
GET_SRD_CNF ()	Get a service confirmation from the confirmation queue to MAC. C_List.Num_entry-- if (C_List.First_Entry.FC.Function = "SRD_H") Service_class := high if (C_List.First_Entry.FC.Function = "SRD_L") Service_class := low D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP S_SAP_index := C_List.First_Entry.SSAP DLSDU := C_List.First_Entry.DLSDU DL_status := C_List.First_Entry.R_Status C_List.Remove()
GET_SRD_IND ()	Get a service indication from the indication queue to MAC. I_List.Num_Entry-- if (I_List.First_Entry.FC.Function = "SRD_H") Service_class := high if (I_List.First_Entry.FC.Function = "SRD_L") Service_class := low D_addr := I_List.First_Entry.DA D_SAP_index := I_List.First_Entry.DSAP S_addr := I_List.First_Entry.SA S_SAP_index := I_List.First_Entry.SSAP DLSDU := I_List.First_Entry.DLSDU R_Status := I_List.First_Entry.Update_status Reference := I_List.First_Entry.Reference I_List.Remove() SAP_List(S_SAP_index).Ibuffer.Insert() SAP_List(S_SAP_index).Ibuffer.Num_entry++

Table E.10 (continued)

Function name	Operations
GET_DXM_IND ()	Get a service indication from the indication queue to MAC. I_List.Num_entry-- if (I_List.First_Entry.FC.Function = "DH") Service_class := high else Service_class := low D_addr := I_List.First_Entry.DA D_SAP_index := I_List.First_Entry.DSAP S_addr := I_List.First_Entry.SA S_SAP_index := I_List.First_Entry.SSAP DLSDU := I_List.First_Entry.DLSDU I_List.Remove() SAP_List(S_SAP_index).Sbuffer.Insert() SAP_List(S_SAP_index).Sbuffer.Num_entry++
GET_TE_CNF	Get a service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP S_SAP_index := C_List.First_Entry.SSAP DL_status := C_List.First_Entry.R_Status C_List.Remove()
GET_CV_CNF	Get a service confirmation from the confirmation queue to MAC. C_List.Num_entry-- D_addr := C_List.First_Entry.DA D_SAP_index := C_List.First_Entry.DSAP S_SAP_index := C_List.First_Entry.SSAP DL_status := C_List.First_Entry.R_Status C_List.Remove()
GET_TE/CV_IND()	Get a service indication from the indication queue to MAC. I_List.Num_entry-- D_addr := I_List.First_Entry.DA D_SAP_index := CS S_addr := I_List.First_Entry.SA S_SAP_index := CS DLSDU := I_List.First_Entry.DLSDU I_List.Remove() SAP_List(S_SAP_index).Ibuffer.Insert() SAP_List(S_SAP_index).Ibuffer.Num_entry++ Get a service indication from the indication queue to MAC.
PUT_HREQ (Function)	Put a service request to the high prior service queue to MAC. H_List.Insert() H_List.First_Entry.DA := D_addr H_List.First_Entry.DSAP := D_SAP_index H_List.First_Entry.SSAP := S_SAP_index H_List.First_Entry.FC.Frame_type := req H_List.First_Entry.FC.Function := Function H_List.First_Entry.DLSDU := DLSDU H_List.Num_entry++
PUT_HUBUFFER (S_SAP_index)	Put a service request to the high prior update buffer of the LSAP. SAP_List(S_SAP_index).Ubuffer.High_reference := Reference SAP_List(S_SAP_index).Ubuffer.High_transmit := Transmit_strategy SAP_List(S_SAP_index).Ubuffer.High_buffer := DLSDU
PUT_LREQ (Function)	Put a service request to the low prior service queue to MAC. L_List.Insert() L_List.First_Entry.DA := D_addr L_List.First_Entry.DSAP := D_SAP_index L_List.First_Entry.SSAP := S_SAP_index L_List.First_Entry.FC.Frame_type := req L_List.First_Entry.FC.Function := Function L_List.First_Entry.DLSDU := DLSDU L_List.Num_entry++
PUT_LUBUFFER (S_SAP_index)	Put a service request to the low prior update buffer of the LSAP. SAP_List(S_SAP_index).Ubuffer.Low_reference := Reference SAP_List(S_SAP_index).Ubuffer.Low_transmit := Transmit_strategy SAP_List(S_SAP_index).Ubuffer.Low_buffer := DLSDU

Table E.10 (continued)

Function name	Operations
RESET_LIST	Reset all activated SAP by setting its entry to invalid value and reset all services queued in the data interface to MAC sublayer. SAP_List (0..63, NIL).I/Ubuffer.Remove() until empty SAP_List (0..63, NIL) := NIL I_List.Num_entry-- I_List.Remove() until empty C_List.Num_entry-- C_List.Remove() until empty
RESET_SAP_LIST (S_SAP_index)	Reset an activated SAP by setting its entry to invalid value. SAP_List(S_SAP_index) := NIL
RESOURCE (S_SAP_index, DLSDU.Len)	Check that local SAP resources are available to handle the requested data SAP_List(S_SAP_index) ≠ NIL SAP_List(S_SAP_index).Function_List_I contains combination (Service_class, Function) DLSDU.Len ≤ DLSDU_length_list-entry of combination (Service_class, Function)
Set_current_values (Variable_name((1 to n), Index(1 to k)))	Set the management variables for confirmation to DLMS-user. LOOP for i from 1 to n for all variable names: Current_value (i) := Variables.Variable_name (i)
SET_RSAP_LIST (S_SAP_index, DLSDU_length_list)	Activate a SAP as SRD responder by setting its entries to the requested values. SAP_List(S_SAP_index).Sbuffer.Low_len := DLSDU_length_list.Max_DLSDU_DXM_length_ind_low SAP_List(S_SAP_index).Sbuffer.High_len := DLSDU_length_list.Max_DLSDU_DXM_length_ind_high
SET_RSAP_LIST (S_SAP_index, Access, DLSDU_length_list, Indication_mode, Publisher_enabled)	Activate a SAP as SRD responder by setting its entries to the requested values. SAP_List(S_SAP_index).Access := Access SAP_List(S_SAP_index).Indication_mode := Indication_mode SAP_List(S_SAP_index).Publisher_enabled := Publisher_enabled SAP_List(S_SAP_index).Ubuffer.Low_buffer.DLSDU.Len) := DLSDU_length_list.Max_DLSDU_length_req_low SAP_List(S_SAP_index).Ubuffer.High_buffer.DLSDU.Len) := DLSDU_length_list.Max_DLSDU_length_req_high SAP_List(S_SAP_index).Ibuffer.Low_len := DLSDU_length_list.Max_DLSDU_length_ind_low SAP_List(S_SAP_index).Ibuffer.High_len := DLSDU_length_list.Max_DLSDU_length_ind_high
SET_SSAP_LIST (S_SAP_index, Access, Service_list)	Activate a SAP by setting its entries to the requested values. SAP_List(S_SAP_index).Access := Access compose SAP_List(S_SAP_index).Function_list_I based on Service_list compose SAP_List(S_SAP_index).Function_list_R based on Service_list

Table E.10 (continued)

Function name	Operations
Set_variable_list (Variable_name ((1 to n), Index(1 to k)), Desired_value (1 to n))	Check that the requested management variable names are inside their ranges and set them to the list of variables and set the corresponding DLM_status: LOOP for i from 1 to n for all variable names: vn := Variable_name(i) if (LOWLIM(vn) ≤ Desired_value(i) ≤ HIGHLIM(vn)) Variables.vn := Desired_value(i) DLM_status(i) := OK else DLM_status(i) := IV with following LOWLIM and HIGHLIM for variables of MAC sublayer: in_ring_desired = 0, 1 data_rate = 9,6 ... 12000 // kbit/s TS = 0 ... 126 Ttr = 1 ... 2²⁴-1 G = 1 ... 100 with Tgud = G * Ttr < 2°24 HSA = TS ... 126 max_retry_limit = 1 ... 8 SYNCHT = Octetstring with length 0 or 2 Tct = 1..2³²-1 maxTsh = 1..256 Tcsi = 1 ... 2³²-1 HW_Release = Visible String with length 0 ... 32 SW_Release = Visible String with length 0 ... 32 with following LOWLIM and HIGHLIM for variables of SRC sublayer: minTsdr = 1 ... 2¹⁶-1 maxTsdr = minTsdr ... 2¹⁶-1 Tsl = maxTsdr ... 2¹⁶-1 Tqui = 0 ... 255 Tset = 1 ... 255 DLPDU_sent_count = 0 Retry_count = 0 DLPDU_sent_count_sr(1 to n) = 0 Error_count(1 to n) = 0 SD_count = 0 SD_error_count = 0 with following LOWLIM and HIGHLIM for variables of PHY sublayer: Medium_redundancy = 0, 1 Interface_mode = "FULL_DUPLEX" or "HALF_DUPLEX" Loop_back_mode = "DISABLED" or "in MDS" or "in MAU" Preamble_extension = 0..7 Tptg = 0..7 Tics = 0..7 Transmitter_output_channel (1 to 8) = "ENABLED" or "DISABLED" Receiver_inpiut_channel (1 to 8) = "ENABLED" or "DISABLED" Preferred_receive_channel = "NONE" or 1..8
Write_ident_list ()	Compose parameter Ident_list for confirmation to DLMS user. Ident_list := Ident_List.Vendor_name + Ident_List.Model_name + Ident_List.HW_release + Ident_List.SW_release
Write_SAPstatus_list ()	Compose parameters Access, Service_type, Role_in_service_list for confirmation to DLMS user. Access := SAP_List(D_SAP_index).Access compose Service_type(1 to n) based on SAP_List(D_SAP_index).Sevice_listl/R compose Role_in_service_list(1 to n) based on SAP_List(D_SAP_index).Sevice_listl/R

E.5 MAC

E.5.1 Primitive definitions

E.5.1.1 Primitives exchanged between DLM and MAC

Table E.11 shows the primitives issued by the DLM to the MAC.

Table E.11 – Primitives issued by DLM to MAC

Primitive name	Associated parameters
MAC_RESET.req	none

Table E.12 shows the primitives issued by the MAC to the DLM.

Table E.12 – Primitives issued by MAC to DLM

Primitive name	Associated parameters
MAC_RESET.cnf	none
MAC_LFAULT.ind	Fault_type
MAC_BFAULT.ind	Fault_type Tj

E.5.1.2 Parameters of MAC primitives

Table E.13 shows the parameters used with primitives exchanged between the DLM and the MAC.

Table E.13 – Parameters used with primitives exchanged between DLM and MAC

Parameter name	Description
Fault_type	This parameter contains the error type. Possible values: State_conflict, Faulty_transceiver Double_token, Duplicate_address, Not_synchronized, Out_of_ring, Time_out, Hsa_error, State_conflict, In_ring
Tj	Jitter time in Isochronous Mode

E.5.2 State machine description

The Media Access Control is responsible for the message exchange and control of the various components using the media.

For a token based system, a main focus of this state machine is the token handling in all situations (see clause 20.3.2). This will be done in the states LISTEN-TOKEN, CLAIM-TOKEN, ACTIVE-IDLE (token-receipt) and PASS-TOKEN. To identify new stations ready to enter the ring, a FDL-Status request will be executed periodically. The reply will be checked in the state AWAIT-STATUS-RESPONSE. The state WAIT-TCT is used in isochronous mode to keep the interval between SYNCH requests constant.

The other main task is the correct execution of service sequences (Request, Response) including retries and duplication detection. This will be done by checking the high and low priority request queues when the station is token holder (state USE-TOKEN). In the state AWAIT-DATA-RESPONSE the reply to the processed service should be received. The state CHECK-TOKEN-PASS is used for checking the Token-hold-time.

The incoming service indications are processed in the states ACTIVE-IDLE, PASSIVE-IDLE and CHECK-TOKEN-PASS. All valid services received will be put into the indication queue and the appropriate reply activity will be invoked.

All local variables of the MAC are shown in Table E.14.

Local Variables**Table E.14 – Local MAC variables**

Struct element	Type	Range	Remark
FCB	[0..126] of U8	0,1	
FCV	[0..126] of U8	0,1	
REQM	S4		
RESM	S5		
Cnf	Bool		Single variables
Dup_add_count	U8	0, 1	
Gap_lp_cnt	U16		
Gap_to_do	Bool		
Gud_timer	U32		
Isochronous_Start	Bool		
LMS_cnt	U16		
Req_h_cnt	U16		
Retry_cnt	U8	0..15	
Second	U8	0, 1	
Token_holder	U8	0..126, NIL	
Tok_cnt	U8		
Tok_err_cnt	U8		
Ttr	U24		
T_cnt	U16	0..259	
TRT			Token Rotation Timer
cv	U24		Current Value of
Start(Ttr)	Trigger		Starting count down (from Ttr)

E.5.3 MAC state table

The MAC state table is shown in Table E.15.

Table E.15 – MAC state table

#	Current state	Event /condition ⇒action	Next state
1	OFFL	/H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	OFFL
2	OFFL	/L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	OFFL
3	OFFL	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
4	OFFL	/Data_Rate ≠ NIL && In_ring_desired ⇒ INIT_FCBV_LIST, RESM := empty, INIT_LMS, Isochronous_Start = TRUE, T_cnt:=0	LISTEN
5	OFFL	/Data_Rate ≠ NIL && !In_ring_desired ⇒ RESM := empty	PASSIVE_I
6	OFFL	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
7	OFFL	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
8	OFFL	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
9	OFFL	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
10	OFFL	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
11	OFFL	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
12	OFFL	SRC_SYNI_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
13	LISTEN	/!In_ring_desired ⇒	PASSIVE_I
14	LISTEN	/H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	LISTEN
15	LISTEN	/L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	LISTEN

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
16	LISTEN	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
17	LISTEN	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ T_cnt := 0	LISTEN
18	LISTEN	SRC_RECEIVE_ERROR.ind ⇒ T_cnt := 0	LISTEN
19	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /SA = TS DA = TS && Dup_add_count > 0 ⇒ Fault_type := Duplicate_address, T_cnt := 0, Dup_add_count := 0 MAC_BFAULT.ind (Fault_type)	OFFL
20	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA = TS SA = TS && Dup_add_count = 0 ⇒ Dup_add_count++, Tok_cnt--, T_cnt := 0	LISTEN
21	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /(DA ≠ TS && SA ≠ TS) && (DA > HSA SA > HSA) && Dup_hsa_count > 0 ⇒ Fault_type := Hsa_error, Dup_hsa_count := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	OFFL
22	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /(DA ≠ TS && SA ≠ TS) && (DA > HSA SA > HSA) && Dup_hsa_count = 0 ⇒ Dup_hsa_count++, Tok_cnt--, T_cnt := 0	LISTEN
23	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA ≤ HSA && SA ≤ HSA && First = NIL ⇒ BUILD_LMS(DA), Token_holder := DA, Tok_cnt--, T_cnt := 0, First := SA	LISTEN
24	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA ≤ HSA && SA ≤ HSA && First ≠ NIL && (SA ≠ Token_holder Tok_cnt=0) ⇒ INIT_LMS, BUILD_LMS(DA), Token_holder := DA, Tok_cnt--, T_cnt := 0, First := DA	LISTEN
25	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA ≤ HSA && SA ≤ HSA && First ≠ NIL && SA = Token_holder && Tok_cnt > 0 && DA ≠ First && LMS_cnt = 0 ⇒ LMS_UPDATE(DA, SA), Token_holder := DA, Tok_cnt--, T_cnt := 0	LISTEN
26	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA ≤ HSA && SA ≤ HSA && First ≠ NIL && SA = Token_holder && Tok_cnt > 0 && DA = First && LMS_cnt = 0 ⇒ LMS_UPDATE(DA, SA), Token_holder := DA, LMS_cnt := 1, Tok_cnt--, T_cnt := 0	LISTEN
27	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA ≤ HSA && SA ≤ HSA && First ≠ NIL && SA = Token_holder && LMS[SA]=DA && Tok_cnt > 0 && DA ≠ First && LMS_cnt = 1 ⇒ Token_holder := DA, Tok_cnt--, T_cnt := 0	LISTEN
28	LISTEN	SRC_RECEIVE_TOKEN.ind (DA, SA) /DA ≠ TS && SA ≠ TS && DA ≤ HSA && SA ≤ HSA && First ≠ NIL && SA = Token_holder && LMS[SA]=DA && Tok_cnt > 0 && DA = First && LMS_cnt = 1 ⇒ Tok_cnt--, T_cnt := 0, LMS_cnt++, RECV_ERR_cnt, SER_ACK_cnt := 0	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
29	LISTEN	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
30	LISTEN	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
31	LISTEN	SRC_SLOT_EVENT.ind /!TIME_OUT ⇒ T_cnt ++	LISTEN
32	LISTEN	SRC_SLOT_EVENT.ind /TIME_OUT ⇒ Fault_type := Time_out, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	CLAIM_T
33	LISTEN	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
34	ACTIVE_I	/!In_ring_desired ⇒	PASSIVE_I
35	ACTIVE_I	/LMS[TS] = NIL && H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	ACTIVE_I
36	ACTIVE_I	/LMS[TS] = NIL && L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	ACTIVE_I
37	ACTIVE_I	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
38	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = SA && FC.Frame = req && FC.Function = FDL_status && Isochronous_mode>0 ⇒ TRT_OFF	ACTIVE_I
39	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) = NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_rdy, FC.Function := OK, RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
40	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) ≠ NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_in_ring, FC.Function := OK, RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
41	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = Ident ⇒ DA := SA, SA := TS, DSAP <⇒ SSAP, RESM := empty, IDENT, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
42	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = DLSAP_status ⇒ DA := SA, SA := TS, DSAP <:= SSAP, RESM := empty, LSAP_STATUS, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
43	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = 127) && FC.Frame = res && SAP_CHECK(DSAP) && B_BUF(DSAP) ⇒ RESM := empty, SETUP_SIND(0,NO), T_cnt := 0	ACTIVE_I
44	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = 127) && FC.Frame = res && (!SAP_CHECK(DSAP) !B_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	ACTIVE_I
45	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = TE FC.Function = CV) && SAP_CHECK(CS) && I_BUF(CS) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
46	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = CV FC.Function = TE) && (!SAP_CHECK(CS) !I_BUF(CS)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
47	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = SDN_H FC.Function = SDN_L) && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
48	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = SDN_H FC.Function = SDN_L) && (!SAP_CHECK(DSAP) !I_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
49	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L FC.Function = SDA_H FC.Function = SRD_L FC.Function = SRD_H FC.Function = SRD_BCT) && RETRY ⇒ DA := RESM.DA, SA := TS, FC := RESM.FC, DSAP := RESM.DSAP, SSAP := RESM.SSAP, DLSDU := RESM.DLSDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
50	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_H FC.Function = SDA_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
51	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L FC.Function = SDA_H) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU, SSAP, DSAP := NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
52	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, if(SAP_List[DSAP].Indication_Mode = ALL Upd_status ≠ NO) SETUP_IND(Ref, Upd_status), DA := R_DA, SA := TS, FC.Function := Upd_status, FC.Frame := rsp, SETUP_STN_TYPE, DSAP ≤> SSAP, DLSDU := R_SDU, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
53	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, DA := R_DA, SA := TS, FC.Function := R_FUNCTION, FC.Frame := rsp, SETUP_STN_TYPE, DSAP <:= SSAP, DLSDU := R_SDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
54	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && !U_BUF(DSAP) ⇒ SETUP_IND(0, NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
55	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && !U_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU, SSAP, DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
56	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L FC.Function = SDA_H FC.Function = SDA_L) && !RETRY && !SAP_CHECK(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RS, DLSDU, SSAP, DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
57	ACTIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA ≠ TS && (DA≠127 (FC.Function ≠ SDN_H && FC.Function ≠ SDN_L)) FC.Frame ≠ req INVALID_FUNCTION) && !(DA = SA && FC.Frame = req && FC.Function = FDL_status && Isochronous_mode=0) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
58	ACTIVE_I	SRC_RECEIVE_ERROR.ind ⇒ T_cnt := 0, TRT_ON	ACTIVE_I
59	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count > 0 ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Dup_add_count := 0, Fault_type := Duplicate_address, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
60	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count = 0 ⇒ Dup_add_count++, TOK_CNT_UPD, Second := 0, RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
61	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && TOKEN_ERROR && Tok_err_cnt > (Limit - 2) ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Fault_type := Out_of_ring, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
62	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && TOKEN_ERROR && Tok_err_cnt ≤ (Limit - 2) ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Tok_err_cnt++, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
63	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] ≠ DA && DA ≠ TS ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
64	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] = DA && DA ≠ TS ⇒ RESM := empty, TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
65	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry ≠ 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
66	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second = 0 ⇒ RESM := empty, Second++, T_cnt := 0, TRT_ON	ACTIVE_I
67	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct = 0 && H_list.Num_entry ≠ 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, LMS_UPDATE(DA, SA), Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
68	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) (LMS[SA] ≤ SA && ((SA < TS) (TS < LMS[SA])))) ⇒ RESM := empty, TTH_INIT, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, GAP_INIT(0), Fault_Type := In_ring, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
69	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry = 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
70	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct=0 && H_list.Num_entry = 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, LMS_UPDATE(DA, SA), Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A
71	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct > 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
72	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct > 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE(DA, SA), TOK_CNT_UPD, LMS_UPDATE, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL,T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
73	ACTIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && (!LMS[LMS[SA]]=DA && !(LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) (LMS[SA] ≤ SA && ((SA < TS) (TS < LMS[SA]))))) ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
74	ACTIVE_I	SRC_SEND_DATA.cnf ⇒	ACTIVE_I
75	ACTIVE_I	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
76	ACTIVE_I	SRC_SLOT_EVENT.ind /!TIME_OUT ⇒ T_cnt ++, TRT_ON	ACTIVE_I
77	ACTIVE_I	SRC_SLOT_EVENT.ind /!TIME_OUT ⇒ RESM := empty, Fault_type := Time_out, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	CLAIM_T
78	ACTIVE_I	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
79	CLAIM_T	/LMS[TS] ≠ NIL && H_list.Num_entry ≠ 0 && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, Isochronous_Start = TRUE SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
80	CLAIM_T	/LMS[TS] ≠ NIL && H_list.Num_entry = 0 && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry, Isochronous_Start = TRUE	CHECK_A

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
81	CLAIM_T	/LMS[TS] ≠ NIL && Tct > 0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
82	CLAIM_T	/LMS[TS] = NIL ⇒ INIT_LMS, BUILD_LMS(TS), Second := 0, Fault_Type := In_ring, Retry_cnt := 0, DA := TS, SA := TS, TTH_INIT, if (NS := (TS+1) mod (HSA+1)) Gap_to_do := FALSE else Gap_to_do := TRUE, Gap_lp_cnt := 0 MAC_BFAULT.ind (Fault_type), SRC_SEND_TOKEN.req (DA, SA)	PASS_T
83	WAIT_TCT	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
84	WAIT_TCT	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
85	WAIT_TCT	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
86	WAIT_TCT	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
87	WAIT_TCT	SRC_SEND_DATA.cnf /Isochronous_Start ⇒ Tct:= TCT.cv, TCT.start(0), Fault_type:=SYNCH, DA:=0x7f,SSAP:=SYNCH.SSAP,DSAP:=SYNCH.DSAP FC.Function:=SDN_H,FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU:=SYNCH.DLSDU, Synch := TRUE, Cnf := FALSE, TRT_ON Isochronous_Start = FALSE SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind (Fault_type)	USE_T
88	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=1 && TCT.cv ≥ Tct ⇒ Tsh :=TCT.cv-Tct, TCT.start(2*Tct-TCT.cv), Fault_type:=Synch_Delay, DA:=0x7f,SSAP:=SYNCH.SSAP,DSAP:=SYNCH.DSAP FC.Function:=SDN_H,FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU:=SYNCH.DLSDU, Synch := TRUE, Cnf := FALSE, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind (Fault_type,Tsh)	USE_T
89	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=2 && TCT.cv ≥ Tct ⇒ Tsh :=Tct, TCT.start(2*Tct-TCT.cv), Fault_type:=SynchDelay, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU,SSAP,DSAP:=NIL SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind(Fault_Type,Tsh)	WAIT_TCT

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
90	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=0 && TCT.cv ≥ Tct && H_list.Num_entry ≠ 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry, SETUP_HREQ SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
91	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Isochronous_mode=0 && TCT.cv ≥ Tct && H_list.Num_entry = 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry	CHECK_A
92	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && Tct-Tpsp ≤ TCT.cv < Tct ⇒ TPSP.start(Tct-TCT.cv)	WAIT_TCT
93	WAIT_TCT	SRC_SEND_DATA.cnf /!Isochronous_Start && TCT.cv < Tct-Tpsp ⇒ DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU,SSAP,DSAP:=NIL SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
94	WAIT_TCT	TPSP expired /Isochronous_mode>0 ⇒ TCT.start(2*Tct-TCT.cv), Fault_type := SYNCH, DA:=0x7f,SSAP:=SYNCH.SSAP,DSAP:=SYNCH.DSAP FC.Function:=SDN_H,FC.Frame:=req, FC.FCB:=0, FC.FCV:=0, DLSDU:=SYNCH.DLSDU, Synch := TRUE, Cnf := FALSE, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU) MAC_BFAULT.ind(Fault_Type)	USE_T
95	WAIT_TCT	TPSP expired / (Isochronous_Mode=0 Isochronous_Mode=3) && H_list.Num_entry ≠ 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry, SETUP_HREQ SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
96	WAIT_TCT	TPSP expired / (Isochronous_Mode=0 Isochronous_Mode=3) && H_list.Num_entry = 0 ⇒ RESM := empty, Req_h_cnt := H_list.Num_entry, T_cnt := 0	CHECK_A
97	WAIT_TCT	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
98	WAIT_TCT	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
99	WAIT_TCT	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
100	USE_T	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
101	USE_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
102	USE_T	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
103	USE_T	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
104	USE_T	SRC_SEND_DATA.cnf /Cnf && !Gap_in_action ⇒ DLPDU_sent_count++ DLPDU_sent_count_sr[DA]++ SER_ACK_cnt++, if (SER_ACK_cnt := SER_ACK_limit) SER_ACK_cnt, RECV_ERR_cnt := 0	AW_DATA
105	USE_T	SRC_SEND_DATA.cnf /!Cnf && !Synch ⇒	CHECK_A
106	USE_T	SRC_SEND_DATA.cnf /!Cnf && Synch && TCT.cv > 143+ maxTsh ⇒ Synch := False, Fault_type := SynchDelay Tsh := TCT.cv - 143 MAC_BFAULT.ind (Fault_type, Tsh)	CHECK_A
107	USE_T	SRC_SEND_DATA.cnf /!Cnf && Synch && TCT.cv ≤ 143+ maxTsh ⇒ Synch = False	CHECK_A
108	USE_T	SRC_SEND_DATA.cnf /Cnf && Gap_in_action ⇒ Gap_in_action := FALSE, SER_ACK_cnt++, if (SER_ACK_cnt := SER_ACK_limit) SER_ACK_cnt, RECV_ERR_cnt := 0	AW_STATUS
109	USE_T	SRC_SEND_TOKEN.cnf (Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
110	USE_T	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
111	USE_T	SRC_SYNI_EVENT.ind /Gap_in_action ⇒ Gap_in_action := FALSE, Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
112	USE_T	SRC_SYNI_EVENT.ind /!Gap_in_action ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Not_synchronized MAC_BFAULT.ind (Fault_type)	ACTIVE_I
113	AW_DATA	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
114	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) / (FC.Frame = req (FC.Function ≠ SC && DA ≠ TS)) ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Double_token, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
115	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) / FC.Frame = rsp && DA = TS && FC.Function ≠ SC && (REQM.DA≠SA (DSAP,SSAP ≠ Nil && (REQM.DSAP ≠ SSAP REQM.SSAP ≠ DSAP))) ⇒ FCV_CLEAR, SETUP_CONM(DS), T_cnt := 0	ACTIVE_I
116	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) / FC.Frame = rsp && DA = TS && FC.Function ≠ SC && REQM.DA=SA && (DSAP,SSAP =Nil (REQM.DSAP = SSAP && REQM.SSAP = DSAP)) ⇒ SETUP_CON(FC, DLSDU), T_cnt := 0, FCB_UPDATE	CHECK_A
117	AW_DATA	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) / FC.Frame = rsp && FC.Function = SC ⇒ SETUP_CON(FC, DLSDU), T_cnt := 0, FCB_UPDATE	CHECK_A
118	AW_DATA	SRC_RECEIVE_ERROR.ind / RECV_ERR_cnt ≥ RECV_ERR_limit ⇒ FCV_CLEAR, SETUP_CONM(DS), T_cnt := 0	LISTEN
119	AW_DATA	SRC_RECEIVE_ERROR.ind / !(Isochronous_mode=2 && REQM.serv_class = high) && Retry_cnt > 0 && FCV = 1 && RECV_ERR_cnt < RECV_ERR_limit ⇒ Retry_Count--, Error_count[SA]--, DA := REQM.DA, SA := TS, FC := REQM.FC, DSAP := REQM.DSAP, SSAP := REQM.SSAP, DLSDU := REQM.DLSDU, Retry_cnt--, T_cnt := 0, RECV_ERR_cnt++ SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
120	AW_DATA	SRC_RECEIVE_ERROR.ind / !(Isochronous_mode=2 && REQM.serv_class = high) && (Retry_cnt = 0 FCV = 0) && RECV_ERR_cnt < RECV_ERR_limit ⇒ FCV_CLEAR, SETUP_CONM(NA), T_cnt := 0	CHECK_A
121	AW_DATA	SRC_RECEIVE_ERROR.ind / Isochronous_mode=2 && REQM.serv_class = high && RECV_ERR_cnt < RECV_ERR_limit ⇒ SETUP_CONM(NA), T_cnt := 0	CHECK_A
122	AW_DATA	SRC_RECEIVE_TOKEN.ind (DA, SA) / In_ring_desired ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Double_token, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
123	AW_DATA	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
124	AW_DATA	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
125	AW_DATA	SRC_SLOT_EVENT.ind /!(Isochronous_mode=2 && REQM.serv_class = high) && Retry_cnt > 0 && FCV = 1 ⇒ Retry_Count--, Error_count[SA]--, DA := REQM.DA, SA := TS, FC := REQM.FC, DSAP := REQM.DSAP, SSAP := REQM.SSAP, DLSDU := REQM.DLSDU, Retry_cnt-- SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
126	AW_DATA	SRC_SLOT_EVENT.ind /!(Isochronous_mode=2 && REQM.serv_class = high) && Retry_cnt = 0 FCV = 0 ⇒ FCV_CLEAR, SETUP_CONM(NA)	CHECK_A
127	AW_DATA	SRC_SLOT_EVENT.ind /Isochronous_mode=2 && REQM.serv_class = high ⇒ SETUP_CONM(NA)	CHECK_A
128	AW_DATA	SRC_SYNI_EVENT.ind /In_ring_desired ⇒ FCV_CLEAR, SETUP_CONM(DS), Fault_type := Not_synchronized MAC_BFAULT.ind (Fault_type)	ACTIVE_I
129	CHECK_A	/ (TTH_AVAILABLE Isochronous_Mode=2 Isochronous_Mode=3) && Req_h_cnt ≠ 0 ⇒ SETUP_HREQ, Req_h_cnt--, SA := TS SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
130	CHECK_A	/TTH_AVAILABLE && Req_h_cnt = 0 && Gap_lp_cnt = 0 ⇒ FC.Frame := req, FC.FCB := 0, FC.FCV := 0, FC.Function := FDL_status, DA := Gap_address, SA := TS, DLSDU, SSAP, DSAP := NIL, Cnf := TRUE, Gap_in_action := TRUE SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
131	CHECK_A	/TTH_AVAILABLE && Req_h_cnt = 0 && Gap_lp_cnt ≠ 0 && L_list.Num_entry ≠ 0 ⇒ SETUP_LREQ, SA := TS SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
132	CHECK_A	/!((TTH_AVAILABLE Isochronous_Mode=2 Isochronous_Mode=3) && Req_h_cnt <> 0) !TTH_AVAILABLE (Req_h_cnt = 0 && Gap_lp_cnt <> 0 && L_list.Num_entry = 0) ⇒ Second := 0, DA := NS(TS), SA := TS SRC_SEND_TOKEN.req (DA, SA)	PASS_T
133	PASS_T	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
134	PASS_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
135	PASS_T	SRC_RECEIVE_ERROR.ind ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
136	PASS_T	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ Fault_type := State_conflict, T_cnt := 0 MAC_LFAULT.ind (Fault_type)	OFFL
137	PASS_T	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
138	PASS_T	SRC_SEND_TOKEN.cnf(Status) /Status = token_pass_failed && Second = 0 ⇒ Second++, DS := NS(TS), SA := TS SRC_SEND_TOKEN.req (DA, SA)	PASS_T
139	PASS_T	SRC_SEND_TOKEN.cnf(Status) /Status = token_pass_failed && Second > 0 ⇒ Second := 0, Fault_type := Faulty_transceiver MAC_BFAULT.ind (Fault_type)	OFFL
140	PASS_T	SRC_SEND_TOKEN.cnf(Status) /Status = no_token_pass ⇒ Second := 0, Fault_type := Faulty_transceiver MAC_BFAULT.ind (Fault_type)	OFFL
141	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && !In_ring_desired ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
142	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && In_ring_desired && Tct > 0 ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
143	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && H_list.Num_entry ≠ 0 && In_ring_desired && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
144	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS = TS && Status = OK && H_list.Num_entry = 0 && In_ring_desired && Tct=0 ⇒ GAP_UPDATE, TTH_UPDATE, Req_h_cnt := H_list.Num_entry	CHECK_A
145	PASS_T	SRC_SEND_TOKEN.cnf(Status) /NS ≠ TS && Status = OK ⇒	CHECK_T
146	PASS_T	SRC_SLOT_EVENT.ind ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
147	PASS_T	SRC_SYNI_EVENT.ind /In_ring_desired ⇒ Fault_type := Not_synchronized MAC_BFAULT.ind (Fault_type)	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
148	CHECK_T	/LMS[TS] = NIL && H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	CHECK_T
149	CHECK_T	/LMS[TS] = NIL && L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	CHECK_T
150	CHECK_T	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
151	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = SA && FC.Frame = req && FC.Function = FDL_status && Tct > 0 ⇒ TRT_OFF	ACTIVE_I
152	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) = NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_rdy, FC.Function := OK, RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
153	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status && LMS(TS) ≠ NIL ⇒ DLSDU,SSAP,DSAP:=NIL, FC.Frame := rsp, FC.Stn-Type := M_in_ring, FC.Function := OK, RESM := empty, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
154	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = Ident ⇒ DA := SA, SA := TS, DSAP <⇒ SSAP, RESM := empty, IDENT, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
155	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = LSAP_status ⇒ DA := SA, SA := TS, DSAP <⇒ SSAP, RESM := empty, LSAP_STATUS, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
156	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = 127) && FC.Frame = res && SAP_CHECK(DSAP) && B_BUF(DSAP) ⇒ RESM := empty, SETUP_SIND(0,NO), T_cnt := 0	ACTIVE_I
157	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = 127) && FC.Frame = res && (!SAP_CHECK(DSAP) !B_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	ACTIVE_I
158	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = TE FC.Function = CV) && SAP_CHECK(CS) && I_BUF(CS) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
159	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = CV FC.Function = TE) && (!SAP_CHECK(CS) !I_BUF(CS)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
160	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = SDN_H FC.Function = SDN_L) && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ RESM := empty, SETUP_IND(0,NO), T_cnt := 0, TRT_ON	ACTIVE_I
161	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = SDN_H FC.Function = SDN_L) && (!SAP_CHECK(DSAP) !I_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
162	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L FC.Function = SDA_H FC.Function = SRD_L FC.Function = SRD_H FC.Function = SRD_BCT) && RETRY ⇒ DA := RESM.DA, SA := TS, FC := RESM.FC, DSAP := RESM.DSAP, SSAP := RESM.SSAP, DLSDU := RESM.DLSDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
163	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_H FC.Function = SDA_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
164	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L FC.Function = SDA_H) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU, SSAP, DSAP := NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
165	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, if(SAP_List[DSAP].Indication_Mode = ALL Upd_status ≠ NO) SETUP_IND(Ref, Upd_status), DA := R_DA, SA := TS, FC.Function := Upd_status, FC.Frame := rsp, SETUP_STN_TYPE, DSAP ≤> SSAP, DLSDU := R_SDU, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
166	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, DA := R_DA, SA := TS, FC.Function := R_FUNCTION, FC.Frame := rsp, SETUP_STN_TYPE, DSAP <:= SSAP, DLSDU := R_SDU, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
167	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && !U_BUF(DSAP) ⇒ SETUP_IND(0,NO), FC.Function := SC, SETUP_RESM, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
168	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) && !U_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU, SSAP, DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
169	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L FC.Function = SDA_H FC.Function = SDA_L) && !RETRY && !SAP_CHECK(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RS, DLSDU, SSAP, DSAP:=NIL, T_cnt := 0, TRT_ON SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	ACTIVE_I
170	CHECK_T	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA ≠ TS && (DA ≠ 127 (FC.Function ≠ SDN_H && FC.Function ≠ SDN_L)) FC.Frame ≠ req INVALID_FUNCTION) && !(DA = SA && FC.Frame = req && FC.Function = FDL_status && Isochronous_mode) ⇒ RESM := empty, T_cnt := 0	ACTIVE_I
171	CHECK_T	SRC_RECEIVE_ERROR.ind / In_ring_desired ⇒ T_cnt := 0	ACTIVE_I
172	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count > 0 && In_ring_desired ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Dup_add_count := 0, Fault_type := Duplicate_address, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
173	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /DUPLICATE_ADDRESS && Dup_add_count = 0 && In_ring_desired ⇒ Dup_add_count++, TOK_CNT_UPD, Second := 0, RESM := empty, T_cnt := 0, TRT_ON	ACTIVE_I
174	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && Tok_err_cnt > (Limit - 2) && In_ring_desired ⇒ INIT_FCBV_LIST, INIT_LMS, RESM := empty, Fault_type := Out_of_ring, Second := 0, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	LISTEN
175	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && TOKEN_ERROR && Tok_err_cnt ≤ (Limit - 2) && In_ring_desired ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Tok_err_cnt++, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
176	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] ≠ DA && DA ≠ TS && In_ring_desired ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
177	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && LMS[SA] = DA && DA ≠ TS && In_ring_desired ⇒ RESM := empty, TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
178	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry ≠ 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
179	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second = 0 && In_ring_desired ⇒ RESM := empty, Second++, T_cnt := 0, TRT_ON	ACTIVE_I
180	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct=0 && H_list.Num_entry ≠ 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, LMS_UPDATE(DA, SA), Second := 0, Req_h_cnt := H_list.Num_entry, SETUP_HREQ, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	USE_T
181	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) (LMS[SA] ≤ SA && ((SA < TS) (TS < LMS[SA])))) && In_ring_desired ⇒ RESM := empty, TTH_INIT, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, GAP_INIT(0), Fault_Type := In_ring, T_cnt := 0, TRT_ON MAC_BFAULT.ind (Fault_type)	ACTIVE_I
182	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct=0 && H_list.Num_entry = 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A
183	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct=0 && H_list.Num_entry = 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE(DA, SA), TOK_CNT_UPD, LMS_UPDATE, Second := 0, Req_h_cnt := H_list.Num_entry, T_cnt := 0, TRT_ON	CHECK_A
184	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[SA] = DA && Tct > 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE, TOK_CNT_UPD, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL, T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT
185	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && LMS[LMS[SA]] = DA && Second > 0 && Tct > 0 && In_ring_desired ⇒ RESM := empty, GAP_UPDATE, TTH_UPDATE(DA, SA), TOK_CNT_UPD, LMS_UPDATE, Second := 0, Req_h_cnt := H_list.Num_entry, DA,SA:=TS, FC.Function := FDL_status, FC.Frame:=req, DLSDU,SSAP,DSAP:=NIL,T_cnt := 0, TRT_OFF SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	WAIT_TCT

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
186	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!DUPLICATE_ADDRESS && !TOKEN_ERROR && DA = TS && (!LMS[LMS[SA]]=DA && !(LMS[DA] = NIL && ((LMS[SA] > SA && LMS[SA] > TS && SA < TS) (LMS[SA] ≤ SA && ((SA < TS) (TS < LMS[SA])))) && In_ring_desired ⇒ RESM := empty, LMS_UPDATE(DA, SA), TOK_CNT_UPD, Second := 0, T_cnt := 0, TRT_ON	ACTIVE_I
187	CHECK_T	SRC_RECEIVE_TOKEN.ind (DA, SA) /!In_ring_desired ⇒ Fault_Type := Double_token, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	PASSIVE_I
188	CHECK_T	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
189	CHECK_T	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
190	CHECK_T	SRC_SLOT_EVENT.ind /Retry_cnt < 2 ⇒ DS := NS(TS), SA := TS, Retry_cnt++ SRC_SEND_TOKEN.req (DA, SA)	PASS_T
191	CHECK_T	SRC_SLOT_EVENT.ind /Retry_cnt = 2 ⇒ EX_LMS (NS), Retry_cnt := 0, DS := NS(TS), SA := TS SRC_SEND_TOKEN.req (DA, SA)	PASS_T
192	CHECK_T	SRC_SYNI_EVENT.ind / In_ring_desired ⇒ Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
193	AW_STATUS	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
194	AW_STATUS	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(FC.Frame = req (FC.Function ≠ SC && DA ≠ TS)) ⇒ Fault_type := Double_token, NEXT_GAP, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
195	AW_STATUS	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /FC.Frame = rsp && DA = TS && SA=GAP_address && FC.Function = OK && FC.Stn-Type = M_rdy ⇒ DA := SA, SA := TS, LMS_UPDATE(DA, SA), GAP_INIT(NIL), T_cnt := 0 SRC_SEND_TOKEN.req (DA, SA)	PASS_T
196	AW_STATUS	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /FC.Frame = rsp && (DA = TS FC.Function = SC) && (FC.Function ≠ OK FC.Stn-Type ≠ M_rdy) ⇒ NEXT_GAP, T_cnt := 0 SRC_SEND_TOKEN.req (DA, SA)	CHECK_A
197	AW_STATUS	SRC_RECEIVE_ERROR.ind /RECV_ERR_cnt ≥ RECV_ERR_limit ⇒ T_cnt := 0	LISTEN

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
198	AW_STATUS	SRC_RECEIVE_ERROR.ind /RECV_ERR_cnt < RECV_ERR_limit ⇒ NEXT_GAP , T_cnt := 0	CHECK_A
199	AW_STATUS	SRC_RECEIVE_TOKEN.ind (DA, SA) /ln_ring_desired ⇒ Fault_type := Double_token, NEXT_GAP, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	ACTIVE_I
200	AW_STATUS	SRC_SEND_DATA.cnf ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
201	AW_STATUS	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
202	AW_STATUS	SRC_SLOT_EVENT.ind ⇒ NEXT_GAP	CHECK_A
203	AW_STATUS	SRC_SYNI_EVENT.ind /ln_ring_desired ⇒ Fault_type := Not_synchronized, NEXT_GAP MAC_BFAULT.ind (Fault_type)	ACTIVE_I
204	PASSIVE_I	/H_List.Num_entry ≠ 0 ⇒ SETUP_HCON_DS	PASSIVE_I
205	PASSIVE_I	/L_List.Num_entry ≠ 0 ⇒ SETUP_LCON_DS	PASSIVE_I
206	PASSIVE_I	MAC_RESET.req ⇒ RESET_HL_LIST MAC_RESET.cnf	OFFL
207	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = FDL_status ⇒ DLSDU, SSAP, DSAP := NIL, FC.Frame := rsp, FC.Stn-Type := Slave, FC.Function := OK, RESM := empty, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
208	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = Ident ⇒ DA := SA, SA := TS, DSAP <:= SSAP, RESM := empty, IDENT, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
209	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && FC.Function = LSAP_status ⇒ DA := SA, SA := TS, DSAP <:= SSAP, RESM := empty, LSAP_STATUS, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
210	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = 127) && FC.Frame = res && SAP_CHECK(DSAP) && B_BUF(DSAP) ⇒ RESM := empty, SETUP_SIND(0, NO), T_cnt := 0	PASSIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
211	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = 127) && FC.Frame = res && (!SAP_CHECK(DSAP) !B_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
212	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = TE FC.Function = CV) && SAP_CHECK(CS) && I_BUF(CS) ⇒ RESM := empty, SETUP_IND(0, NO), T_cnt := 0, TRT_ON	PASSIVE_I
213	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = CV FC.Function = TE) && (!SAP_CHECK(CS) !I_BUF(CS)) ⇒ RESM := empty, T_cnt := 0, TRT_ON	PASSIVE_I
214	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = SDN_H FC.Function = SDN_L) && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ RESM := empty, SETUP_IND(0, NO), T_cnt := 0	PASSIVE_I
215	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA = TS DA = 127) && FC.Frame = req && (FC.Function = SDN_H FC.Function = SDN_L) && (!SAP_CHECK(DSAP) !I_BUF(DSAP)) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
216	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L FC.Function = SDA_H FC.Function = SRD_L FC.Function = SRD_H FC.Function = SRD_BCT) && RETRY ⇒ DA := RESM.DA, SA := TS, FC := RESM.FC, DSAP := RESM.DSAP, SSAP := RESM.SSAP, DLSDU := RESM.DLSDU, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
217	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_H FC.Function = SDA_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) ⇒ SETUP_IND(0, NO), DA := NIL, SA := NIL, FC.Function := OK, DLSDU, SSAP, DSAP := NIL, SETUP_RESM, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
218	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SDA_L FC.Function = SDA_H) && !RETRY && SAP_CHECK(DSAP) && !I_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU, SSAP, DSAP := NIL, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
219	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SAP_CHECK(DSAP) && I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, if(SAP_List[DSAP].Indication_Mode = ALL Upd_status ≠ NO) SETUP_IND(Ref, Upd_status), DA := R_DA, SA := TS, FC.Function := Upd_status, FC.Frame := rsp, SETUP_STN_TYPE, DSAP ≤ SSAP, DLSDU := R_SDU, SETUP_RESM, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I

Table E.15 (continued)

#	Current state	Event /condition ⇒action	Next state
220	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SÄP_CHECK(DSAP) && !I_BUF(DSAP) && U_BUF(DSAP) ⇒ SETUP_REPLY, DA := R_DA, SA := TS, FC.Function := R_FUNCTION, FC.Frame := rsp, SETUP_STN_TYPE, DSAP <:= SSAP, DLSDU := R_SDU, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
221	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SÄP_CHECK(DSAP) && I_BUF(DSAP) && !U_BUF(DSAP) ⇒ SETUP_IND(0, NO), FC.Function := SC, SETUP_RESM, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
222	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L) && !RETRY && SÄP_CHECK(DSAP) && !I_BUF(DSAP) && !U_BUF(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RR, DLSDU, SSAP, DSAP:=NIL, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
223	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /DA = TS && FC.Frame = req && (FC.Function = SRD_H FC.Function = SRD_BCT FC.Function = SRD_L FC.Function = SDA_H FC.Function = SDA_L) && !RETRY && !SÄP_CHECK(DSAP) ⇒ RESM := empty, DA := SA, SA := TS, FC.Frame := rsp, SETUP_STN_TYPE, FC.Function := RS, DLSDU, SSAP, DSAP:=NIL, T_cnt := 0 SRC_SEND_DATA.req (DA, SA, FC, DSAP, SSAP, DLSDU)	PASSIVE_I
224	PASSIVE_I	SRC_RECEIVE_DATA.ind (DA, SA, FC, DSAP, SSAP, DLSDU) /(DA ≠ TS && (DA≠127 (FC.Function ≠ SDN_H && FC.Function ≠ SDN_L)) (FC.Frame ≠ req && (DA ≠ 127) INVALID_FUNCTION) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
225	PASSIVE_I	SRC_RECEIVE_ERROR.ind ⇒ T_cnt := 0	PASSIVE_I
226	PASSIVE_I	SRC_RECEIVE_TOKEN.ind (DA, SA) ⇒ RESM := empty, T_cnt := 0	PASSIVE_I
227	PASSIVE_I	SRC_SEND_DATA.cnf ⇒	PASSIVE_I
228	PASSIVE_I	SRC_SEND_TOKEN.cnf(Status) ⇒ Fault_type := State_conflict MAC_LFAULT.ind (Fault_type)	OFFL
229	PASSIVE_I	SRC_SLOT_EVENT.ind /!TIME_OUT ⇒ T_cnt ++	PASSIVE_I
230	PASSIVE_I	SRC_SLOT_EVENT.ind /TIME_OUT ⇒ Fault_type := Time_out, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	PASSIVE_I

Table E.15 (*continued*)

#	Current state	Event /condition ⇒action	Next state
231	PASSIVE_I	SRC_SYNI_EVENT.ind ⇒ Fault_type := Not_synchronized, T_cnt := 0 MAC_BFAULT.ind (Fault_type)	PASSIVE_I

E.5.4 Functions

All functions of the MAC are summarized in Table E.16.

Table E.16 – MAC function table

Function name	Operations
IDENT	- Prepares Ident-Response send Ident data, if Ident buffer available send SC, if no Ident buffer
LSAP_STATUS	- Prepares Ident-Response send LSAP data, if LSAP-status buffer available send SC, if no LSAP-status buffer
INIT_FCBV_LIST	FCV[0..126]:= 0, FCB[0..126]:= 1
INIT_LMS	LMS[0..126]:= NIL Tok_err_cnt:=0, Tok_cnt:=255 First :=NIL, LMS_cnt:=0
BUILD_LMS(Ad)	LMS[0..126]:= NIL, LMS[Ad]:=Ad
LMS_UPDATE	if(DA < LMS[SA] ≤ SA SA < DA < LMS[SA] LMS[SA] ≤ SA < DA) LMS[DA] := LMS[SA], LMS[SA] := DA else if (DA = LMS[LMS[SA]]) LMS[LMS[SA]] := NIL, LMS[SA] :=DA else Tok_err_cnt++
EX_LMS (Ad)	LMS[TS]:= LMS[Ad], LMS[Ad]:=NIL
TOK_CNT_UPD	if (Tok_cnt--=0) Tok_cnt:=255, Tok_err_cnt:=0, Dup_add_cnt:=0
DUPLICATE_ADDRESS	SA = TS SA > HSA DA > HSA
TOKEN_ERROR	LMS[SA] := NIL
FCB_UPDATE	FCB[DA]:= not FCB[DA], FCV[DA]:=1
FCV_CLEAR	FCV[DA]:=0, FCB[DA]:=1
SETUP_STN_TYPE	if (In_ring_desired = FALSE) FC.Stn-Type:=Slave else if (LMS[TS] = NIL) FC.Stn-Type:=M_rdy else FC.Stn-Type:=M_in_ring
R_FUNCTION	if (Upd_sts=DL) FC.Function:=RDL else FC.Function:=RDH
SETUP_RESM	RESM.DA := DA RESM.DSAP := DSAP RESM.SSAP := SSAP RESM.FC := FC RESM.DLSDU := DLSDU
RETRY	RESM≠NIL && RESM.SA=SA && DA=TS && FCV[DA]=1 && FC.FCV=1 && FCB[DA]=FC.FCB

Table E.16 (continued)

Function name	Operations
SETUP_HREQ	H_List.Num_entry-- REQM.DA,DA := H_List.First_Entry.DA REQM.DSAP,DSAP := H_List.First_Entry.DSAP REQM.SSAP,SSAP := H_List.First_Entry.SSAP FC := H_List.First_Entry.FC REQM.DLSDU,DLSDU := H_List.First_Entry.DLSDU Cnf := H_List.First_Entry.conf REQM.serv_class := high SA:=TS if (FC.Function≠ FDL_Status, Ident, LSAP_Status, SDN_H, SDN_L) (FC.FCB:=FCB[DA], FC.FCV:=FCV[DA]) else (FC.FCB:=0, FC.FCV:=0) FC.Frame:=req REQM.FC:=FC H_List.Remove()
SETUP_LREQ	L_List.Num_entry-- REQM.DA,DA := L_List.First_Entry.DA REQM.DSAP,DSAP := L_List.First_Entry.DSAP REQM.SSAP,SSAP := L_List.First_Entry.SSAP FC := L_List.First_Entry.FC REQM.DLSDU,DLSDU := L_List.First_Entry.DLSDU Cnf := L_List.First_Entry.conf REQM.serv_class := low SA:=TS if (FC.Function≠ FDL_Status, Ident, LSAP_Status, SDN_H, SDN_L) (FC.FCB:=FCB[DA], FC.FCV:=FCV[DA]) else (FC.FCB:=0, FC.FCV:=0) FC.Frame:=req REQM.FC:=FC L_List.Remove()
SETUP_CONM(Err_type)	C_List.Insert() C_List.Last_Entry.DA := REQM.DA C_List.Last_Entry.DSAP := REQM.DSAP C_List.Last_Entry.SSAP := REQM.SSAP C_List.Last_Entry.serv_class := REQM.serv_class C_List.Last_Entry.FC := Err_type C_List.Last_Entry.DLSDU := NIL C_List.Num_entry++
RESET_HL_LIST	while (H_List.Num_entry ≠ 0) SETUP_HCON_DS while (L_List.Num_entry ≠ 0) SETUP_LCON_DS

Table E.16 (continued)

Function name	Operations
SETUP_HCON_DS	H_List.Num_entry-- C_List.Insert() C_List.Last_Entry.DA := H_List.First_Entry.DA C_List.Last_Entry.DSAP := H_List.First_Entry.DSAP C_List.Last_Entry.SSAP := H_List.First_Entry.SSAP C_List.Last_Entry.serv_class := H_List.First_Entry.serv_class C_List.Last_Entry.FC:= H_List.First_Entry.FC C_List.Last_Entry.Status:= DS C_List.Num_entry++ H_List.Remove()
SETUP_LCON_DS	L_List.Num_entry-- C_List.Insert() C_List.Last_Entry.DA := L_List.First_Entry.DA C_List.Last_Entry.DSAP := L_List.First_Entry.DSAP C_List.Last_Entry.SSAP := L_List.First_Entry.SSAP C_List.Last_Entry.serv_class := L_List.First_Entry.serv_class C_List.Last_Entry.FC:= L_List.First_Entry.FC C_List.Last_Entry.Status:= DS C_List.Num_entry++ L_List.Remove()
SETUP_CON(FC,DLSDU)	C_List.Insert() C_List.Last_Entry.DA := REQM.DA C_List.Last_Entry.DSAP := REQM.DSAP C_List.Last_Entry.SSAP := REQM.SSAP C_List.Last_Entry.serv_class := REQM.serv_class C_List.Last_Entry.FC:= REQM.FC C_List.Last_Entry.DLSDU := DLSDU C_List.Last_Entry.Status:= FC C_List.Num_entry++
SETUP_IND(Ref,Upd_sts)	SAP_List[DSAP].lbuffer.Num_entry-- I_List.Insert() I_List.Last_Entry.DA := DA I_List.Last_Entry.SA := SA I_List.Last_Entry.DSAP := DSAP I_List.Last_Entry.SSAP := SSAP I_List.Last_Entry.FC := FC I_List.Last_Entry.DLSDU := DLSDU I_List.Last_Entry. Status := Upd_sts I_List.Last_Entry.Reference := Ref I_List.Num_entry++ SAP_List[DSAP].lbuffer.Remove()

Table E.16 (continued)

Function name	Operations
SETUP_SIND(Ref,Upd_sts)	SAP_List[DSAP].Sbuffer.Num_entry-- I_List.Insert() I_List.Last_Entry.DA := DA I_List.Last_Entry.SA := SA I_List.Last_Entry.DSAP := DSAP I_List.Last_Entry.SSAP := SSAP I_List.Last_Entry.FC := FC I_List.Last_Entry.DLSDU := DLSDU I_List.Last_Entry.Status := OK I_List.Num_entry++ SAP_List[DSAP].Sbuffer.Remove()
SAP_CHECK (DSAP)	(SA= SAP_List[DSAP].Access SAP_List[DSAP].Access=NIL) && (FC.Function is in SAP_List[DSAP].Function_List_R) && DLSDU.Len ≤ SAP_List[DSAP].LenList[FC.Function]
I_BUF(DSAP)	SAP_List[DSAP].Ibuffer.First_Entry ≠ NIL
B_BUF(DSAP)	SAP_List[DSAP].Sbuffer.First_Entry ≠ NIL
U_BUF(DSAP)	SAP_List[DSAP].Ubuffer.High_buffer ≠ NIL SAP_List[DSAP].Ubuffer.Low_buffer ≠ NIL
SETUP_REPLY	if (FC.Function = SRD_BCT) R_DA=127 else R_DA=SA If (SAP_List[DSAP].Ubuffer.High_buffer.Len ≠ 0) (Upd_sts := DH, Ref := SAP_List[DSAP].Ubuffer.High_reference, R_SDU := SAP_List[DSAP].Ubuffer.High_buffer, if(SAP_List[DSAP].Ubuffer.High_transmit = SINGLE) SAP_List[DSAP]. Ubuffer.High_buffer=NIL) else(Upd_sts := DL, Ref := SAP_List[DSAP].Ubuffer.Low_reference, R_SDU := SAP_List[DSAP].Ubuffer.Low_buffer, if(SAP_List[DSAP].Ubuffer.Low_transmit = SINGLE) SAP_List[DSAP]. Ubuffer.Low_buffer=NIL)
DECODE(func)	if (func = SDN_H) (Service := SDN, Serv_class := High) if (func = SDA_H) (Service := SDA, Serv_class := High) if (func = SRD_H) (Service := SRD, Serv_class := High) if (func = SDN_L) (Service := SDN, Serv_class := Low) if (func = SDA_L) (Service := SDA, Serv_class := Low) if (func = SRD_L) (Service := SRD, Serv_class := Low)
TIME_OUT	(In_ring_desired=true && T_cnt = 2*TS+5) (In_ring_desired=false && T_cnt = 259)
INVALID_FUNCTION	FC.Function = 0 FC.Function = 1 FC.Function = 2 FC.Function = 7 FC.Function = 8 FC.Function = 10 FC.Function = 11
GAP_INIT	Gap_to_do := true GAP_address := (TS + 1) mod (HSA + 1) if (GAP_address = LMS [TS]) GAP_to_do := false, Gud_timer := Tgud

Table E.16 (continued)

Function name	Operations
NEXT_GAP	GAP_lp_cnt := NIL GAP_address := (GAP_address + 1) mod (HSA + 1) if (GAP_address = LMS [TS]) GAP_to_do := false, Gud_timer := Tgud
GAP_UPDATE	Gud_timer := Gud_timer - Ttr, if (GAP_to_do && GAP_lp_cnt = NIL) GAP_lp_cnt := L_list.Num_entry, if (!GAP_to_do && Gud_timer < 0) (GAP_address := (TS + 1) mod (HSA + 1) if (GAP_address = LMS [TS]) GAP_to_do := false, Gud_timer := Tgud
TTH_INIT	Trr := Ttr, TRT.start(Ttr)
TTH_UPDATE	Trr := Ttr - TRT.cv, TRT.start(Ttr)
TTH_AVAILABLE	Trr < TRT.cv
TRT_ON	if (TRT.stopped) TRT.start(Trr)
TRT_OFF	TRT.stop, Trr := TRT.cv, TRT.stopped := TRUE

E.6 SRU

E.6.1 Overview

SRU contains the following parts:

- Message oriented Main-SM (SRC)
- Character Receive SM (CRX)
- Timer-SM (TIM)
- Character Send SM (CTX)

Machine internal communication is done without any time delay.

The CTX and CRX machines are used only in asynchronous mode to transfer a bit oriented stream in a character oriented stream. This task can be done with standard components (called Universal Asynchronous Receiver / Transmitter) and are not described further. The CRX (together with the physical receive units) shall allow a distortion of the input signal of +/- 10%.

The timer module is also a standard component that just implements the required timers. The Wait-service selects the specific idle-timers. The reversed timer is started with StartIdle and Start. The other timer services are used to indicate the expiration of timers.

The SRC is responsible for coding/encoding of DLPDUs. This module is responsible for the transmission procedures as well (see clause 21 for further description for send receive procedures).

Both the asynchronous and the synchronous interfaces are shown in Figure E.2.

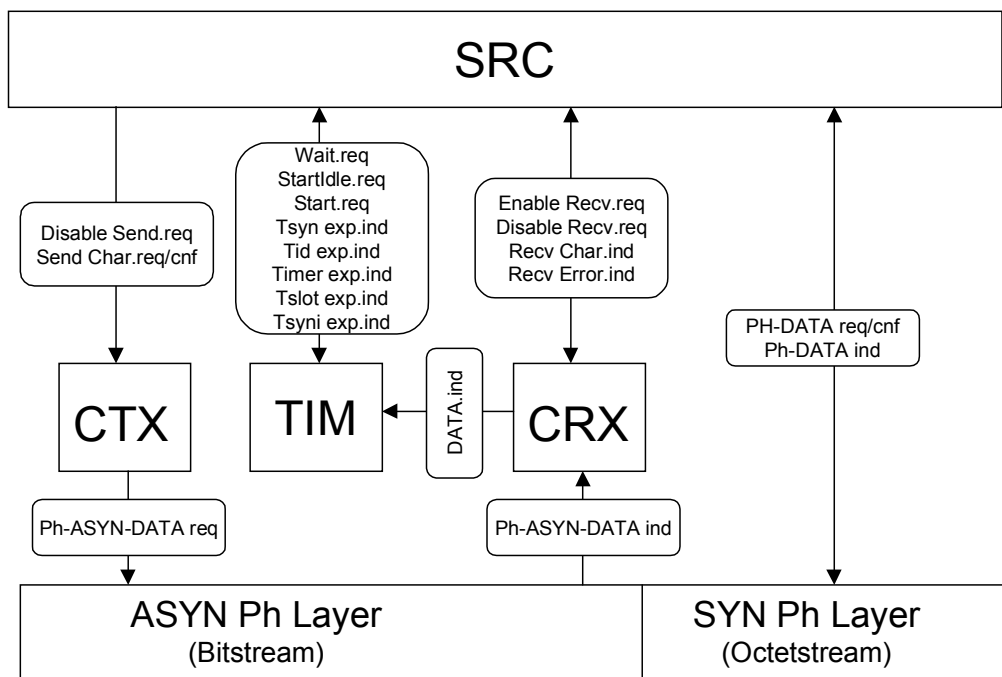


Figure E.2 – Structure of the SRU Machine

E.6.2 Character send SM (CTX)

This state machine implements an UART which has the ability to send without gaps between stop-bit and preceding start-bit. In conjunction with the underlying physical layer it produces a maximum of less than 0,3%. For data rates of 1500 kbit/s and higher the maximum clock drift of $\pm 0,03$ % should not be exceeded. This model describes a SRC which gets an indication after a complete character has been send (this model implies a zero delay between SRC and CTX). Real devices may implement a double buffering which has influence on the termination of a send sequence (a sender empty indication is required).

Interface to SRC

- Send_Char.req/.cnf
- Disable_Send.req

E.6.3 Character receive SM (CRX)

This state machine implements an UART which has the ability to receive without gaps between stop-bit and preceding start-bit. It has to detect stop-bit and parity errors. Together with the underlying physical layer it has to detect at least signals with 10% distortion correctly. The change between Stop and Start has a tolerance of $\pm 40\%$. If there is a signal change detectible by the underlying physical layer within 10% and 90% of the signal, the resulting bit-value is undefined.

When a change to start signal without an valid character is detected with enabled receiver the CRX has to report an error to the SRC. It has be at least as sensitive as the Timer-SM (a trigger for the timer must trigger the receiver as well).

Interface to SRC:

- Receive_Char.ind
- Enable_Recv.req
- Disable_Recv.req

- Receive_Error.ind

E.6.4 Timer-SM (TIM)

The timer module implements three different timers.

One Timer is used for one shot events and will be started by the SRC with a start.req primitive.

Two Timers start when the receiver is disabled or the bus changes from active (logical 0) to inactive (logical 1).

Timer for Tsyn/Tid

This timer is set to 0 when receiver is enabled and the signal is ZERO. If a synchronous Ph Layer is used the timer is loaded with StartIdle.req.

Two thresholds will produce indications:

- The threshold Tsyn is a fixed value
- The threshold Tid will be loaded according to Tid1/Tid2 with the wait.req primitive

Timer for Tsyni/Tslot

This timer is reloaded with Tslot -Tsyn when Tsyn-timer expires

This timer is reloaded with Tslot when this timer expires with Tslot

This timer is reloaded with Tsyn, when the receiver becomes active

Interface to SRC:

- Tsyn_exp.ind
- Tid_exp.ind
- Tslot_exp.ind
- Tsyni_exp.ind
- Timer_exp.ind
- Wait.req
- Start.req
- StartIdle.req

CTX, CRX and TIM are not described as state machines.

E.6.5 Primitive definition of SRC

E.6.5.1 Primitives exchanged between DLM and SRC

This interface is not worked out further. The primitives issued by the DLM to the SRC are shown in Table E.17.

Table E.17 – Primitives issued by DLM to SRC

Primitive name	Associated parameters
SRC_RESET.req	(none)

Primitives Exchanged between SRC and DLM

This interface is not described in detail. The primitives issued by the SRC to the DLM are shown in Table E.18.

Table E.18 – Primitives issued by SRC to DLM

Primitive name	Associated parameters
SRC_RESET.cnf	(none)
MAC_LFAULT.ind	Fault_type
MAC_BFAULT.ind	Fault_type

E.6.5.2 Primitives exchanged between MAC and SRC

The primitives issued by the MAC to the SRC are shown in Table E.19.

Table E.19 – Primitives issued by MAC to SRC

Primitive name	Associated parameters
SRC_SEND_TOKEN.req	DA, SA
SRC_SEND_DATA.req	DA, SA, FC, DSAP, SSAP, DLSDU

The primitives issued by the SRC to the MAC are shown in Table E.20.

Table E.20 – Primitives issued by SRC to MAC

Primitive name	Associated parameters
SRC_SEND_TOKEN.cnf	Status
SRC_SEND_DATA.cnf	(none)
SRC_RECEIVE_DATA.ind	DA, SA, FC, DSAP, SSAP, DLSDU
SRC_RECEIVE_TOKEN.ind	DA, SA
SRC_RECEIVE_ERROR.ind	(none)
SRC_SLOT_EVENT.ind	(none)
SRC_SYNI_EVENT.ind	(none)

E.6.5.3 Parameters of SRC Primitives

All parameters used with primitives exchanged between the MAC and the SRC are shown in Table E.21.

Table E.21 – Parameters used with primitives exchanged between MAC and SRC

Parameter name	Description
DA	Station address of the receiving unit
SA	Station address of the sending unit
FC	The FC structure is described in Table E.22
DSAP	Identifier of a remote Service Access Point
SSAP	Identifier of the local Service Access Point
DLSDU	Data unit of a DLS-user
Status	Status of the service execution

Table E.22 – FC structure

FC.frame	FC.function	FCB,FCV	FC.stn-type
req	SDN_H,SDN_L, SDA_H, SDA_L, SRD_H, SRD_L, TE, CV, FDL_Status, LSAP_Status, Ident	0,0: all services 0,1: SYN for SRD,SDA 1,0: normal Op. SDA,SRD 1,1: normal Op. SDA,SRD	
rsp	OK, RS, RR, DL, DH, NR, RDL, RDH, SC		Slave, M_n_rdy, M_rdy, M_in_ring

E.6.6 State machine description

The Send Receive Control forms messages out of a stream of data offered by the Physical Layer and vice versa. The SRC supports both Synchronous and Asynchronous Physical layers.

The local variables of the SRC are shown in Table E.23.

Local Variables

Table E.23 – Local variables of SRC

Name	Type	Range	Remark
SDNM	Bool		SDN Marker = SDN (was) executed
SType	Enum	(I/R)	Send type (Initiator/Response)
DAE	U8	0,0x80	Address Extension
SAE	U8	0,0x80	Address Extension
F	U8		FC-Field
Fcs	U8		Checksum
Char	U8		
l	U8		
j	U8		
TOK	Bool		Marker Token was sent
RxSD	Bool		Marker SD was received
SDdata	U8		Storage of SD
C1,C2	U8		FCS components

E.6.7 SRC state table

The state table of the SRC is shown in Table E.24.

Table E.24 – SRC state table

#	Current state	Event /condition ⇒action	Next state
1	any-state	SRC_RESET.req /Data_rate = NIL ⇒ SRC_RESET.cnf	OFFL
2	any-state	SRC_RESET.req /PHSYN && Data_rate ≠ NIL ⇒ SRC_RESET.cnf	Idle
3	any-state	SRC_RESET.req /PHSYN && Data_rate ≠ NIL ⇒ SRC_RESET.cnf, StartIdle.req	AW_SD
4	Idle	Tsyn_exp.ind ⇒ Enable_Recv.req	AW_SD
5	Idle	Tid_exp.ind ⇒	Idle
6	Idle	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind	Idle
7	Idle	SRC_SEND_TOKEN.req (DA,SA) ⇒ SDNM := FALSE Wait.req (Tid1)	SND_SD4

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
8	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = FALSE ⇒ SET_SDNM(FC.Function) Wait.req (Tid1)	ISND_SD1
9	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 DSAP ≠ NIL SSAP ≠ NIL) && FC.Frame = req && SDNM = FALSE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Wait.req (Tid1)	ISND_SD2
10	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = TRUE ⇒ SET_SDNM(FC.Function) Wait.req (Tid2)	ISND_SD1
11	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 DSAP ≠ NIL SSAP ≠ NIL) && FC.Frame = req && SDNM = TRUE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Wait.req (Tid2)	ISND_SD2
12	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = rsp && FC.Function ≠ SC ⇒ SDNM := FALSE	RSND_SD1
13	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 DSAP ≠ NIL SSAP ≠ NIL) && FC.Frame = rsp ⇒ SET_DAE, SET_SAE, SDNM := FALSE, j:=(SAE+DAE)/0x80+3	RSND_SD2
14	Idle	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /FC.Frame = rsp && FC.Function = SC ⇒ SDNM := FALSE	SND_SC
15	AW_SD	SRC_SEND_TOKEN.req (DA,SA) ⇒ SDNM := FALSE Wait.req (Tid1)	SND_SD4
16	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = FALSE ⇒ SET_SDNM(FC.Function) Disable_Recv.req Wait.req (Tid1)	ISND_SD1
17	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /(DLSDU.Len ≠ 0 DSAP ≠ NIL SSAP ≠ NIL) && FC.Frame = req && SDNM = FALSE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Disable_Recv.req Wait.req (Tid1)	ISND_SD2

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
18	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = req && SDNM = TRUE ⇒ SET_SDNM(FC.Function) Disable_Recv.req Wait.req (Tid2)	ISND_SD1
19	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len ≠ 0 DSAP ≠ NIL SSAP ≠ NIL && FC.Frame = req && SDNM = TRUE ⇒ SET_DAE, SET_SAE, SET_SDNM(FC.Function), j:=(SAE+DAE)/0x80+3 Disable_Recv.req Wait.req (Tid2)	ISND_SD2
20	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len = 0 && DSAP = NIL && SSAP = NIL && FC.Frame = rsp && FC.Function ≠ SC ⇒ SDNM := FALSE Disable_Recv.req	RSND_SD1
21	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /DLSDU.Len ≠ 0 DSAP ≠ NIL SSAP ≠ NIL && FC.Frame = rsp ⇒ SET_DAE, SET_SAE, SDNM := FALSE, j:=(SAE+DAE)/0x80+3 Disable_Recv.req	RSND_SD2
22	AW_SD	SRC_SEND_DATA.req (DA,SA,FC,DSAP,SSAP,DLSDU) /FC.Frame = rsp && FC.Function = SC ⇒ SDNM := FALSE Disable_Recv.req	SND_SC
23	AW_SD	RX_DATA(data) /data = SD1 ⇒ SD_count++, DLSDU.Len := 0, CHECKINI(data), i:=0	AW_DA
24	AW_SD	RX_DATA(data) /data = SD2 ⇒ SD_count++, CHECKINI(data)	AW_LE
25	AW_SD	RX_DATA(data) /data = SD3 ⇒ SD_count++, DLSDU.Len := 8, CHECKINI(data)	AW_DA
26	AW_SD	RX_DATA(data) /!PHSYN && data = SC ⇒ SD_count++, FC.Function := SC, FC.Frame := rsp, i:=0 SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Disable_Recv.req	Idle
27	AW_SD	RX_DATA(data) /PHSYN && data = SC ⇒ SD_count++, FC.Function := SC, FC.Frame := rsp, i:=0, TOK := FALSE	AW_CRC1
28	AW_SD	RX_DATA(data) /data = SD4 ⇒ SD_count++	AW_DAT
29	AW_SD	RX_DATA(data) /data ≠ SC, SD1, SD2, SD3, SD4 ⇒ SD_error_count++ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
30	AW_SD	Recv_Error.ind ⇒ SD_error_count++ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
31	AW_SD	Tslot_exp.ind ⇒ SRC_SLOT_EVENT.ind	AW_SD
32	AW_SD	Tsyn_exp.ind /!PHSYN ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req, Enable_Recv.req	AW_SD
33	AW_SD	Tsyn_exp.ind /PHSYN ⇒	AW_SD
34	AW_SD	Ph-DATA.ind(SOD,data) ⇒	AW_SD
35	AW_LE	RX_DATA(data) /DLSDU.Len > 3 && DLSDU.Len < 250 ⇒ DLSDU.Len := (data - 3), i:=0	AW_LER
36	AW_LE	RX_DATA(data) /DLSDU.Len < 4 DLSDU.Len > 249 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
37	AW_LER	RX_DATA(data) /data = DLSDU.Len + 3 ⇒	AW_SDR
38	AW_LER	RX_DATA(data) /data ≠ (DLSDU.Len + 3) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
39	AW_SDR	RX_DATA(data) /data = SD2 ⇒	AW_DA
40	AW_SDR	RX_DATA(data) /data ≠ SD2 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
41	AW_DA	RX_DATA(data) ⇒ DA := data AND 0x7f, DAE := data AND 0x80, CHECKS(data), DSAP := NIL	AW_SA
42	AW_SA	RX_DATA(data) ⇒ SA := data AND 0x7f, SAE := data AND 0x80, CHECKS(data), SSAP := NIL	AW_FC
43	AW_FC	RX_DATA(data) /DLSDU.Len > 0 && DAE ≠ 0 ⇒ CHECKS(data), SETUP_FC	AW_DSAP
44	AW_FC	RX_DATA(data) /DLSDU.Len > 0 && DAE = 0 && SAE ≠ 0 ⇒ CHECKS(data), SETUP_FC	AW_SSAP

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
45	AW_FC	RX_DATA(data) /DLSDU.Len > 0 && DAE = 0 && SAE = 0 ⇒ CHECKS(data), SETUP_FC	AW_DATA
46	AW_FC	RX_DATA(data) /!PHSYN && DLSDU.Len = 0 && DAE = 0 && SAE = 0 ⇒ CHECKS(data), SETUP_FC	AW_FCS
47	AW_FC	RX_DATA(data) /PHSYN && DLSDU.Len = 0 && DAE = 0 && SAE = 0 ⇒ CHECKS(data), SETUP_FC	AW_CRC1
48	AW_FC	RX_DATA(data) /DLSDU.Len = 0 && (DAE ≠ 0 SAE ≠ 0) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
49	AW_DSAP	RX_DATA(data) /data && 0xc0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
50	AW_DSAP	RX_DATA(data) /DLSDU.Len > 1 && SAE ≠ 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_SSAP
51	AW_DSAP	RX_DATA(data) /DLSDU.Len > 1 && SAE = 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_DATA
52	AW_DSAP	RX_DATA(data) /!PHSYN && DLSDU.Len = 1 && SAE = 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_FCS
53	AW_DSAP	RX_DATA(data) /PHSYN && DLSDU.Len = 1 && SAE = 0 ⇒ DSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_CRC1
54	AW_DSAP	RX_DATA(data) /DLSDU.Len = 1 && SAE ≠ 0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
55	AW_SSAP	RX_DATA(data) /data && 0xc0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
56	AW_SSAP	RX_DATA(data) /DLSDU.Len > 1 ⇒ SSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_DATA
57	AW_SSAP	RX_DATA(data) /!PHSYN && DLSDU.Len = 1 ⇒ SSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_FCS
58	AW_SSAP	RX_DATA(data) /PHSYN && DLSDU.Len = 1 ⇒ SSAP:=data, DLSDU.Len := DLSDU.Len -1, CHECKS(data)	AW_CRC1

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
59	AW_DATA	RX_DATA(data) /DLSDU.Len > i + 1 ⇒ DLSDU.Data[i] := data, i := i + 1, CHECKS(data)	AW_DATA
60	AW_DATA	RX_DATA(data) /!PHSYN && DLSDU.Len = i + 1 ⇒ DLSDU.Data[i] := data, CHECKS(data)	AW_FCS
61	AW_DATA	RX_DATA(data) /PHSYN && DLSDU.Len = i + 1 ⇒ DLSDU.Data[i] := data, CHECKS(data), TOK := FALSE	AW_CRC1
62	AW_FCS	RX_DATA(data) /data = Fcs mod 256 ⇒	AW_ED
63	AW_FCS	RX_DATA(data) /data ≠ (Fcs mod 256) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
64	AW_ED	RX_DATA(data) /data = ED && DA ≠ 0x7f ⇒ SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Start.req (Tsdr), Disable_Recv.req	Idle
65	AW_ED	RX_DATA(data) /data = ED && DA = 0x7f ⇒ SDNM := TRUE SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Start.req (Tsdr), Disable_Recv.req	Idle
66	AW_ED	RX_DATA(data) /data ≠ ED ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
67	AW_CRC1	RX_DATA(data) /data = C1 ⇒	AW_CRC2
68	AW_CRC1	RX_DATA(data) /data ≠ C1 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	AW_EODAF
69	AW_CRC2	RX_DATA(data) /data = C2 ⇒	AW_EODA
70	AW_CRC2	RX_DATA(data) /data ≠ C2 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	AW_EODAF
71	AW_EODA	RX_DATA(data) ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	AW_EODAF
72	AW_EODA	Ph-DATA.ind(EODA,data) /!TOK && DA ≠ 0x7f ⇒ SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU), Start.req (Tsdr), StartIdle.req	AW_SD

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
73	AW_EODA	Ph-DATA.ind(EODA,data) /TOK && DA ≠ 0x7f ⇒ SRC_RECEIVE_TOKEN.ind (DA, SA), StartIdle.req	AW_SD
74	AW_EODA	Ph-DATA.ind(EODA,data) /DA = 0x7f ⇒ SDNM := TRUE SRC_RECEIVE_DATA.ind (DA,SA,FC,DSAP,SSAP,DLSDU),StartIdle.req	AW_SD
75	AW_EODAF	Ph-DATA.ind(DATA,data) ⇒	AW_EODAF
76	AW_EODAF	Ph-DATA.ind(EODA,data) ⇒ StartIdle.req	AW_SD
77	AW_DAT	RX_DATA(data) /(data && 0x80) = 0 ⇒	AW_SAT
78	AW_DAT	RX_DATA(data) /(data && 0x80) ≠ 0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
79	AW_SAT	RX_DATA(data) /PHSYN && (data && 0x80) = 0 ⇒ SRC_RECEIVE_TOKEN.ind (DA, SA), Disable_Recv.req	Idle
80	AW_SAT	RX_DATA(data) /PHSYN && (data && 0x80) = 0 ⇒ TOK := TRUE	AW_CRC1
81	AW_SAT	RX_DATA(data) /(data && 0x80) ≠ 0 ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
82	SND_SD4	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind Disable_Recv.req	Idle
83	SND_SD4	Tsyn_exp.ind ⇒	SND_SD4
84	SND_SD4	Timer_exp.ind ⇒	SND_SD4
85	SND_SD4	Tid_exp.ind /PHSYN ⇒ data:=SD4 Enable_Recv.req, TX_DATA(data)	SND_DAT
86	SND_SD4	Tid_exp.ind /PHSYN ⇒ CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAT
87	SND_SOAT	Ph-DATA.cnf ⇒ data := SD4 Ph-DATA.req(DATA,data)	SND_DAT

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
88	SND_DAT	TX_DATA_CNF ⇒ CHECKS(DA), data:=DA TX_DATA(data)	SND_SAT
89	SND_SAT	TX_DATA_CNF /!PHSYN ⇒ CHECKS(DA), data:=SA TX_DATA(data)	SND_TOK
90	SND_SAT	TX_DATA_CNF /PHSYN ⇒ CHECKS(DA), data:=SA TX_DATA(data)	ST_SY_C1
91	SND_TOK	TX_DATA_CNF /!RxSD ⇒ Status := no_token_pass SRC_SEND_TOKEN.cnf(Status) Disable_Recv.req, Disable_Send.req Start.req(Tqui)	AW QUI
92	SND_TOK	TX_DATA_CNF /RxSD && SDdata≠SD4 ⇒ Status := token_pass_failed SRC_SEND_TOKEN.cnf(Status) Disable_Recv.req, Disable_Send.req Start.req(Tqui)	AW QUI
93	SND_TOK	TX_DATA_CNF /RxSD && SDdata=SD4 ⇒ Status := token_pass_ok SRC_SEND_TOKEN.cnf(Status) Disable_Recv.req, Disable_Send.req Start.req(Tqui)	AW QUI
94	ST_SY_C1	Ph-DATA.cnf ⇒ data := Crc1 Ph-DATA.req(DATA,data)	ST_SY_C2
95	ST_SY_C2	Ph-DATA.cnf ⇒ data := Crc2 Ph-DATA.req(DATA,data)	ST_SY_EO
96	ST_SY_EO	Ph-DATA.cnf ⇒ Ph-DATA.req(EODA,data)	ST_SY_END
97	ST_SY_END	Ph-DATA.cnf /PHSYN && !RxSD ⇒ Status := no_token_pass SRC_SEND_TOKEN.cnf(Status), Start.req(Tqui)	AW QUI
98	ST_SY_END	Ph-DATA.cnf /PHSYN && RxSD && SDdata≠SD4 ⇒ Status := token_pass_failed SRC_SEND_TOKEN.cnf(Status), Start.req(Tqui)	AW QUI

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
99	ST_SY_END	Ph-DATA.cnf /PHSYN && RxSD && SDdata=SD4 ⇒ Status := token_pass_ok SRC_SEND_TOKEN.cnf(Status), Start.req(Tqui)	AW_QUI
100	ISND_SD1	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind Disable_Recv.req	Idle
101	ISND_SD1	Tsyn_exp.ind ⇒	ISND_SD1
102	ISND_SD1	Timer_exp.ind ⇒	ISND_SD1
103	ISND_SD1	Tid_exp.ind /SA ≠ DA FC.Function ≠ 9 ⇒ SType:=I data:=SD1 CHECKINI(data) TX_DATA(data)	SND_DAD
104	ISND_SD1	Tid_exp.ind /!PHSYN && SA = DA && FC.Function = 9 ⇒ SType:=R data:=SD1 CHECKINI(data) TX_DATA(data)	SND_DAD
105	ISND_SD1	Tid_exp.ind /SA ≠ DA FC.Function ≠ 9 ⇒ SType:=I CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAD
106	ISND_SD1	Tid_exp.ind /SA = DA && FC.Function = 9 ⇒ SType:=R CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAD
107	ISND_SD2	Tsyni_exp.ind ⇒ SRC_SYNI_EVENT.ind Disable_Recv.req	Idle
108	ISND_SD2	Tsyn_exp.ind ⇒	ISND_SD2
109	ISND_SD2	Timer_exp.ind ⇒	ISND_SD2
110	ISND_SD2	Tid_exp.ind /!PHSYN ⇒ SType:=I data:=SD2 CHECKINI(data) TX_DATA(data)	SND_LE
111	ISND_SD2	Tid_exp.ind /PHSYN ⇒ SType:=I CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAL

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
112	RSND_SD1	Tsyn_exp.ind ⇒	RSND_SD1
113	RSND_SD1	Timer_exp.ind /!PHSYN ⇒ SType:=R data:=SD1 CHECKINI(data) TX_DATA(data)	SND_DAD
114	RSND_SD1	Timer_exp.ind /PHSYN ⇒ SType:=R CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAD
115	RSND_SD2	Tsyn_exp.ind ⇒	RSND_SD2
116	RSND_SD2	Timer_exp.ind /!PHSYN ⇒ SType:=R data:=SD2 CHECKINI(data) TX_DATA(data)	SND_LE
117	RSND_SD2	Timer_exp.ind /PHSYN ⇒ SType:=R CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAL
118	SND_SC	Timer_exp.ind /!PHSYN ⇒ SType:=R, data:=SC, CHECKINI(data) TX_DATA(data)	SND_ENDD
119	SND_SC	Timer_exp.ind /PHSYN ⇒ SType:=R, CHECKINI(data) Ph-DATA.req(SOA,data)	SND_SOAS
120	SND_SOAL	Ph-DATA.cnf ⇒ data := SD2 Ph-DATA.req(DATA,data)	SND_LE
121	SND_LE	TX_DATA_CNF ⇒ data := DLSDU.Len + j, CHECKSYN(data) TX_DATA(data)	SND_LER
122	SND_LER	TX_DATA_CNF ⇒ CHECKSYN(data) TX_DATA(data)	SND_SD2R
123	SND_SD2R	TX_DATA_CNF ⇒ data := SD2, CHECKSYN(SD2) TX_DATA(data)	SND_DAD

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
124	SND_SOAD	Ph-DATA.cnf ⇒ data := SD1 Ph-DATA.req(DATA,data)	SND_DAD
125	SND_DAD	TX_DATA_CNF ⇒ data := DA + DAE, CHECKS(data) TX_DATA(data)	SND_SAD
126	SND_SAD	TX_DATA_CNF ⇒ data := SA + SAE, CHECKS(data) TX_DATA(data)	SND_FC
127	SND_FC	TX_DATA_CNF /DAE ≠ 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_DSAP
128	SND_FC	TX_DATA_CNF /DAE = 0 && SAE ≠ 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_SSAP
129	SND_FC	TX_DATA_CNF /DLSDU.Len > 0 && DAE = 0 && SAE = 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_DATA
130	SND_FC	TX_DATA_CNF /DLSDU.Len = 0 && DAE = 0 && SAE = 0 ⇒ SET_F, data:= F, CHECKS(data) TX_DATA(data)	SND_FCS
131	SND_DSAP	TX_DATA_CNF /SAE ≠ 0 ⇒ data := DSAP, CHECKS(data) TX_DATA(data)	SND_SSAP
132	SND_DSAP	TX_DATA_CNF /DLSDU.Len>0 && SAE = 0 ⇒ data := DSAP, CHECKS(data) TX_DATA(data)	SND_DATA
133	SND_DSAP	TX_DATA_CNF /DLSDU.Len = 0 && SAE = 0 ⇒ data := DSAP, CHECKS(data) TX_DATA(data)	SND_FCS
134	SND_SSAP	TX_DATA_CNF /DLSDU.Len > 0 ⇒ data := SSAP, CHECKS(data) TX_DATA(data)	SND_DATA
135	SND_SSAP	TX_DATA_CNF /DLSDU.Len = 0 ⇒ data := SSAP, CHECKS(data) TX_DATA(data)	SND_FCS

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
136	SND_DATA	TX_DATA_CNF /DLSDU.Len > i ⇒ data:=DLSDU.Data[i], i := i + 1, CHECKS(data) TX_DATA(data)	SND_DATA
137	SND_DATA	TX_DATA_CNF /!PHSYN && DLSDU.Len = i ⇒ data:=DLSDU.Data[i], CHECKS(data) TX_DATA(data)	SND_FCS
138	SND_DATA	TX_DATA_CNF /PHSYN && DLSDU.Len = i ⇒ data:=DLSDU.Data[i], CHECKS(data) Ph-DATA.req(DATA,data)	SND_SY_C1
139	SND_FCS	TX_DATA_CNF ⇒ data := (Fcs mod 256) TX_DATA(data)	SND_ED
140	SND_ED	TX_DATA_CNF ⇒ data := ED TX_DATA(data)	SND_ENDD
141	SND_ENDD	TX_DATA_CNF ⇒ SRC_SEND_DATA.cnf, Disable_Send.req, Start.req(Tqui)	AW QUI
142	SND_SOAS	Ph-DATA.cnf ⇒ data := SC Ph-DATA.req(DATA,data)	SD_SY_C1
143	SD_SY_C1	Ph-DATA.cnf ⇒ data := Crc1 Ph-DATA.req(DATA,data)	SD_SY_C2
144	SD_SY_C2	Ph-DATA.cnf ⇒ data := Crc2 Ph-DATA.req(DATA,data)	SD_SY_EO
145	SD_SY_EO	Ph-DATA.cnf ⇒ Ph-DATA.req(EOA,data)	SD_SY_END
146	SD_SY_END	Ph-DATA.cnf ⇒ SRC_SEND_DATA.cnf, Start.req(Tqui)	AW QUI
147	AW QUI	Timer_exp.ind /PHSYN ⇒ StartIdle.req	AW_SD
148	AW QUI	Timer_exp.ind /!PHSYN && SType=I && !SDNM ⇒ Enable_Recv.req	AW_SD
149	AW QUI	Timer_exp.ind /!PHSYN && SType=R SDNM ⇒	idle

Table E.24 (continued)

#	Current state	Event /condition ⇒action	Next state
150	S-State	RX_DATA /!RxSD ⇒ SDdata := data, RxSD :=TRUE	S-State
151	S-State	RX_DATA /RxSD ⇒	S-State
152	S-State	Ph-DATA.ind(EOA,Data) ⇒	S-State
153	S-State	Ph-DATA.ind(EOAD,Data) ⇒	S-State
154	AW-State	Recv_Error.ind ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req	Idle
155	AW-State	Tsyn_exp.ind ⇒ SRC_RECEIVE_ERROR.ind, Disable_Recv.req, Enable_Recv.req	AW_SD
156	AW-State	Ph-DATA.ind(EOA,Data) ⇒ SRC_RECEIVE_ERROR.ind, StartIdle.req	AW_SD

E.6.8 Functions

All function of the SRC are shown in Table E.25.

Table E.25 – SRC functions

Function	Description
PHSYN	Data_rate = 31,25 kbit/s
CHECKINI(data)	if (PHSYN) then (Init CRC(C1,C2)) else (Fcs := 0)
CHECKS(data)	if (PHSYN) then (Update CRC(C1,C2)) else (Fcs := 0)
CHECKSYN(data)	Update CRC(C1,C2)
RX_DATA(data)	if (PHSYN) then (Ph-DATA.ind(DATA,data)) else (Recv_Char.ind(data))
TX_DATA(data)	if (PHSYN) then (Ph-DATA.req(DATA,data)) else (Send_Char.req(data))
TX_DATA_CNF	if (PHSYN) then (Ph-DATA.cnf) else (Send_Char.cnf)
SET_SDNM	(if (FC.Function = SDN_L, SDN_H,TE,CV) SDNM = TRUE else SDNM = FALSE)
SETUP_FC	(FC.Function = Char AND 0x0f, if (Char AND 0x40) (FC.FCB = Char AND 0x20, FC.FCV = Char AND 0x10, FC.Frame_type =req) else (FC.Stn-Type= Char AND 0x30, FC.Frame_type =rsp))
SET_F	(F = FC.Function, if (FC.Frame_type =req) (0x80 + FC.FCB * 0x20 + FC.FCV * 0x10) else (F = F + FC.Stn-Type* 0x30))

Annex F (informative)

–

Type 3: Exemplary token procedure and message transfer periods

F.1 Procedure of token passing

For explanatory purposes, the description of the token rotation time in 20.3.2.6 and message priorities in 20.3.2.7 are amended. The timers are presented as functions of the time t .

The token rotation time is measured by using the token-rotation-timer. At the beginning (t_{i-1} in Figure F.1) the token-rotation-timer is loaded with the target rotation time T_{TR} and decrement each bit time.

After receiving a token, the DL-entity may carry out high priority and low priority message transfer periods, according to the following rules.

- On reception of the token (t_i) the T_{RR} (real rotation time for the last token cycle) is derived from the token-rotation-timer by reading its current value that represents T_{TH} and calculating $T_{RR}(i) = T_{TR} - T_{TH}$. The timer is loaded with the value T_{TR} and restarted.
- **One** high priority message transfer period is always possible at this moment (just after token reception).
- A **further** high priority or low priority message transfer period or generally a low priority message transfer period may only be carried out if time to hold the token is still available at the instant of execution, that means that the actual value of T_{RR} of the last, just ending token cycle (from the viewpoint of this station) is less than the current value of the token-rotation-timer.

In order to avoid unnecessary timers in an implementation, the increase of T_{RR} during token usage is measured indirectly by use of the token-rotation-timer, which is measuring the token hold time of the just started actual token cycle. As known from the timer concepts of IEC 61131-3, one timer can be used for measurement of parallel but shifted times and therefore the availability of the token holding time is derived from the token-rotation-timer. Thus, a message transfer period can be executed, if at the instant of execution, the read value of the token-rotation-timer is greater than T_{RR} , as shown in Figure F.1.

Figure F.1 shows the relationship of T_{TR} , T_{RR} , T_{TH} .

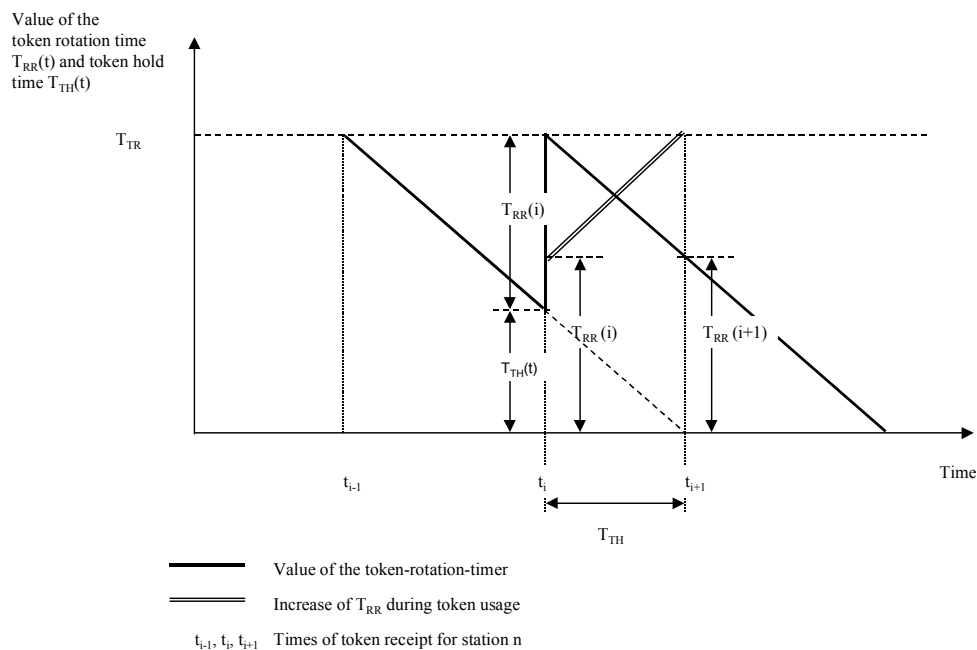


Figure F.1 – Derivation of the token holding time (T_{TH})

F.2 Examples for token passing procedure

The consumption of available token holding time depends of the actual working load. In the following examples the different working load situations are demonstrated together with the usage of the token holding time.

The figures illustrate the behavior of the token passing in cases of varying load. The used time scale only demonstrates the relationship without any relation to real-time. Below are listed some enumerated cases to explain the Token Passing procedure.

Case one: A startup situation is depicted in Figure F.2. The token is passed by the token holder without usage of the available token holding time (T_{TH}). During the start up phase no messages are sent. As depicted in Figure F.2, the timer value of each token holder is always greater than zero in case of token receipt, for example, the read value from token-rotation-timer is greater than zero. Thus T_{TH} is still available.

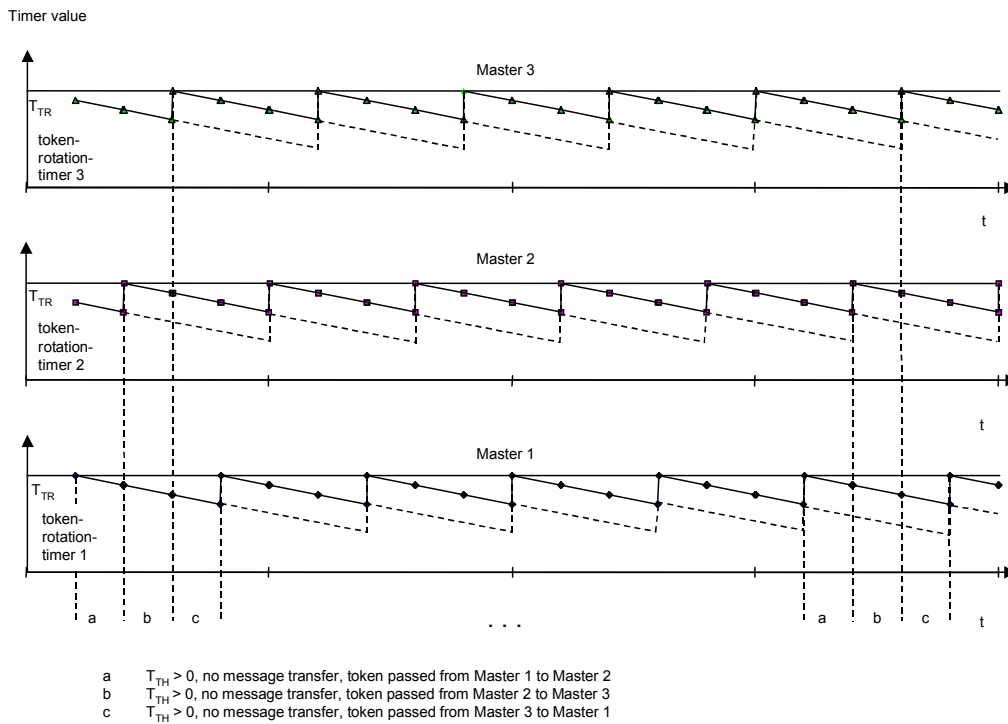


Figure F.2 – No usage of token holding time (T_{TH})

Case two: The bus transfer time is shared fairly between all token holders (see Figure F.3). Each token holder only uses a part of the token holding time (T_{TH}). The actual token holder only sends one message in each token cycle. After that it passes the token to the next token holder. The T_{TH} used by a token holder has a constant (for that token holder) value (see Figure F.3: a, b, c):

As depicted in Figure F.3 some T_{TH} time is still available in each station at the time of token receipt because the read value of the token-rotation-timer is greater than zero at that instant.

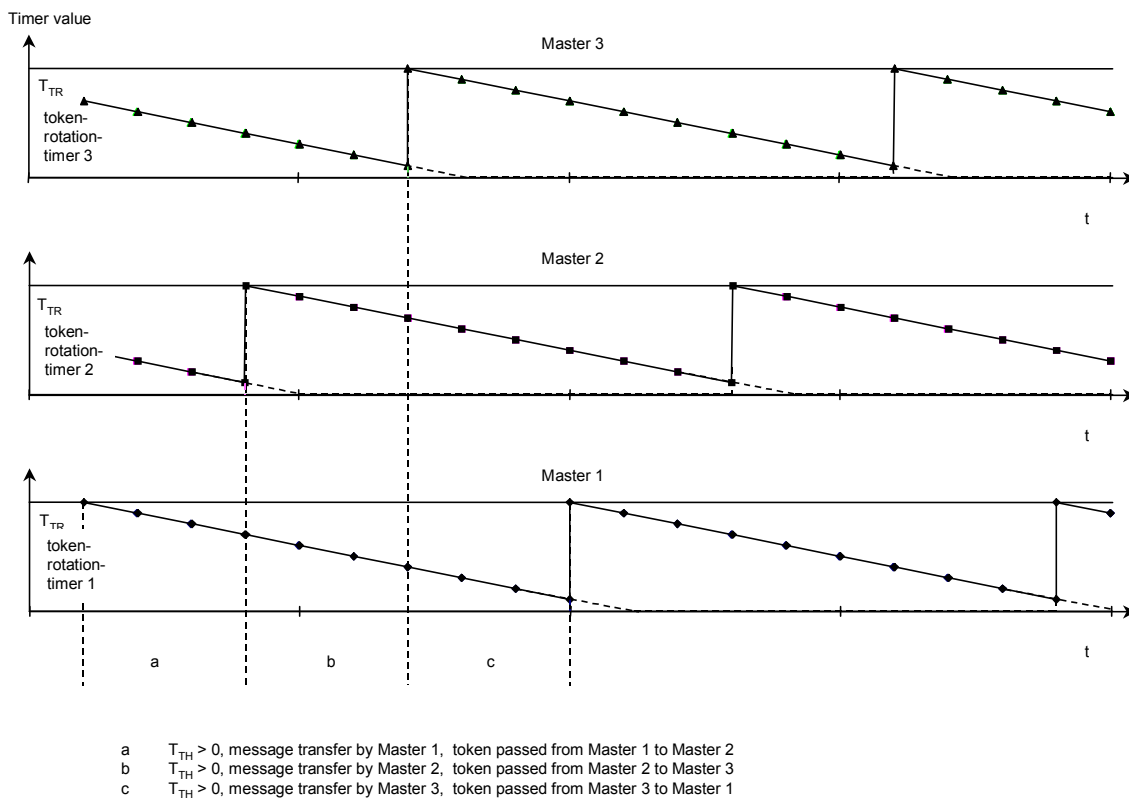


Figure F.3 – Usage of token holding time (T_{TH}) for message transfer (equivalence between T_{TH} of each Master station)

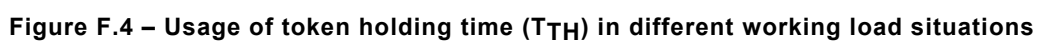
Case three: This case is characterized by a load change, from no load to a maximum load, that is a worst-case situation. This means that each token holder has a maximum amount of messages to send after a period of no message requests. That leads to the fact that each token holder tries to use the whole available token holding time for that token rotation. The Token Passing procedure guarantees a fair access to the medium for all token holders even in this situation.

In case of using the whole available bus transfer time (T_{TR}) by one token holder within one token cycle, no local T_{TH} time is available on token arrival in any of the other token holders for that cycle, so they are only allowed to send one high priority message. Within the next token cycle the token holder that had used the whole T_{TR} previously is limited to one high priority message, after which it immediately passes the token. This situation is depicted in Figure F.4. After a start up phase in which the token is passed by the token holders, Master station 1 uses the whole available bus transfer time to send messages (see Figure F.4: d). In this case Master station 2 and Master station 3 have no T_{TH} , indicated by a read value of zero of the token-rotation-timer at token receipt. Because Master station 3 has to transfer a high priority message it sends this message after receiving the token (see Figure F.4: f).

In the next token cycle (g), Master station 1 passes the token immediately after token receipt as its read timer value has reached zero and it has no high priority messages waiting. Now Master station 2 has the possibility to use the whole bus transfer time for message transfer within this token cycle (h). Master station 2 also uses all the available T_{TH} time because of its high working load, after which it is forced to pass the token to Master station 3.

Master station 3 is now limited by its local T_{TH} that has reached zero, and as it has no high priority messages, it passes the token immediately after receipt (see Figure F.4: i). In the following token cycle no T_{TH} is available for Master station 1 or Master station 2. They could send one high priority message, but in this example they pass the token immediately without

LICENSED TO MECON Limited - RANCHI/BANGALORE
FOR INTERNAL USE AT THIS LOCATION ONLY, SUPPLIED BY BOOK SUPPLY BUREAU.



F.3 Examples for message transfer periods – asynchronous transmission

The following examples give a detailed overview about the possible message transfer periods for asynchronous transmission according to typical messages used by the DLS-user:

Legend for the following examples:

T_{SC}	bit time to transmit a single character SC; (11 bits)
T_{SAP}	bit time to transmit D_SAP_index and S_SAP_index; (22 bits)
T_{SD1}	bit time to transmit a DLPDU with start delimiter SD1; (66 bits)
T_{SD2}	bit time to transmit a DLPDU with start delimiter SD2 (without DLSDU and Address-Extensions); (99 bits)
T_{SD3}	bit time to transmit a DLPDU with start delimiter SD3, DLPDU format of fix length with data field; (154 bits)
Req-PDU	number of octets in the DATA_UNIT
Res-PDU	number of octets in the DATA_UNIT

EXAMPLES

a) GAP request with reply (T_{GAP1})

$$T_{GAP1} = T_{ID1} + T_{SDR} + 2 \times T_{SD1} + 2 \times T_{TD} \quad (F.1)$$

b) GAP request with no reply (T_{GAP2})

$$T_{GAP2} = T_{ID1} + T_{SD1} + T_{SL} \quad (F.2)$$

c) Cyclic message transfer (T_{CY})

$$T_{CY1} = T_{ID1} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 11 \times (\text{Req-PDU} + \text{Res-PDU}) \quad (F.3)$$

$$T_{CY2} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SC} + 2 \times T_{TD} + 11 \times \text{Req-PDU} \quad (F.4)$$

$$T_{CY3} = T_{ID1} + T_{SD2} + T_{SL} + 11 \times \text{Req-PDU} \quad (F.5)$$

d) Acyclic message transfer with response (T_{ACR})

$$T_{ACR1} = T_{ID1} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 2 \times T_{SAP} + 11 \times \text{Res-PDU} \quad (F.6)$$

$$T_{ACR2} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SD3} + 2 \times T_{TD} + T_{SAP} \quad (F.7)$$

e) Acyclic message transfer with acknowledgement (T_{ACY})

$$T_{ACY1} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SC} + 2 \times T_{TD} + 11 \times \text{Req-PDU} \quad (F.8)$$

$$T_{ACY2} = T_{ID1} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SD1} + 2 \times T_{TD} + 11 \times \text{Req-PDU} \quad (F.9)$$

f) Broadcast message transfer (T_{BC})

$$T_{BC} = T_{ID2} + T_{SD2} + T_{SAP} + T_{TD} + 11 \times \text{Req-PDU} \quad (F.10)$$

F.4 Examples for message transfer periods – synchronous transmission

The following examples give a detailed overview about the possible message transfer periods for synchronous transmission according to typical messages used by the DLS-user:

Legend for the following examples:

T_{SC}	bit time to transmit a single character SC including CRC; (24 bits)
T_{SAP}	bit time to transmit D_SAP_index and S_SAP_index; (16 bits)
T_{SD1}	bit time to transmit a DLPDU with start delimiter SD1; (48 bits)
T_{SD2}	bit time to transmit a DLPDU with start delimiter SD2 (without DLSDU and Address-Extensions); (72 bits)
T_{SD3}	bit time to transmit a DLPDU with start delimiter SD3, DLPDU format of fix length with data field; (112 bits)
T_{PhL}	bit time to transmit preamble, physical start delimiter and end delimiters well as post-transmission gap time; unit in bit (40 bits: 16 bits preamble, 8 bits start delimiter and 8 bits end delimiter, 8 bits post-transmission gap time)
Req-PDU	number of octets in the DATA_UNIT
Res-PDU	number of octets in the DATA_UNIT

EXAMPLES

a) GAP request with reply (T_{GAP1})

$$T_{GAP1} = 2 \times T_{PhL} + T_{SDR} + 2 \times T_{SD1} + 2 \times T_{TD} \quad (F.11)$$

b) GAP request with no reply (T_{GAP2})

$$T_{GAP2} = T_{PhL} + T_{SD1} + T_{SL} \quad (F.12)$$

c) Cyclic message transfer (T_{CY})

$$T_{CY1} = 2 \times T_{PhL} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 8 \times (\text{Req-PDU} + \text{Res-PDU}) \quad (F.13)$$

$$T_{CY2} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SC} + 2 \times T_{TD} + 8 \times \text{Req-PDU} \quad (F.14)$$

$$T_{CY3} = T_{PhL} + T_{SD2} + T_{SL} + 8 \times \text{Req-PDU} \quad (F.15)$$

d) Acyclic message transfer with response (T_{ACR})

$$T_{ACR1} = 2 \times T_{PhL} + T_{SDR} + 2 \times T_{SD2} + 2 \times T_{TD} + 2 \times T_{SAP} + 8 \times \text{Res-PDU} \quad (F.16)$$

$$T_{ACR2} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SD3} + 2 \times T_{TD} + T_{SAP} \quad (F.17)$$

e) Acyclic message transfer with acknowledgement (T_{ACY})

$$T_{ACY1} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SC} + 2 \times T_{TD} + 8 \times \text{Req-PDU} \quad (F.18)$$

$$T_{ACY2} = 2 \times T_{PhL} + T_{SDR} + T_{SD2} + T_{SAP} + T_{SD1} + 2 \times T_{TD} + 8 \times \text{Req-PDU} \quad (F.19)$$

f) Broadcast message transfer (T_{BC})

$$T_{BC} = T_{PhL} + T_{SD2} + T_{SAP} + T_{TD} + 8 \times \text{Req-PDU} \quad (F.20)$$

Annex G (informative)

Type 7: Object Modeling

G.1 Modeling of the IDENTIFIER object

Class:	IDENTIFIER
Key Attribute:	I_Number
Attribute:	Type_APER.cyc (TRUE, FALSE)
Constraint:	Type_APER.cyc=TRUE
Attribute:	<i>Inherit from</i> BUFFER
Attribute:	Validity_ID_APER.cyc (TRUE, FALSE)
Attribute:	RQ (IN_PROGRESS, VACANT)
Attribute:	Type_MSG.cyc (TRUE, FALSE)
Constraint:	Type_MSG.cyc=TRUE
Attribute:	<i>Inherit from</i> QUEUE
	(Q_MSG_CYC)
Attribute:	Validity_ID_Consumed (TRUE, FALSE)
Attribute:	Type_Prod (TRUE, FALSE)
Constraint:	Type_Prod=TRUE
Attribute:	<i>Inherit from</i> BUFFER
	(B_DAT.prod)
Attribute:	Validity_ID_Produced (TRUE, FALSE)
Attribute:	Transmission (PERIODIC, APERIODIC)
Constraint:	Transmission=PERIODIC
Constraint:	Type_APER.cyc=FALSE
Attribute:	Aperiodic_Request (NON_SPEC_RQ_INHIBITED, FREE, NONE)
Constraint:	Aperiodic_Request=NON_SPEC_RQ_INHIBITED
Attribute:	<i>Inherit from</i> BUFFER
	(B_REQ)
Attribute:	RQ (IN_PROGRESS, VACANT)
Constraint:	Aperiodic_Request=FREE
Attribute:	RQ (IN_PROGRESS, VACANT)
Attribute:	PR (URGENT, NORMAL)
Attribute:	Reference QUEUE
	(Q_REQ1 OR Q_REQ2)
Constraint:	Type_MSG.cyc=FALSE
Attribute:	Type_MSG.aper (TRUE, FALSE)
Constraint:	Type_MSG.aper=TRUE
Attribute:	MSG (IN_PROGRESS, VACANT)
Attribute:	Reference FILE
	(Q_MSG.aper)

G.2 Description of the IDENTIFIER object attributes

G.2.1 I_Number

This attribute corresponds to a 16-bit word associated with a DL-subnetwork variable and which characterizes the variable uniquely in the DL-subnetwork.

G.2.2 Type_APER.cyc

This attribute characterizes the range of the buffer transfer specified requests.

When a local DLS-user performs a buffer transfer specified explicit request, the associated identifiers are deposited in the B_REQ buffer associated with the identifier indicated at the request.

This request can either remain local in case the B_REQ buffer is scanned during the cyclic part of the scanning table or be transmitted to the bus arbitrator in the other case.

This attribute takes the value TRUE when the buffer transfer specified explicit request service is supported by this identifier and that the scanning of the B_REQ buffer is included in the cyclic part of the scanning table (no RP_DAT_RQ to be transmitted). This service can be offered by produced, consumed, produced and consumed identifiers or by identifiers which are neither produced nor consumed.

This attribute takes the value FALSE.

- when no buffer transfer specified explicit request service is supported by this identifier,
- when the buffer transfer specified explicit request service is supported by this identifier but the request must be transmitted to the bus arbitrator (Scanning of the B_REQ buffer included in the cyclic part of the scanning table). This service can be offered only by the produced identifiers.

When the identifier supports the specified explicit request service, the Type_APER.cyc attribute corresponds to RQ_INHIBITED.

G.2.3 Inherit from BUFFER

The Identifier class inherits from the BUFFER class, thus giving it the attributes relative to B_REQ.

G.2.4 ID_APER.cyc validity

This binary attribute concerns any identifier supporting the buffer transfer specified explicit request periodic service. When this attribute takes the value TRUE, the list of identifiers contained in B_REQ transits on the bus after passage of the identifier request concerned. When this attribute takes the value FALSE, the producer/consumer entity does not answer the ID_RQ requests on this identifier.

G.2.5 RQ

This attribute characterizes the status of a buffer transfer request.

When the buffer transfer explicit requests are transmitted to the bus arbitrator, the requester has information representing the request status (bit RQ).

In the case of a specified request, this attribute has the value IN_PROGRESS, between the instant at which the request was placed in B_REQ and the end of transmission of the RP_RQ following reception of the corresponding ID_RQ type PDU.

In the case of a free request, this attribute has the value IN_PROGRESS between the instant at which a DLCEP-identifier is assigned to the queue in which the request has been placed and the end of transmission of the RP_RQ following reception of the corresponding ID_RQ type PDU.

Outside these intervals, the status of the request is VACANT to indicate that the identifier does not bear any buffer transfer explicit request.

G.2.6 Type MSG.cyc

Two message transfer modes are defined: one (aperiodic) uses a queue Q_MSG.aper reserved for this transfer mode and linked dynamically to identifiers, the other (cyclic) uses a set of Q_MSG.cyc queues of which each one is linked statically to a specific identifier.

This attribute takes the value TRUE when the scanning of the message identifier DLPDU is included in the bus arbitrator cyclic scanning (no RP_DAT_MSG to be transmitted). This service can be offered by produced, consumed, produced and consumed identifiers or by identifiers which are neither produced nor consumed.

It takes the value FALSE:

- when no messaging service is supported by this identifier,
- when the messaging service supported by this identifier requires that the request be transmitted to the bus arbitrator so that the message identifier DLPDU circulates in the messaging periodic window. This service can only be offered by produced identifiers (see Type_MSG.aper).

Inherit from FILE

The Identifier class inherits from the Queue class which contains all the attributes concerning the Q_MSG.cyc resources. The latter corresponds to the queue associated with each identifier supporting the cyclic messaging.

G.2.7 Type_Cons

This attribute defines if a DLCEP-identifier is consumed or not.

Inherit from BUFFER

The Identifier class inherits from the BUFFER class thus giving it the attributes concerning the B_DATcons.

G.2.8 ID_Consumed_Validity

This binary attribute concerns all consumed identifiers. When this attribute takes the value TRUE, a DLCEP-identifier consumes the data meant for it. When this attribute takes the value FALSE, a DLCEP-identifier does not answer in consumption to the requests of the bus arbitrator on this identifier.

G.2.9 Type_Prod

This defines if a DLCEP-identifier is produced or not.

Inherit from BUFFER

The identifier class inherits from the BUFFER class thus giving it the attributes concerning the B_DAT.prod.

G.2.10 Produced ID validity

This binary attribute concerns all the produced identifiers. When this attribute takes the value TRUE, the value contained in B_DAT.prod transits on the bus after passage of the variable identifier.

When this attribute takes the value FALSE, a DLCEP-identifier does not reply in production to the requests of the bus arbitrator on this identifier. Besides, the possible associations between this identifier and Q_RREQ1, Q_REQ2 or Q_MSG.aper are interrupted.

G.2.11 Transmission

A DLCEP-identifier is exchanged periodically if it appears in the cyclic part of the bus arbitrator scanning table. Otherwise the exchange is triggered on explicit request of a DLS-user.

This attribute conditions the possibility of having some services on this identifier; buffer transfer specified explicit request with RQ_INHIBITED false, buffer transfer free explicit request and periodic messaging.

It thus allows restricting the latter to cyclic identifiers only.

Aperiodic request

Three buffer transfer explicit request services are defined:

- The identifier which will support the request is specified. This request is not transmitted to the bus arbitrator. The B_REQ is scanned cyclically (RQ_INHIBITED true). In this case the attribute takes the value TRUE.
- The identifier which will support the request is specified. This request is transmitted to the bus arbitrator, the B_REQ buffer is not scanned cyclically. In this case the RQ_INHIBITED attribute takes the value FALSE and the Aperiodic_Request attribute the value SPEC_NON_RQ_INHIBITED value.
- The identifier which will support the request is not specified. The Aperiodic_Request takes the value FREE.

These three types of service are exclusive. If the Aperiodic_Request attribute has the value NOTHING, the identifier will not support any.

G.2.12 Inherit from BUFFER

The identifier class inherits from the BUFFER class, thus giving it the attributes relative to B_REQ.

G.2.13 RQ

See above.

G.2.14 PR

This attribute indicates if the buffer transfer free explicit request must be processed urgently or at the normal speed.

For a specified explicit request, only the urgent priority exists.

The existence of two priority levels for the free explicit requests implying the corresponding resources in the station (Q_REQi) and in the bus arbitrator (Q_IDRQi).

G.2.15 Reference_QUEUE

This attribute allows designating the Q_REQ1 or Q_REQ2 resource, objects resulting from the instance of the queue class. These queues are available to all the identifiers supporting the buffer transfer free explicit request service.

Q_REQ1 contains list(s) of identifiers from one or several normal free explicit requests which have not yet been transmitted to the bus arbitrator.

G.2.16 MSG.aper type

Two message transfer modes are defined: one (aperiodic) uses an queue reserved for this transfer mode and dynamically linked to identifiers, the other (cyclic) uses a set of queues each of which is statically linked to a specific identifier.

This attribute takes the value TRUE when the scanning of the message identifier DLPDU is not included in the bus arbitrator cyclic scanning, that is it is necessary to transmit a bus arbitrator request so that consequently the message identifier DLPDU circulates in the course of the aperiodic window reserved for the messaging. This service can only be offered by produced identifiers.

It takes the value FALSE:

- When no messaging service is supported by this identifier,
- When the scanning of the message identifier DLPDU is included in the bus arbitrator cyclic scanning (see Type_MSG.cyc).

G.2.17 MSG

This attribute characterizes the state of a message aperiodic transfer request.

It has the value IN_PROGRESS between the instant when the identifier has been associated with the queue reserved for transfer of messages and the reception of the message transaction linked to the corresponding ID_MSG DLPDU. Outside this interval, the status of the request is VACANT to indicate that the identifier bears no message transfer request.

G.2.18 QUEUE_Reference

This attribute allows designating the Q_MSG.aper object resource resulting from the instance of the Queue class. This queue is available to all the identifiers supporting the aperiodic messaging service.

It contains the message(s) resulting from one or several message transfer requests which have not yet been satisfied.

G.3 Modeling of the QUEUE object

Class:	QUEUE
Key Attribute:	Q_Number
Attribute:	Queue_status (EMPTY, FULL, PLACE AVAILABLE)
Attribute:	Type (Q_REQ, Q_MSG)
Constraint:	Type=Q_REQ
Attribute:	Content Q_REQ (identifier lists queue)
Attribute:	Connection (TRUE, FALSE)
Constraint:	Type=Q_MSG
Attribute:	Content Q_MSG (message queue)
Attribute:	Num_Attachment (integer)

G.4 Description of the QUEUE object attributes

G.4.1 Q_Number

This attribute associated with a queue allows characterizing the latter uniquely in the DLE.

G.4.2 Queue_status

This attribute describes the actual use of the queue. The possible values are:

An Q_MSG queue is empty when it contains no message. An F_MSG queue can contain an empty message; this will consist only of the source address and the destination address. In this case the Queue_Status attribute takes the value PLACE AVAILABLE or FULL.

G.4.3 Q_REQ content

This attribute corresponds to the content of the Q_REQ1 and Q_REQ2 queues. It is the set of identifier lists which have not yet been transmitted to the bus arbitrator. Each list has been covered locally by a free explicit request and is transmitted on the local link automatically. At an RP_RQ reply several lists can be transmitted in so far as all these lists are not made of more than 64 identifiers.

G.4.4 Attachment

This attribute indicates if the queue is associated with a DLCEP-identifier or not.

G.4.5 Q_MSG content

This attribute corresponds to the content of the Q_MSG.received, Q_MSG.Cyc and Q_MSG.aper. files. It is a messaging file.

In the case of Q_MSG.cyc and Q_MSG.aper. files, it includes all the messages covered by a local message transfer request which has not been satisfied. In the case of Q_MSG.received, it is the set of messages for which the station has been recognized as destination.

G.4.6 Attachment_Number

This attribute indicates how many associations are missing between the queue and its identifiers. It is a whole number greater than 0 and lower than the number of places in the queue. It thus corresponds to the number of messages present in the queue less the number of identifiers associated with the latter.

G.5 Modeling of the BUFFER object

Class:	BUFFER
Key Attribute:	B_Number
Attribute:	Type (B_DAT, B_REQ)
Attribute:	Availability (OCCUPIED, FREE)
Constraint:	Type=B_DAT
Attribute:	Content Prod_Cons (variable value)
Constraint:	Type=B_REQ
Attribute:	Content B_REQ (list of identifiers)

G.6 Description of the BUFFER object attributes

G.6.1 B_Number

This attribute associated with a buffer allows characterizing the latter uniquely in a DLE.

G.6.2 Type

This attribute allows determining if the buffer is of the B_DAT type (produced or consumed) or B_REQ.

G.6.3 Availability

This attribute is meant to solve the problems of access conflict which could take place.

G.6.4 Prod_Cons content

This attribute corresponds to the content of a B_DAT or B_DATcons buffer. This is the value of a produced or consumed variable.

G.6.5 B_REQ content

This attribute corresponds to the content of a B_REQ buffer. It is a list of identifiers coming from a buffer transfer specified explicit request.

Annex H (informative)

–

Type 7: Topology of multi-segment DL-subnetwork

H.1 Introduction

The purpose of this appendix is to describe how to specify the topology of a multi-segment DL-subnetwork. The aim is to propose a data structure, which could be minimal while allowing correct operation of the bridge retransmission function.

The topology of a DL-subnetwork can first be specified globally, in order to verify a certain number of properties (connectivity, non-meshing, etc.); then on the basis of this specification the local data base specific to each bridge must be calculated in order to ensure it operates correctly.

Although this appendix proposes a method to achieve this goal, only the specifications of the data structures, global or local to each bridge, which define the DL-subnetwork topology, as well as the properties which it should fulfil, must be taken into account in the standard. The suggested method shows how to obtain a solution to the problem by taking into account certain optimization problems.

H.2 Global specification

The topology of a multi-segment DL-subnetwork can be defined by the following elements:

- the set S of its segments:
$$S = \left\{ S_i \mid i \in [1, n] \right\}$$
- the set B of its bridges:
$$B = \left\{ b_k \mid k \in [1, m] \right\}$$
- and for each bridge of B, the data of a matrix B^k of dimension $n \times n$. whose coefficients b_{ij}^k are defined by:
 - $b_{ij}^k = 0$ if $i = j$.
 - $b_{ij}^k = \bullet$ if the bridge b^k does not allow transfer of messages from segment S_i to segment S_j .
 - $b_{ij}^k = a$ with $a \in \mathbf{R}^{+*}$, if the bridge b^k allows the transfer of messages from segment S_i towards segment S_j , with a as load coefficient which allows taking into account of a different efficiency rate according to the transfers.

A load coefficient b_{ij}^k can represent the load, as a rate of occupation of the medium, of the retransmission segment s_j . In reality, either the destination is directly s_j , or there are several paths possible, passing through intermediate segments, to reach s_j , and in this case the choice shall be to pass by the least loaded path.

It is of course possible to take as coefficients of the same value (1 for example).

If a bridge allows two-way retransmission with the same load coefficient for the two directions, its matrix is symmetrical.

The matrix B^k of a bridge also allows knowing all the segments to which it is connected:

- either in reception, $S_r^k = \{ \text{segments whose corresponding line in the matrix includes at least one non-null finite coefficient} \}$ note $n_r^k = \text{card} (S_r^k)$,
- or in transmission, $S_e^k = \{ \text{segments whose corresponding column in the matrix includes at least one non-null finite coefficient} \}$; note $n_e^k = \text{card} (S_e^k)$.

H.3 Local specification

The information which a bridge must have locally allows it to answer the following question: when I receive a message on such a segment $sr_i \in S_r^k$ destined for such another segment s_j , must I do nothing or must I retransmit on segment $se_j \in S_e^k$.

To fulfil this purpose, it is enough to allocate to each bridge b^k a transfer matrix T^k with dimensions $n_r^k \times n$, whose elements r_{ij}^k are defined by:

- the line index $i \in [1, n_r^k]$ references segments sr_i connected in reception ($\in S_r^k$).
- the column index $j \in [1, n]$ references the segments s_j of the DL-subnetwork ($\in S$),
- $r_{ij}^k = 0$ if on reception of a message on segment $sr_i \in S_r^k$ addressed to segment s_j , the bridge must retransmit to segment se_j .
- $r_{ij}^k = se_j$, with $se_j \in S_e^k$; if on reception of a message on segment $sr_i \in S_r^k$ addressed to segment s_j , the bridge must retransmit to segment se_j .

NOTE Indexes i and j correspond to channel numbers whereas sr_i is the segment connected in reception to channel i and se_j is the segment connected in transmission to channel j .

H.4 Properties

The properties which should satisfy the DL-subnetwork are connectivity and non-meshing.

Connectivity consists in ensuring that there is always a path from any given segment of S to any other segment of S .

Non-meshing consists in ensuring that the transmission of a message from a transmitter located on segment s_e and addressed to a receiver located on segment s_r can be routed by only one path (thus preventing the message from being received more than once).

In fact, it is the definition of the local specification of each bridge and the calculation of its transfer matrix which ensure this property: by definition, on reception of a message on segment sr_i addressed to segment s_j , either the bridge does not retransmit it, in particular if segment sr_i is equal to segment s_j , or the bridge retransmits it on a single segment se_j , whereas by calculation

of the matrix, it is necessary to make sure that one and only one bridge, connected in reception to segment s_{r_i} , retransmits the message.

H.5 Methods

The method consists in calculating the matrix C of the minimum loads of the paths between any two segments. By this way we check the connectivity since none of these coefficients is infinite.

The second stage consists in calculating the transfer matrix of each bridge so that this gives the global DL-subnetwork the property of non-meshing while preserving the property of connectivity.

H.5.1 Minimum load matrix

a) P row load matrix

Definition: the load matrix of rank P , C^P , with dimensions $n \times n$, is the matrix whose coefficients C_{ij}^P give the minimum load to travel from segment i towards segment j by passing via not more than P bridges.

We have:

- $C_{ij}^P = 0$,
- $C_{ij}^P = \bullet$, if there is no path from segment i to segment j by passing via not more than P bridges,
- if C_{ij}^P is finite, the optimal corresponding path includes not more than P bridges (it can include less than P).

Obtaining by recurrence:

- The coefficients C_{ij}^1 , of the load matrix of row 1, C^1 , are given by:

$$C_{ij}^1 = \min_k \in [1, m] \left(b_{ij}^k \right)$$

We have:

- $C_{ij}^1 = 0$,
- $C_{ij}^1 = \bullet$, if there is no permanent bridge allowing transfer from segment i towards segment j ,
- C_{ij}^1 finite, if there exists one or more bridges allowing the transfer from segment i towards segment j .

- the coefficient C_{ij}^P , for $P > 1$, of the load matrix of row P , C^P , are given by:

$$C_{ij}^P = \min_k \in [1, m] \left(C_{ik}^1 + C_{kj}^{P-1} \right)$$

In reality, the minimum load between two segments i and j passing via P bridges corresponds to a path composed of:

- a bridge allowing the passage from segment i to segment k , with a minimum load c_1 ,
- and a path with minimum load c_2 between this segment k and segment j , passing via $P-1$ bridges.

The intermediate bridge is most often used so that $c_1 + c_2$ are minimal.

b) Minimum load matrix

Definition: the minimum load matrix C , dimension $n \times n$, is the matrix whose coefficients c_{ij} give the minimum load to go from segment i to segment j . We thus have:

- $c_{ij} = 0$
- $c_{ij} = \infty$, if there is no path from segment i to segment j ,
- if c_{ij} is finite, there is a path $s_1 = i, s_2 \dots s_L = j$ with length L , passing via bridges b_{k^*} ,

$$c_{ij} = \sum_{h=1}^{h=L-1} b_{s_h s_{h+1}}^{k^*}$$

— $h \in [1, L - 1]$, whose load is c_{ij} with $c_{ij} =$

By definition the connectivity is well ensured by the fact that all the minimum load matrix coefficients C are finite.

Property: the minimum load matrix C is the limit of the load matrixes of row P , when P tends to

$$C = \lim_{P \rightarrow \infty} C^P$$

infinity:

In reality, the series C^P is stationary, at least from row m , where m is the number of bridges. A path which passes via more than m bridges passes at least twice via the same bridge and cannot thus have a minimum load.

Suppose Q the row from which the C^P series is stationary ($C^{Q+1} = C^Q$).

H.5.2 Calculation of the transfer matrices

Suppose now that the DL-subnetwork is connected.

The transfer matrices T^k are calculated by iteration according to the number P of bridges, from 1 to Q , requiring a minimum load path from a source segment s and a destination segment d .

At the start, $T^k = 0$ for every k .

The transfer matrix coefficients are referenced in the same manner as in L.3, that is:

- the line index $i \in [1, n^{r^k}]$ references the s_{r^k} segments connected in reception ($\in S_{r^k}$),
- the column index $j \in [1, n]$ references the s_j segments of the DL-subnetwork ($\in S$).

a) Passage of a segment s to a segment d via 1 bridge

- For all pairs of segments s and d such that c_{sd}^1 is finite.
- for one and only one k (as selected) such that $c_{sd}^1 = b_{sd}^k$,
- the following assignment is performed for the transfer matrix T^k :
- for $i \in [1, n_r^k]$ such that $s_{r_i} = s_j$ for $j \in [1, n]$ such that $s_j = d, r_{ij}^k = s_j$

b) Passage of a segment s to a segment d via P bridges

- For every pair of segments s and d such that c_{sd}^p is finite whereas c_{sd}^{p-1} is infinite.
- For one and only one k (as selected) such that $\exists s' \mid C_{sd}^P = b_{ss'}^k + c_{sd}^{P-1}$,
- the following assignment is thus performed for the T^k transfer matrix:
- for $i \in [1, n_r^k]$ such that $s_{r_i} = s_j$ for $j \in [1, n]$ such that $s_j = d \dots r_{ij}^k = s_j$ with $s_j = s'$

NOTE In the last two paragraphs, the bridge k which verifies the necessary property is not necessarily unique, but an assignment must be made for a single bridge to ensure the property of non-meshing.

Annex I (informative)

Type 7: Management of transmission errors

I.1 Transmission of RP_DAT_XX

If a transmission error is detected during the transmission of a given response DLPDU, the transmission is interrupted and:

- The DL-SEND primitive is not generated.
- In the case of a configured identifier for the service of the free explicit request service, the RQ indicator is not modified,
- In the case of a DLCEP-identifier configured for message aperiodic transfer, the MSG indicator value is unchanged.

The transmission of DL-SEND_ind and the management of the RQ and MSG indicators are performed only at the end of the RP_DAT_XX DLPDU transmission and if the variable producer has not detected an error during transmission. The evaluation DL-subnetwork of Figure I.1 describes this operation.

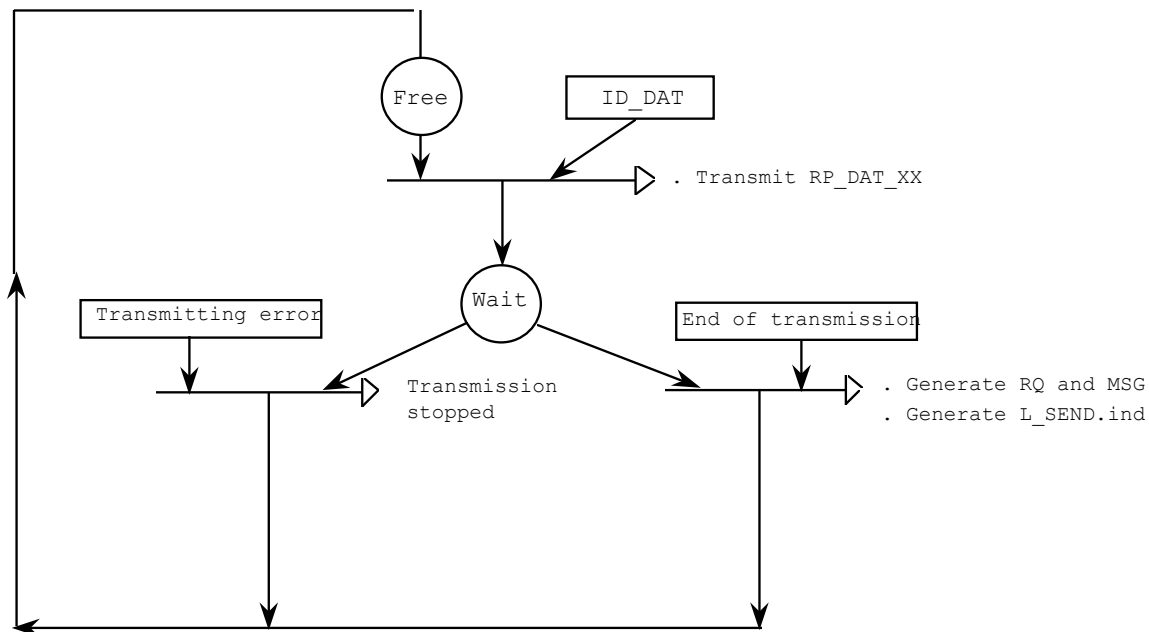


Figure I.1 – Evaluation DL-subnetwork for transmission of RP_DAT_XX

I.2 Transmission of a free RP_RQ(1/2)

On detection of an error during transmission of RP_RQ1 or RP_RQ2, forming continuation of a buffer transfer free explicit request, the producing entity does not generate any DL-FREE-UPDATE confirm primitive and does not scroll the list of identifiers. The evaluation DL-subnetwork of Figure I.2 describes this operation.

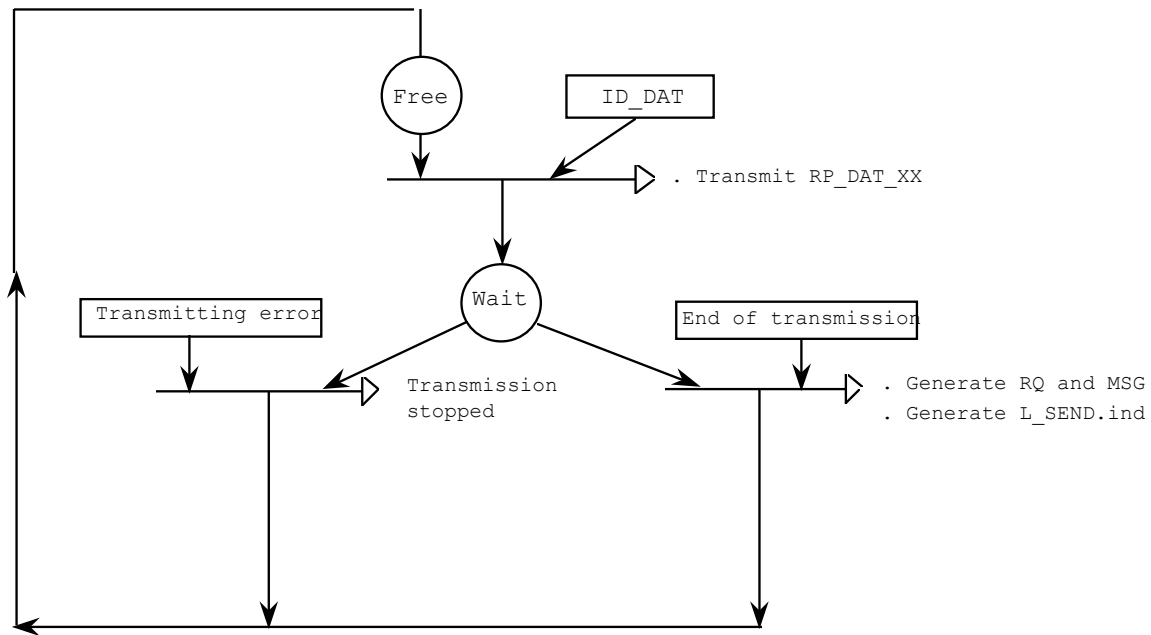


Figure I.2 – Evaluation DL-subnetwork for transmission of a free RP_RQ(1/2)

After transmission of the RP_RQ without error, the request is detached from the identifier concerned (RQ set to 0); If the FIFO is not empty, the request will be attached at the next reception of an ID_DAT concerning a DLCEP-identifier configured for this service. The RQ bit of this identifier shall be set to 1.

I.3 Transmission of the specified RP_RQ1

On detection of an error during transmission of RP_RQ1, following a buffer transfer specified explicit request, the producing entity interrupts the transmission of the DLPDU and does not generate a DL-SPEC-UPDATE confirm primitive.

Besides, the RQ indicator remains set to 1: the buffer transfer request shall be transmitted to the BA at the next transaction in case the RQ_INHIBITED indicator of the identifier has the FALSE value. The evaluation DL-subnetwork of Figure I.3 describes this operation.

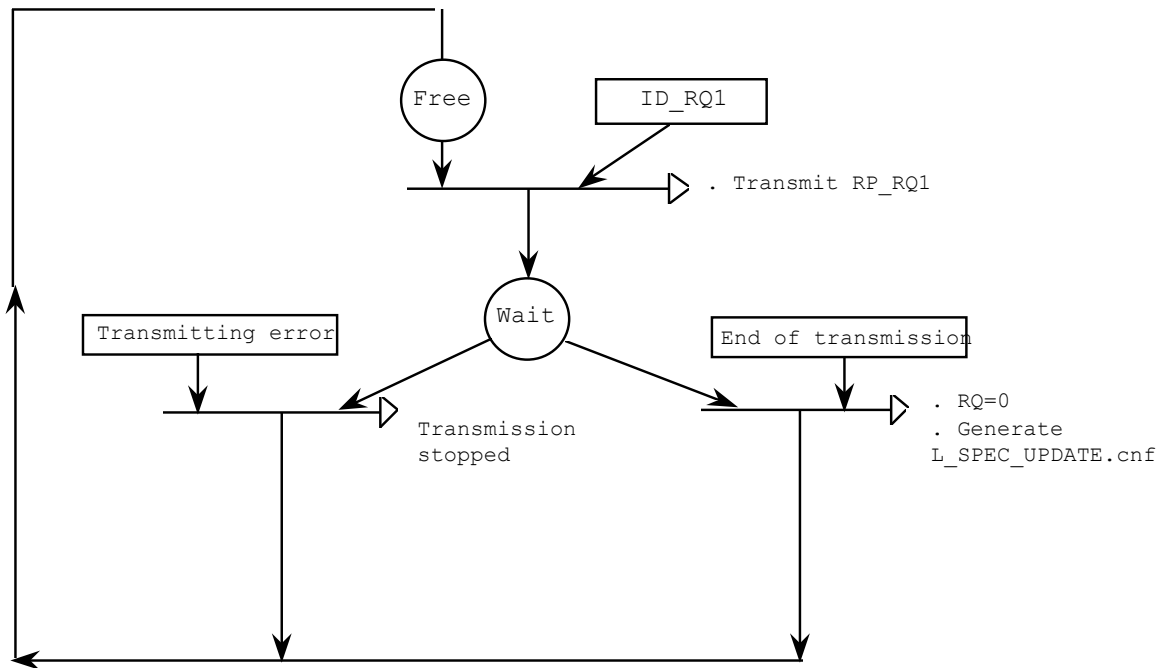


Figure I.3 – Evaluation DL-subnetwork for transmission of the specified RP_RQ1

I.4 Transmission of RP_MSG_NOACK

Several behaviors are possible during an error at transmission of an RP_MSG_NOACK:

- The producer/consumer entity interrupts the transmission and restores in the initial context, keeping the message. The transmission of the RP_MST_NOACK is then postponed to the next request from the bus arbitrator.
- The producer/consumer entity interrupts the transmission keeping only the message. The link layer indicates the loss of the message to the DLS-user with the help of the appropriate confirmation.

I.4.1 First behavior

On detection of an error during transmission of RP_MSG_NOACK, the producing entity interrupts the transmission of the DLPDU.

Besides,

- It does not generate the primitive of the DL-MESSAGE-ACK confirm primitive.
- The place in the message queue is not freed.
- If the identifier associated with the request is not included in the BA periodical scanning table, the MSG indicator remains positioned to 1; the message transfer request will be transmitted to the BA at the next transaction.

The evaluation DL-subnetwork of Figure I.4 describes this operation.

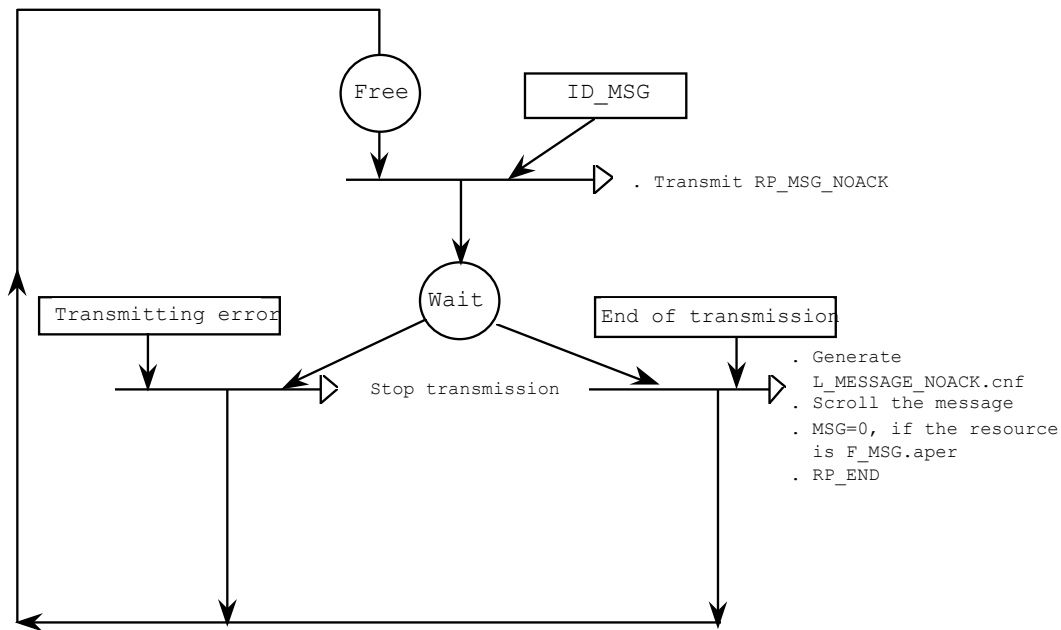


Figure I.4 – Evaluation DL-subnetwork for transmission of RP_MSG_NOACK, first behavior

I.4.2 Second behavior

On detection of an error during the transmission of RP_MSG_NOACK, the producing entity interrupts the transmission of the DLPDU.

Besides,

- It generates a DL-MESSAGE-ACK confirm primitive whose status signifies Transmission error (1)
- The place in the queue is freed.

The evaluation DL-subnetwork of Figure I.5 describes this operation.

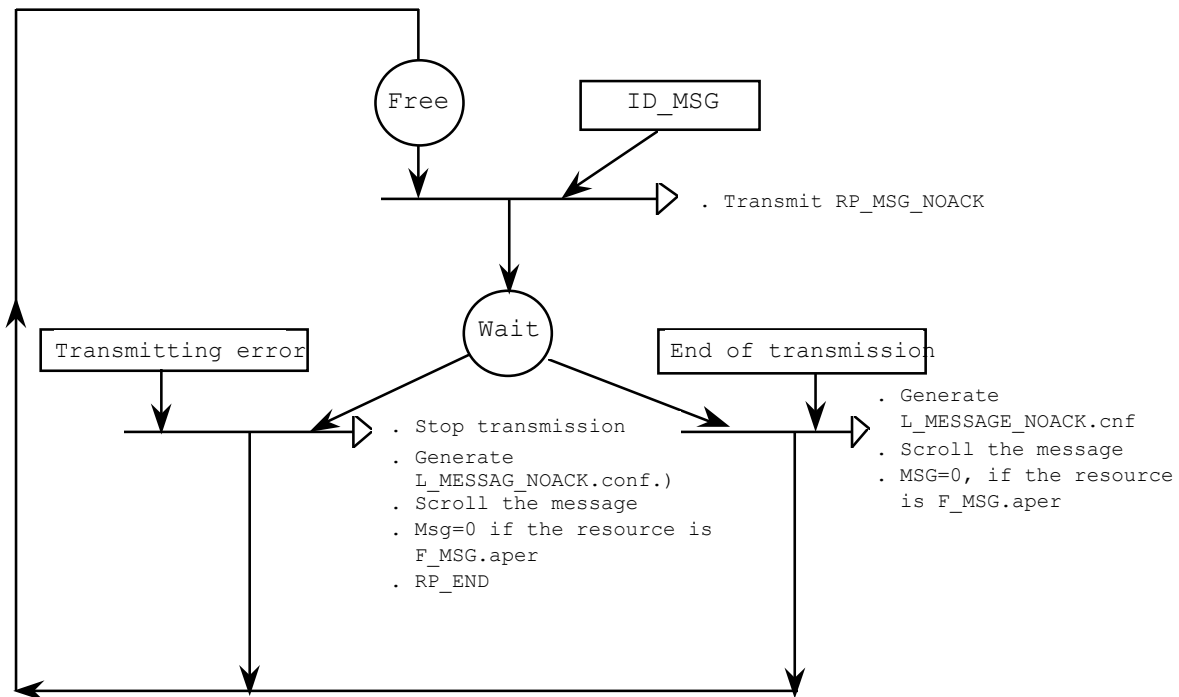


Figure I.5 – Evaluation DL-subnetwork for transmission of RP_MSG_NOACK, second behavior

NOTE This status, not provided for by this standard, is to be added in the case of choice of this behavior at error during transmission.

I.5 Transmission of RP_MSG_ACK

Several behaviors are possible during an error at transmission of RP_MSG_ACK:

- The producer/consumer entity interrupts the transaction and is restored in the initial context, keeping the message. The transmission of the RP_MSG_ACK is then postponed to the next request coming from the bus arbitrator.
- The producer consumer entity interrupts the transmission and if the restart counter permits, retransmits the RP_MSG_ACK DLPDU immediately.

I.5.1 First behavior

On detection of an error during transmission of RP_MSG_ACK, the producing entity interrupts the transmission of the DLPDU.

Besides,

- It does not generate the DL-MESSAGE-ACK confirm primitive.
- The place in the message queue is not freed.
- If the identifier associated with the request does not form part of the BA periodic scanning table, the MSG indication shall be positioned at 1; the message transfer request will be transmitted to the BA at the next transaction.

The evaluation DL-subnetwork of Figure I.6 describes this operation.

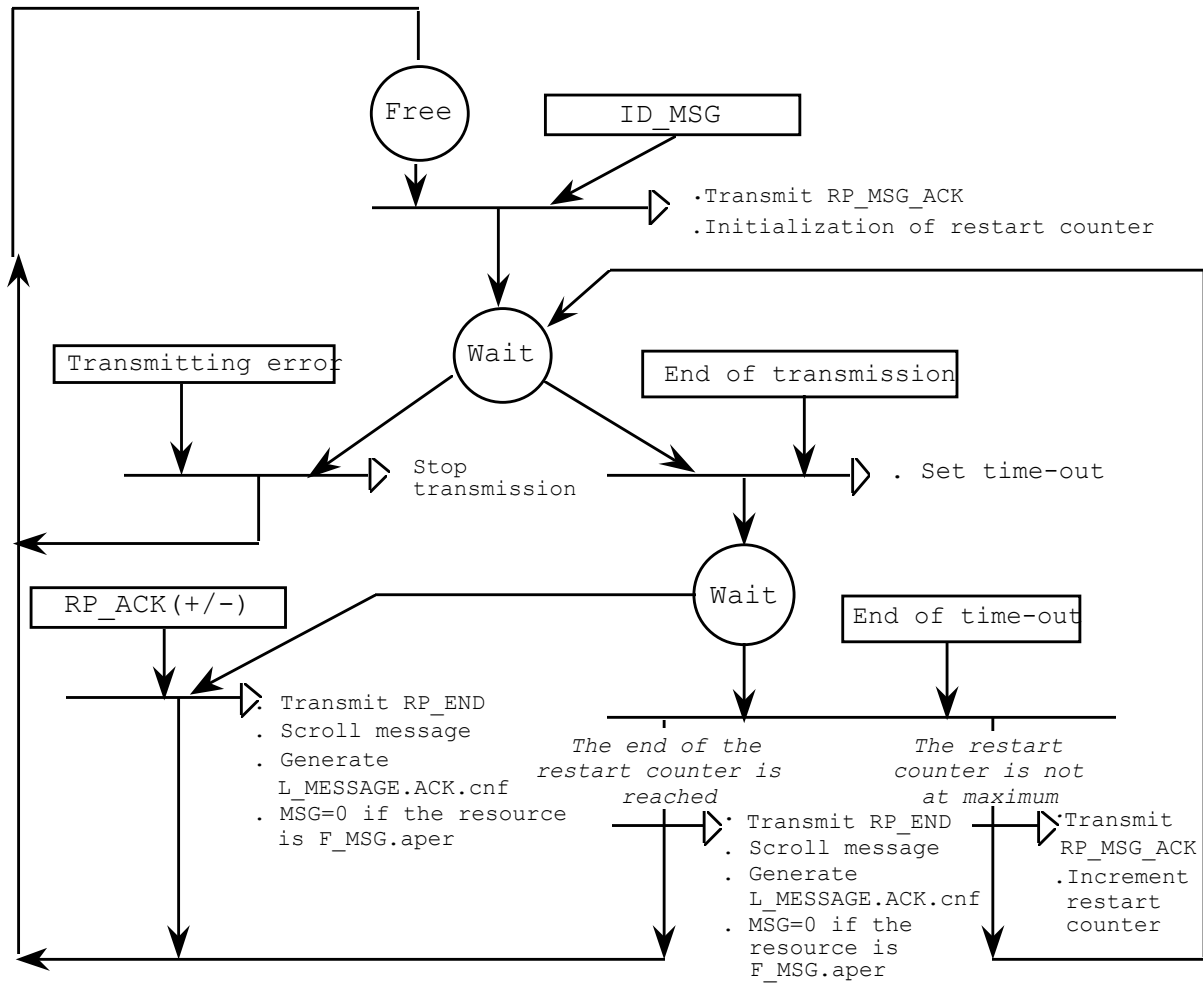


Figure I.6 – Evaluation DL-subnetwork for transmission of RP_MSG_ACK, first behavior

I.5.2 Second behavior

On detection of an error during transmission of the RP_MSG_ACK DLPDU, the actions by the producing entity are the following:

- Interruption of the DLPDU transmission
- If the restart counter is not reached the producing entity performs a restart immediately without waiting for the expiration of time-out T6.

The evaluation DL-subnetwork of Figure I.7 describes this operation.

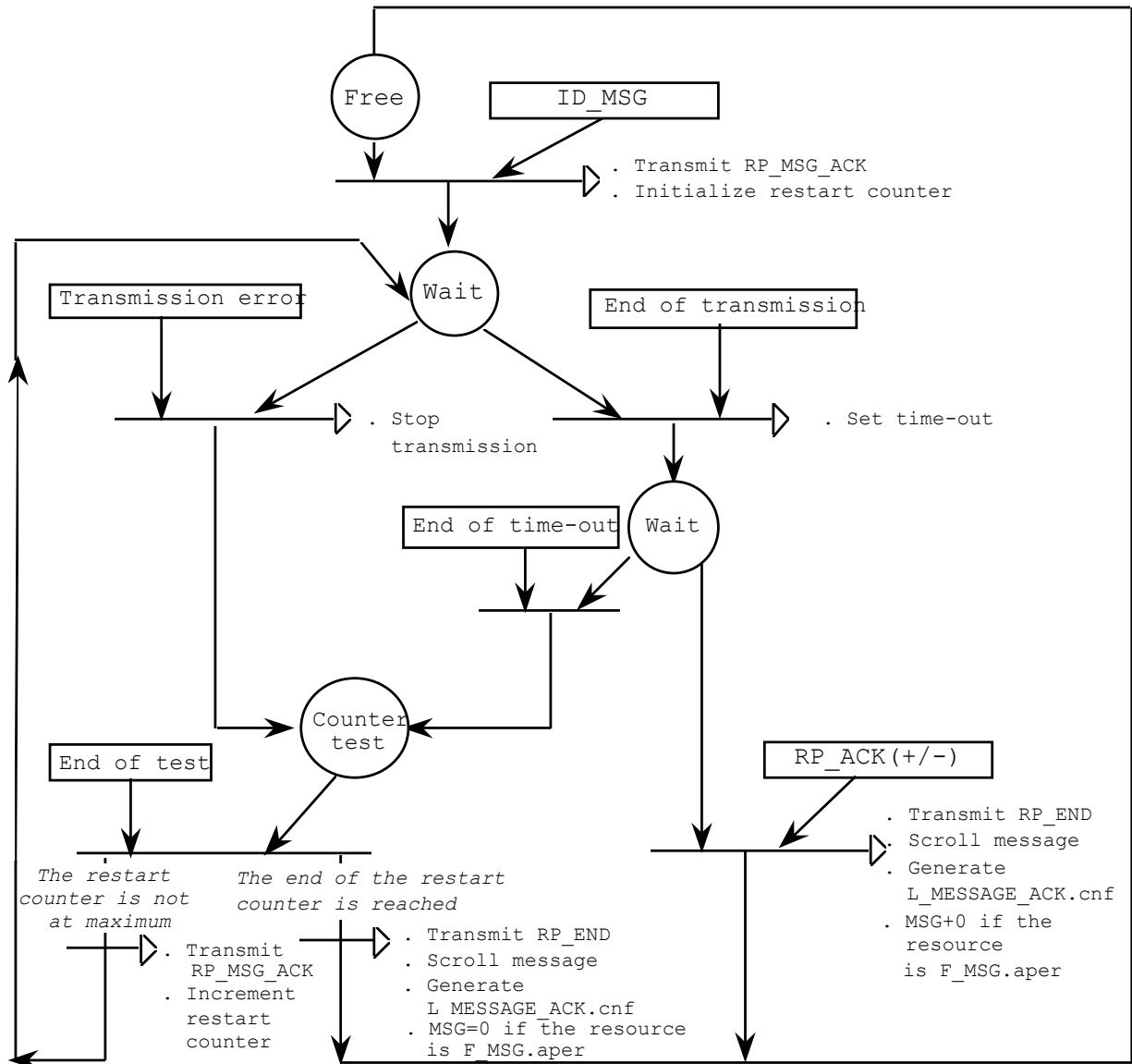


Figure I.7 – Evaluation DL-subnetwork for transmission of RP_MSG_ACK, second behavior

Annex J (informative)

Type 8: Implementation possibilities of definite PNM2 functions

J.1 Acquiring the current configuration

J.1.1 Configuration data memory in the master

The configuration data is stored with slave information code as an image of the ring in the sequence of the position of the slaves in the ring. The ring segment level contains the level of the slave if the bus configuration is depicted as a tree structure. Figure J.1 shows a DL-subnetwork configuration in the form of a tree structure and the ring segment level of the individual branches. The positions of the slaves in the ring correspond to their numbering:

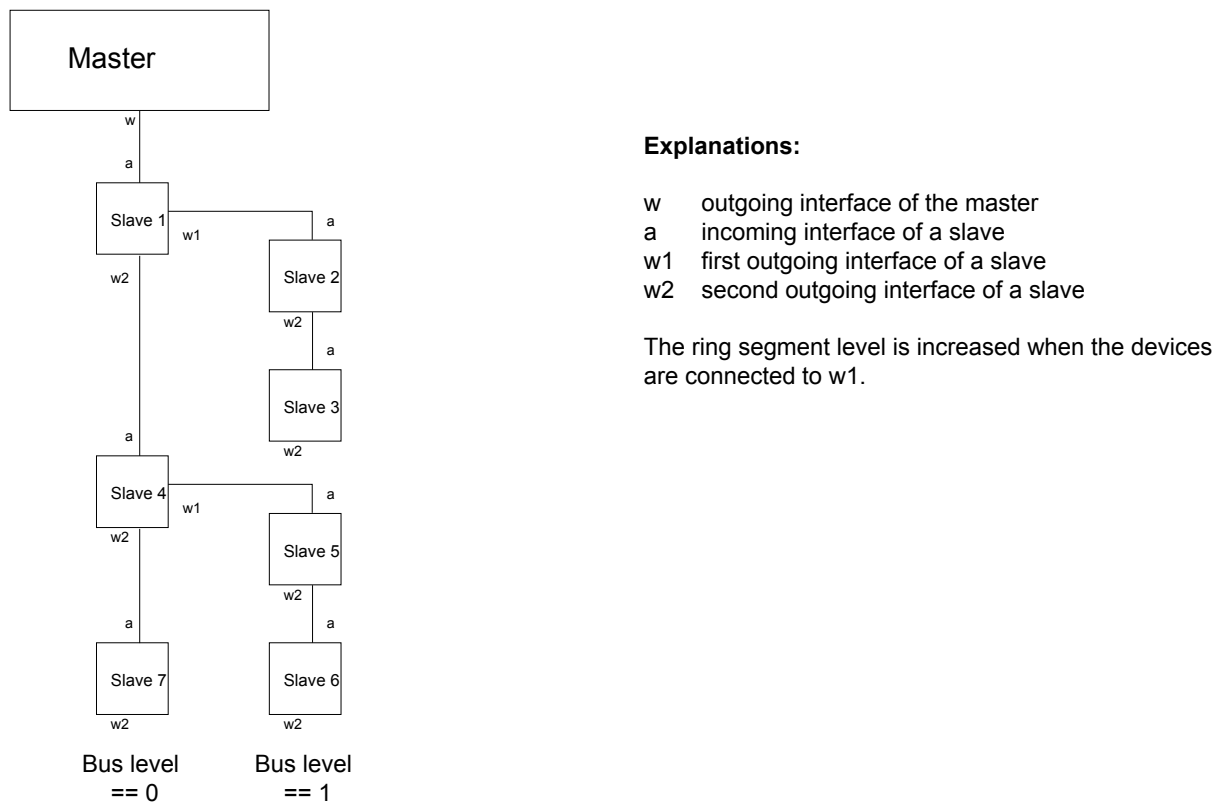


Figure J.1 – DL-subnetwork configuration in the form of a tree structure

The DL-subnetwork configurations are stored in the master as matrices (see Table J.1):

Table J.1 – DL-subnetwork configuration in the form of a matrix

Device code	Ring segment level	No. of the slave
...	...	1
...	...	2
...	...	3
...	...	4
...	...	...
...	...	n

J.1.2 Acquire_Configuration function

The function is described with the `Acquire_Configuration.request` and `Acquire_Configuration.confirm` primitives. The `Acquire_Configuration.request` primitive does not contain any parameters, the `Acquire_Configuration.confirm` primitive contains the configuration to be acquired with a result (+) or the `error_code` with a result (-) (see Table J.2).

Table J.2 – Acquire_Configuration

Parameter name	Request	Confirm
Argument	M	
Result (+) <code>current_configuration</code>		S M
Result (-) <code>error_code</code>		S M

result (+):

A current configuration could be acquired.

current_configuration:

This parameter contains the currently acquired configuration in the form of the device codes and the ring segment level as matrix.

result (-):

No configuration could be acquired.

error_code:

The `error_code` describes the error cause. Possible errors are:

— Too many cycles with errors when the configuration was acquired.

J.1.3 State machine for the acquisition of the configuration

J.1.3.1 General

The current configuration is acquired by connecting the outgoing interfaces of the slaves step by step. In order to get an algorithm that is as fast as possible and has short bus cycle times, the interfaces are closed again when the end of a branch is reached.

As the procedure is repeated for every outgoing interface, the acquisition of the configuration can be described by a recursion. The `Acquire_Configuration.request` primitive is called again for every outgoing interface. Thus, the state machine shown in Figure J.2 applies to every call of the `Acquire_Configuration.request` primitive:

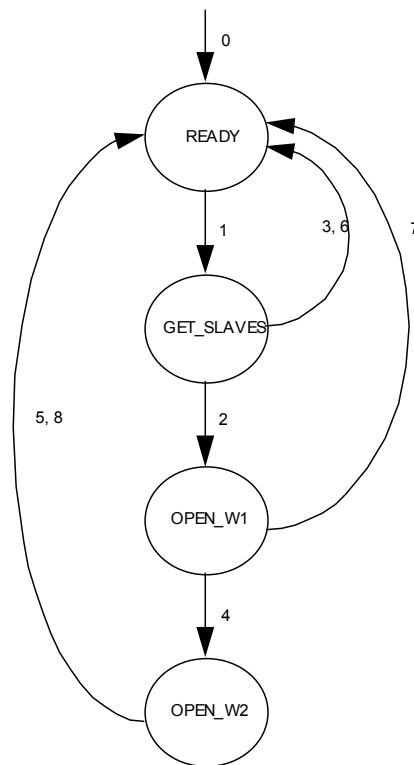


Figure J.2 – State machine for the acquisition of the current configuration

J.1.3.2 States of the state machine

READY

The state machine is ready to respond to an `Acquire_Configuration.request` primitive.

GET_SLAVES

The state machine runs ID cycles without connecting the outgoing interfaces to identify the slaves which are already in the ring.

OPEN_W1

The state machine opens the outgoing interface W1 of the last identified slave and acquires the configuration at W1 by means of an `Acquire_Configuration.request` primitive.

OPEN_W2

After the configuration has been acquired the state machine closes the outgoing interface W1, opens the outgoing interface W2 of the last identified slave and acquires the configuration at W2 by means of an `Acquire_Configuration.request` primitive.

Table J.3 describes the state transitions.

Table J.3 – State transitions of the state machine for the acquisition of the current configuration

Initial state event condition ⇒ action	Transition	Follow-up state
Power on	0	READY
READY Acquire_Configuration.request ⇒ run ID cycles without opening or closing the outgoing interfaces, to identify slaves which already exist in the ring.	1	GET_SLAVES
GET_SLAVES ID cycles ended at least one new slave could be detected ⇒ open W1, Acquire_Configuration.request	2	OPEN_W1
GET_SLAVES ID cycles completed no new slave could be detected ⇒ Acquire_Configuration.confirm (+)	3	READY
OPEN_W1 Acquire_Configuration.confirm (+) ⇒ close W1, open W2, Acquire_Configuration.request	4	OPEN_W2
OPEN_W2 Acquire_Configuration.confirm (+) ⇒ close W2, Acquire_Configuration.confirm (+)	5	READY
GET_SLAVES, OPEN_W1 or OPEN_W2 Acquire_Configuration.confirm (-) or several ID cycles could not be completed without errors ⇒ Acquire_Configuration.confirm (-)	6 - 8	READY

J.2 Comparing the acquired and stored configurations prior to a DL-subnetwork error

J.2.1 General

After the current configuration has been acquired, the data of two configurations is stored in the master: the stored configuration, and the currently acquired configuration. Then these two configurations are compared. The example stops the configuration after the first error has been detected. The comparison may provide the following results:

- No error has been detected, that is, the two configurations are identical.
- No current configuration could be detected, that is, the second configuration list is empty.
- The configuration became longer.
- The configuration became shorter.
- A ring segment became longer.
- A ring segment became shorter.

— At a certain DL-subnetwork position there is a slave with another device code.

A comparison is carried out by the Check_Configuration function.

J.2.2 Check_Configuration function

The Check_Configuration function is described with the Check_Configuration.request and Check_Configuration.confirm primitives.

The Check_Configuration.request primitive has no parameters. Besides the result (+ or -), the confirmation contains in the event of an error an error_code and an add_code with the error position in the ring (see Table J.4).

Table J.4 – Check_Configuration

Parameter name	Request	Confirm
Argument	M	
Result (+)		S
Result (-)		S
error_code		M
slave_position		M

result (+):

No error has been detected, that is, the two configurations are identical.

result (-):

An error has been detected, that is, the two configurations are not identical.

error_code:

This parameter describes the type of error. Possible errors are the respective meanings of the slave_position:

- No configuration available; slave_position does not have a meaning.
- The configuration became shorter; slave_position indicates the first missing slave.
- The configuration became larger; slave_position indicates the first additional slave.
- Additional slave to W1 of slave_position
- The slave with the slave_position number is missing.
- The slave with the slave_position number is a slave with an incorrect device code.

slave_position:

Slave_position contains the error position in the ring.

J.2.3 Compare_Slave function

The Compare_Slave function compares the device code and the ring segment level of two slaves and returns the result of the comparison in the result and error_code parameters (see Table J.5).

Table J.5 – Compare_Slave

Parameter name	Request	Confirm
Argument	M	
saved_data	M	
current_data	M	
Result (+)		S
Result (-)		S
error_code		M

saved_data:

This parameter contains the ID code and the ring segment level of a slave which is stored in the master's configuration.

current_data:

This parameter contains the ID code and the ring segment level of a slave of the currently acquired configuration.

result (+):

The two slaves have the same ID code and ring segment level.

result (-):

The two slaves have different ID codes and/or ring segment levels.

error_code:

Error_code describes the type of distinction. The following is possible:

- The ring segment level of the slave in the currently acquired configuration is higher.
- The ring segment level is lower.
- The ID codes are different.

A ring segment level which is too high or too low has a higher priority than an invalid ID code. Thus, error_code gives no information on the ID code when the ring segment level is incorrect. However, if a wrong ID code is reported, the ring segment level is definitely okay.

J.2.4 State Machine for Comparing the Configuration Data**J.2.4.1 General**

Figure J.3 shows the state machine for comparing two configurations.

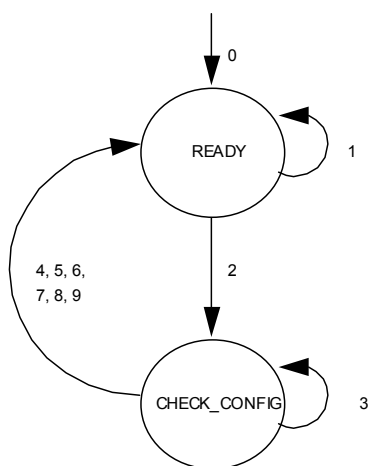


Figure J.3 – State machine for comparing two configurations

J.2.4.2 States of the state machine

READY

The state machine is ready to execute the Check_Config function.

CHECK_CONFIG

The configurations are being compared by means of the Check_Config function.

Table J.6 describes the state transitions.

Table J.6 – State transitions of the state machine for comparing two configurations

Initial state event \condition ⇒ action	Transition	Follow-up state
Power_On	0	READY
READY Check_Config.request \quantity of current identified slaves == 0 ⇒ Check_Config.confirm (-) with error_code = 'no configuration available'	1	READY
READY Check_Config.request \quantity of identified slaves != 0 ⇒ slave_no = 1, m = slave quantity in current configuration, n = slave quantity in stored configuration, Compare_Slave.request(saved_config.[slave_no], current_config.[slave_no])	2	CHECK_CONFIG
CHECK_CONFIG Compare_Slave.confirm (+) \slave_no < n AND slave_no < m ⇒ slave_no++, Compare_Slave.request(saved_config.[slave_no], current_config.[slave_no])	3	CHECK_CONFIG
CHECK_CONFIG Compare_Slave.confirm (+) \slave_no == n AND slave_no == m ⇒ Check_Config.confirm (+)	4	READY
CHECK_CONFIG Compare_Slave.confirm (+) \slave_no < n AND slave_no == m ⇒ Check_Config.confirm (-) with error_code = 'the configuration became shorter' and slave_position = slave_no + 1	5	READY
CHECK_CONFIG Compare_Slave.confirm (+) \slave_no == n AND slave_no < m ⇒ Check_Config.confirm (-) with error_code = 'the configuration became longer' and slave_position = slave_no + 1	6	READY
CHECK_CONFIG Compare_Slave.confirm (-) \error_code == 'ring segment level higher than expected' ⇒ Check_Config.confirm (-) with error_code = 'additional slave at W1 of slave_ - position' and slave_position = slave_no - 1	7	READY
CHECK_CONFIG Compare_Slave.confirm (-) \error_code == 'ring segment level lower than expected' ⇒ Check_Config.confirm (-) with error_code = 'slave missing' and slave_position = slave_no	8	READY
CHECK_CONFIG Compare_Slave.confirm (-) \error_code == 'different ID codes' ⇒ Check_Config.confirm (-) with error_code = 'wrong slave' and slave_position = slave_no	9	READY

J.2.4.3 State Machine for Comparing One Line of Two Configuration Matrices

Figure J.4 shows the state machine for comparing one line of two configuration matrices.

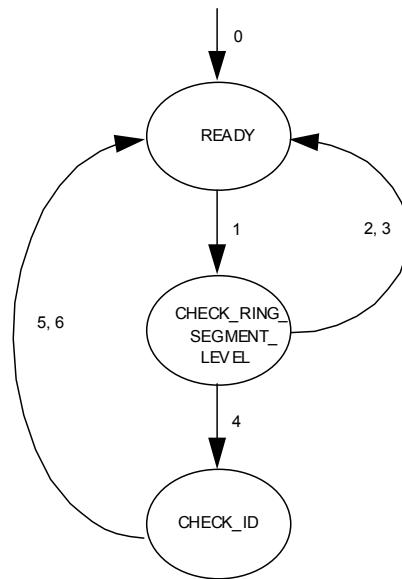


Figure J.4 – State machine for comparing one line of two configuration matrices

J.2.4.4 States of the state machine

READY

The state machine is ready to execute the Compare_Slave function.

CHECK_RING_SEGMENT_LEVEL

The Compare_Slave function was called. The ring segment levels are compared.

CHECK_ID

After the comparison of the ring segment levels the ID codes are compared as well. This comparison only takes place when the ring segment levels are identical.

Table J.7 describes the state transitions.

Table J.7 – State transitions of the state machine for comparing one line of two configuration matrixes

Initial state event condition ⇒ action	Transition	Follow-up state
Power on	0	READY
READY Compare_Slave.request ⇒ compare ring segment levels	1	CHECK_RING_SEGMENT_LEVEL
CHECK_RING_SEGMENT_LEVEL ring segment level higher than expected ⇒ Compare_Slave.confirm (-) with error_code = 'ring segment level higher than expected'	2	READY
CHECK_RING_SEGMENT_LEVEL ring segment level lower than expected ⇒ Compare_Slave.confirm (-) with error_code = 'ring segment level lower than expected'	3	READY
CHECK_RING_SEGMENT_LEVEL ring segment level are not identical ⇒ compare the ID codes	4	CHECK_ID
CHECK_ID ID codes are identical ⇒ Compare_Slave (+)	5	READY
CHECK_ID ID codes are not identical ⇒ Compare_Slave (-) with error_code = 'different ID codes'	6	READY

BIBLIOGRAPHY

IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems*

IEC 60847:1989, *Characteristics of LANs*

IEC 60870-5-1:1990, *Telecontrol equipment and systems – Part 5-1: Transmission protocols – Transmission frame formats*¹⁵

IEC 60955:1989, *Process data highway, Type C (PROWAY C), for distributed process control systems*

IEC 61131-2:1992, *Programmable controllers – Part 2: Equipment requirements and tests*

ISO/IEC 1177:1985, *Information processing – Character structure for start/stop and synchronous character oriented transmission*

ISO/IEC 2022:1994, *Information technology – Character code structure and extension techniques*

ISO/IEC 3309:1993, *Information technology – Telecommunications and information exchange between systems – High-level data link control (HDLC) procedures – Frame structure.*

ISO/IEC 8802 (all parts), *Information technology – Telecommunications and information exchange between systems – Local and Metropolitan area networks*

ISO/IEC TR 8802-1:1997, *Specific requirements – Part 1: Overview of Local Area Network Standards*

ISO/IEC 8802-2:1998, *Specific requirements – Part 2: Logical link control*

ISO/IEC 8802-3:1996, *Specific requirements – Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications*

ISO/IEC 8802-5:1998, *Specific requirements – Part 5: Token ring access method and physical layer specifications*

ISO/IEC 8802-4:1990, *Information processing systems – Local area networks – Part 4: Token-passing bus access method and physical layer specifications*

ISO/IEC 9314-2:1989, *Information processing systems – Fibre Distributed Data Interface (FDDI) – Part 2: Token Ring Media Access Control (MAC)*

ISO/IEC 9646, *Information technology – Open Systems Interconnection – Conformance testing methodology and framework*

ISO/IEC 9646-1:1994, *Part 1: General concepts*

ISO/IEC 9646-2:1994, *Part 2: Abstract test suite specification*

ISO/IEC 10646-1:1993, *Information technology – Universal Multiple-Octet Coded Character Set (UCS) – Part 1: Architecture and Basic Multilingual Plane*

¹⁵ This publication has been deleted in the IEC and is no longer available.

ISO/IEC 15802-1:1995, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common specifications – Part 1: Medium Access Control (MAC) service definition*

ISO 8509:1987, *Information processing systems – Open System Interconnection – Service Conventions*

ITU-T V.41, *Code-independent error-control system*.

ANSI X3.66:1979 (R1990), *Advanced data communication control procedures (ADCCP)*

ANSI X3.159-1989, *Information Systems – Programming Language C*

ANSI X3J16 / ISO WG21 *committee draft working paper for a C++ standard*

IEEE Std 1451.1:1996, *Smart Transducer Interface for Sensors and Actuators – Network Capable Application Processor (NCAP) Information Model*

IEEE Std 1451.2:1997, *Smart Transducer Interface for Sensors and Actuators – Transducer to Microprocessor Communication Protocols and Transducer Electronic Data Sheets (TEDS) Formats*

Internet Engineering Task Force (IETF), *Request for Comments (RFC)*:

RFC 791:1981, *Internet Protocol*

RFC 793:1981, *Transmission Control Protocol*

RFC 1213:1991, *Management Information Base for Network Management of TCP/IP-based internets : MIB-II*

RFC 1643:1994, *Definitions of Managed Objects for the Ethernet-like Interface Types*

IETF <draft-ietf-dhc-fqdn-option-01.txt>: March 2, 2001, *The DHCP Client FQDN Option*

INDEX

- Addressing. *See* 6.3
 - service access points
 - single (DLSAP-address), **38**
 - single or group of (DL(SAP)-address), **38**
- Bridges
 - Functionality, 40, 88, 89, 90, 114, 115, 216, 220, 275, 323, 324
 - Requirements, 98, 117, 129, 134, 179, 217, 231, 232, 236, 244, 248, 303, 322, 394, 397, 400, 413
 - DLPDU modification, 87, 132
 - Reserved addresses, 105, 107, 108
- Connectionless service
 - Data exchange. *See* 8.5–8.7, 9.4.1, 10.3.3
 - Listener query. *See* 10.3.4
 - remote-acknowledged Data transfer. *See* 8.7, 10.3.1
- Connection-oriented service
 - Connection establishment. *See* 8.1, 9.1, 10.2.1
 - Connection release. *See* 8.2, 9.2, 10.2.1
 - Reset. *See* 8.3, 9.3, 10.2.2
 - State transition diagram for a DLCEP, **247**, 940
 - Subscriber query. *See* 10.2.3
 - variables and timers. *See* 6.7.4
- delocalization, 133
- Delocalization, **39**, **134**
- DL(SAP)-address. *See* Addressing, service access points, single or group
- DLE initialization, **347–49**
 - State transition diagram, **347**, 932
- DL-management information. *See* clause 12.4.3.1
- DL-paths, 88, 89, 221, 249, 275, 384, 396, 427, 959
- DLSAP, **37**
- DLSAP-address. *See* Addressing, service access points, single
- DL-time, 41, 43, 119, 237. *also* 8.9–8.12, 9.6–9.8, 10.4.1
- Fractional Duty Cycle (FDC), **44**, 90, 115, 127, 216, 317, 323, 326
 - Requirements, 244, 248, 249, 313, 316, 320, 341, 346, 398
- Link
 - extended, 38
 - local, 37
- Link Active Scheduler (LAS), **41**. *See also* Token, Scheduler token
 - Functionality, 89, 115
 - Requirements, 126–29, 137, 143, 145, 147, 154, 162, 170, 178, 180, 182, 183, 184, 186, 188, 189–90, 192, 194–97, 203–5, 209, 212, 213–15, 216, 218, 232, 264, 301, 304, 310, 345, 410, 414. *also* *clause 11*
 - Reserved addresses, 108
 - timers, 45
- Link Master (LM), **42**
 - Functionality, 90, 115
 - Requirements, 126–29, 211, 215, 345, 348, 359, 360, 361, 362, 363, 394, 398, 414
 - timers, 42
- Profiles, implementation. *See* clause 12.6
- Quality of Service (QoS)
 - DLCEP class, **91**
 - DLCEP data delivery features, **91**
 - DLPDU authentication, **92**
 - Maximum confirm delay, **92–93**
 - Maximum DLSDU size, **93–94**
 - Priority, **91–92**
 - Queuing policy, **94–96**
 - Remote DLE confirmed, **97**
 - Residual activity, **92**
 - Scheduling, **93**
 - Timeliness, **44**, **96–97**
- Redundancy
 - within the Data Link Layer, 88, 275, 384. *See also* Bridges *and* DL-paths
- Scheduling service
 - Compel service. *See* 10.4.2
 - Implicit scheduling. *See* 10.4.3
 - Sequence scheduling. *See* 10.4.3

Sequence subsetting. *See* 10.4.4

SPDU

Definitions. *See* 11.3

Requirements. *See* 12.3, 11.4

SPDUs. *See* clause 11

Timeliness, 245, 246–47. *See also* Quality of Service (QoS)

Timers

DLCEP related

usage of, 266, 272, 274, 277, 279, 280–82, 284, 286, 287, 288

DLCEP retransmission request monitor, **125**

DLCEP sent segments monitor, **123**

Immediate response recovery delay

usage of. tx_irrd_timer_usage

Immediate response recovery delay monitor, **118**

Implementation considerations, 44

Maximum confirm delay

usage of, 251, 253, 255, 256, 257, 258, 259, 260–61, 262, 265, 268, 269, 270, 274, 279–80, 283, 284, 285, 286, 288, 289, 290, 291, 292, 293, 296, 298, 299

Maximum confirm delay monitor, **120**

Node time counter, **119**

Node timer, **43**

update, 301

usage of, 184

Remaining duration counter, **117**

Residual activity monitor, **125**

Residual activity stimulus, **125**

Slot time, **43**, **116**

usage of, 190

Time distribution period

usage of, 301, 304

Time distribution period monitor, **120**

User response delay

usage of, 253–56, 258

Token

Delegated token, 39, 44

creation of, 41, 139, 192, 201

usage of, 41, 42, 89, 131, 135–37, 139, 141, 142, 145, 147, 150, 156, 164, 173, 174, 175, 182, 185, 217, 218, 310

Dominant token, **39**, 135, 139, 141

Reply token, 39, 45

assumption of, 190

creation of, 89, 131, 139, 187, 188

usage of, 41, 42, 89, 118, 139, 141, 148, 154, 162, 173, 174, 175, 177, 178, 183, 236, 249

Scheduler token, 44

assumption of, 137, 139, 143, 145, 178, 180, 182, 184, 188, 192, 197, 205, 209, 210, 212, 215, 218, 348, **353**

dropping of, 192, 201, 208, 212, 214, 215

retention of, 214, 215

usage of, 41, 117, 135–37, 139, 141, 142, 145, 147, 150, 155, 156, 164, 173, 175, 183, 184, 185, 189, 190, 191, 194, 195, 203, 212, 216, 218, **351–52**

Type 1

Addressing

connection endpoint (DLCEP), **39**

scheduling endpoint (DLSEP), **40**

Bridges, **39**

DLCEP-address. *See* Addressing, connection endpoint

DL-Connection (DLC)

multi-peer, **42**

peer, **43**

DLSEP-address. *See* Addressing, scheduling endpoint

Paths, 40

Type 2

DLPDU, **46**

fixed tag, **47**

generic tag, **47**

guardband, **47**

moderator, **48**

Type 6
Bridges, **39**
DL-Connection (DLC)
multi-peer, **56**
peer, **43, 56**



Standards Survey

The IEC would like to offer you the best quality standards possible. To make sure that we continue to meet your needs, your feedback is essential. Would you please take a minute to answer the questions overleaf and fax them to us at +41 22 919 03 00 or mail them to the address below. Thank you!

Customer Service Centre (CSC)

International Electrotechnical Commission

3, rue de Varembe
1211 Genève 20
Switzerland

or

Fax to: **IEC/CSC** at +41 22 919 03 00

Thank you for your contribution to the standards-making process.

A Prioritaire

Nicht frankieren
Ne pas affranchir



Non affrancare
No stamp required

RÉPONSE PAYÉE

SUISSE

Customer Service Centre (CSC)
International Electrotechnical Commission
3, rue de Varembe
1211 GENEVA 20
Switzerland



Q1 Please report on **ONE STANDARD** and **ONE STANDARD ONLY**. Enter the exact number of the standard: (e.g. 60601-1-1)

.....

Q2 Please tell us in what capacity(ies) you bought the standard (tick all that apply). I am the/a:

- purchasing agent ☐
 librarian ☐
 researcher ☐
 design engineer ☐
 safety engineer ☐
 testing engineer ☐
 marketing specialist ☐
 other.....

Q3 I work for/in/as a:
(tick all that apply)

- manufacturing ☐
 consultant ☐
 government ☐
 test/certification facility ☐
 public utility ☐
 education ☐
 military ☐
 other.....

Q4 This standard will be used for:
(tick all that apply)

- general reference ☐
 product research ☐
 product design/development ☐
 specifications ☐
 tenders ☐
 quality assessment ☐
 certification ☐
 technical documentation ☐
 thesis ☐
 manufacturing ☐
 other.....

Q5 This standard meets my needs:
(tick one)

- not at all ☐
 nearly ☐
 fairly well ☐
 exactly ☐

Q6 If you ticked NOT AT ALL in Question 5 the reason is: (tick all that apply)

- standard is out of date ☐
 standard is incomplete ☐
 standard is too academic ☐
 standard is too superficial ☐
 title is misleading ☐
 I made the wrong choice ☐
 other

Q7 Please assess the standard in the following categories, using the numbers:

- (1) unacceptable,
 (2) below average,
 (3) average,
 (4) above average,
 (5) exceptional,
 (6) not applicable

- timeliness.....
 quality of writing.....
 technical contents.....
 logic of arrangement of contents
 tables, charts, graphs, figures.....
 other

Q8 I read/use the: (tick one)

- French text only ☐
 English text only ☐
 both English and French texts ☐

Q9 Please share any comment on any aspect of the IEC that you would like us to know:

.....



ISBN 2-8318-6972-2



ICS 25.040; 35.100; 35.240.50
