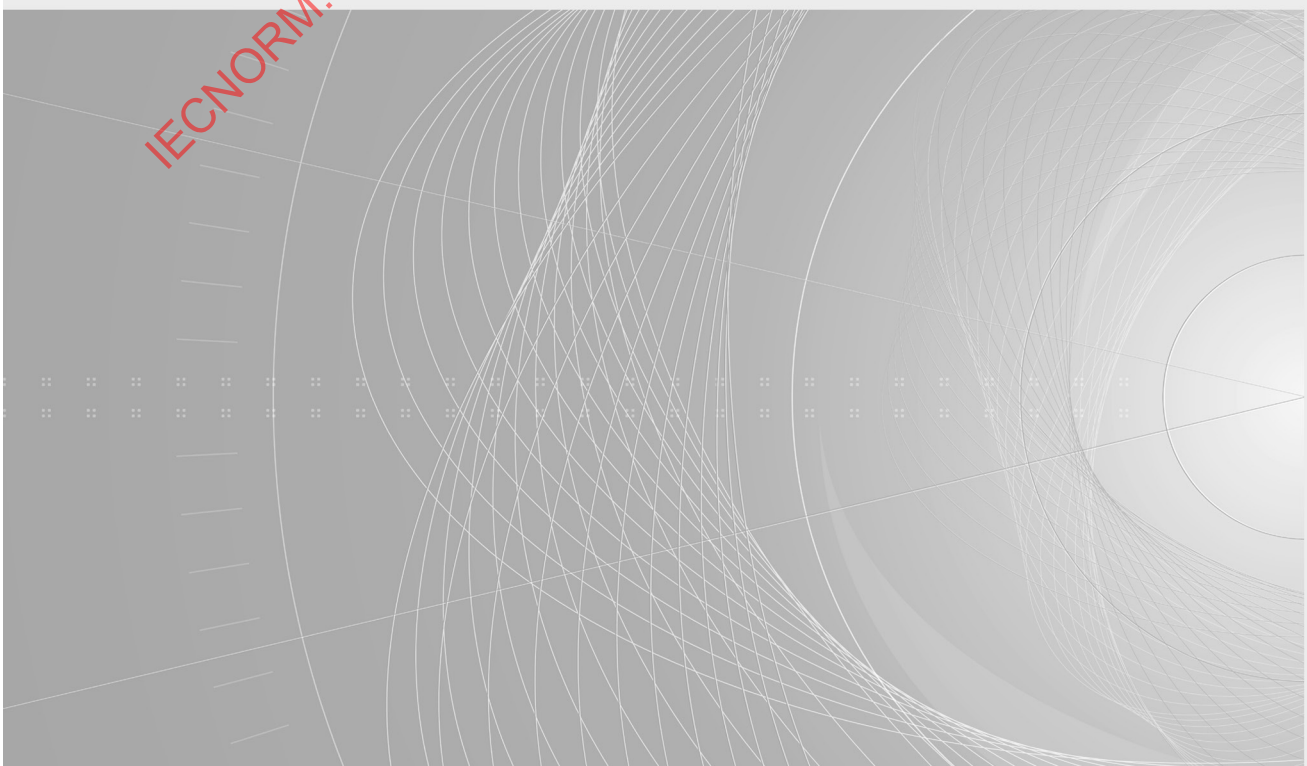
IEC 61158-4-21

Edition 3.0 2023-03

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 4-21: Data-link layer protocol specification – Type 21 elements



IEC 61158-4-21:2023-03(en)

IEC Secretariat
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

About the IEC
The International Electrotechnical Commission (IEC) is the leading global organization that prepares and publishes International Standards for all electrical, electronic and related technologies.

About IEC publications
The technical content of IEC publications is kept under constant review by the IEC. Please make sure that you have the latest edition, a corrigendum or an amendment might have been published.

IEC publications search - webstore.iec.ch/advsearchform
The advanced search enables to find IEC publications by a variety of criteria (reference number, text, technical committee, …). It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished
Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc
If you wish to give us your feedback on this publication or need further assistance, please contact the Customer Service Centre: sales@iec.ch.

IEC Products & Services Portal - products.iec.ch
Discover our powerful search engine and read freely all the publications previews. With a subscription you will always have access to up to date content tailored to your needs.

Electropedia - www.electropedia.org
The world's leading online dictionary on electrotechnology, containing more than 22 300 terminological entries in English and French, with equivalent terms in 19 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC 61158-4-21

Edition 3.0 2023-03

INTERNATIONAL STANDARD

Industrial communication networks – Fieldbus specifications – Part 4-21: Data-link layer protocol specification – Type 21 elements

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.20; 35.110

ISBN 978-2-8322-6555-0

Warning! Make sure that you obtained this publication from an authorized distributor.

CONTENTS

FOREWORD 7
INTRODUCTION 9
1 Scope 10
1.1 General 10
1.2 Specifications 10
1.3 Procedures 10
1.4 Applicability 10
1.5 Conformance 10
2 Normative references 11
3 Terms, definitions, symbols and abbreviated terms 11
3.1 Reference model terms and definitions 11
3.2 Service convention terms and definitions 13
3.3 Common terms and definitions 14
3.4 Additional Type 21 definitions 17
3.5 Common symbols and abbreviations 17
3.6 Additional Type 21 symbols and abbreviations 18
4 Overview of the data-link protocol 18
4.1 General 18
4.2 Overview of medium access control 19
4.3 Service assumed from the physical layer 19
4.4 DLL architecture 20
4.4.1 General 20
4.4.2 DLL management (DLM) interface support function 21
4.5 Data type 22
4.5.1 General 22
4.5.2 Boolean 22
4.5.3 Unsigned integer 22
4.5.4 Signed integer 22
4.5.5 Octet String 23
4.5.6 Visible String 23
4.5.7 Time of day 23
4.6 Local parameters and variables 24
4.6.1 General 24
4.6.2 DLE configuration parameters 24
4.6.3 Queues to support data transfer 25
4.6.4 Variables to support SAP management 25
4.6.5 Variables to support local device information management 27
4.6.6 Variables and counter to support network information management 31
4.6.7 Variables and counter to support a device path information management 35
4.6.8 Variables, counters, timers, and queues to support path table management 39
5 General structure and encoding 39
5.1 Overview 39
5.2 MAPDU structure and encoding 39
5.3 Common MAC frame structure, encoding and elements of procedure 40
5.3.1 MAC frame structure 40

5.3.2 Elements of the MAC frame 40
5.3.3 Elements of the Type 21 DLPDU 41
5.4 Order of bit transmission 49
5.5 Invalid DLPDU 49
6 DLPDU structure and procedure 49
6.1 General 49
6.2 Common DLPDU Field 49
6.2.1 General 49
6.2.2 Version 50
6.2.3 Length 50
6.3 DL-DATA Transfer 50
6.3.1 DT DLPDU 50
6.4 DL-SPDATA Transfer 53
6.4.1 SPDT DLPDU 53
6.5 Network control messages 55
6.5.1 General 55
6.5.2 NCM_LA DLPDU 55
6.5.3 NCM_AT DLPDU 56
6.5.4 NCM_LS DLPDU 58
6.5.5 NCM_RS DLPDU 59
6.5.6 NCM_AR_DLPDU 60
6.5.7 NCM_AR DLPDU structure 60
7 DLE elements of procedure 61
7.1 Overall structure 61
7.2 DL-protocol machine (DLPM) 61
7.2.1 Overview 61
7.2.2 Primitive definitions 61
7.2.3 DLPM state table 65
7.2.4 DLPM functions 68
7.3 DLL management Protocol 70
7.3.1 Overview 70
7.3.2 Primitive definitions 70
7.3.3 DLM state table 73
7.3.4 DLM functions 97
8 Constants and error codes 106
8.1 General 106
8.2 Constants 106
8.3 Data-link layer error codes 108
Bibliography 109

Figure 1 – Interaction of PhS primitives with DLE 19
Figure 2 – Data-link layer architecture 21
Figure 3 – Relationships of DLSAPs, DLSAP-addresses, and group DL-addresses 26
Figure 4 – Common MAC frame format for Type 21 DLPDU 40
Figure 5 – MAC frame format for other protocols 40
Figure 6 – Version and Length field 41
Figure 7 – DST_addr field 42

Figure 8 – SRC_addr field 43
Figure 9 – Frame Control Field 43
Figure 10 – Extension field 46
Figure 11 – DSAP field 47
Figure 12 – Source service access point field 48
Figure 13 – Length of group mask and extension information 48
Figure 14 – Group mask option field 48
Figure 15 – Common DLPDU field 50
Figure 16 – Building a DT DLPDU 50
Figure 17 – DT DLPDU structure 50
Figure 18 – SPDT DLPDU structure 54
Figure 19 – NCM_LA DLPDU structure 55
Figure 20 – DLL structure and elements 61
Figure 21 – State transition diagram of the DLPM 65
Figure 22 – State transition diagram of DLM 74

Table 1 – DLL components 20
Table 2 – UNSIGNED*n* data type 22
Table 3 – INTEGER*n* data type 23
Table 4 – DLE configuration parameters 24
Table 5 – Queues to support data transfer 25
Table 6 – Variables to support SAP management 26
Table 7 – Variables to support device information management 27
Table 8 – DL–entity identifier 27
Table 9 – Device flags 28
Table 10 – DLM state 28
Table 11 – Device Unique Identification 28
Table 12 – Unique identification of device connected to R-port1 29
Table 13 – Unique identification of device connected to R-port2 29
Table 14 – MAC address 29
Table 15 – Port information 30
Table 16 – Protocol version 30
Table 17 – Device type 31
Table 18 – Device description 31
Table 19 – Hop count 31
Table 20 – Variables to support managing network information 32
Table 21 – Topology 32
Table 22 – Collision count 32
Table 23 – Device count 33
Table 24 – Topology change count 33
Table 25 – Last topology change time 33
Table 26 – RNMP device UID 33
Table 27 – RNMS device UID 34

Table 28 – LNM device UID for R-port1 34
Table 29 – LNM device UID for R-port2 34
Table 30 – Network flags 35
Table 31 – Variables and counter to support managing path information 36
Table 32 – Hop count for R-port1 direction 36
Table 33 – Hop count for R-port2 direction 37
Table 34 – Preferred R-port 37
Table 35 – Destination R-port 37
Table 36 – In net count 38
Table 37 – In net time 38
Table 38 – Out net count 39
Table 39 – Out net time 39
Table 40 – Version and Length 42
Table 41 – Destination DL–entity identifier 42
Table 42 – Source DL–entity identifier 43
Table 43 – Frame control 44
Table 44 – Extension 47
Table 45 – Destination service access point 47
Table 46 – source service access point 48
Table 47 – DT DLPDU parameters 51
Table 48 – Primitives exchanged between DLS-user and DLE to send a DT DLPDU 52
Table 49 – Primitives exchanged between DLS-user and DLEs to receive a DT DLPDU 53
Table 50 – SPDT DLPDU Parameters 54
Table 51 – Primitive exchanged between DLS-User and DLEs to send an SPDT DLPDU 54
Table 52 – Primitives exchanged between DLS-user and DLEs to receive an SPDT DLPDU 55
Table 53 – NCM_LA DLPDU parameters 56
Table 54 – NCM_AT DLPDU parameters 57
Table 55 – NCM_LS DLPDU parameters 58
Table 56 – NCM_RS DLPDU parameters 59
Table 57 – NCM_AR DLPDU parameters 60
Table 58 – Primitives exchanged between DLPM and DLS-user 62
Table 59 – Parameters exchanged between DLPM and DLS-user 63
Table 60 – Primitives exchanged between DLPM and DLM 64
Table 61 – Parameters used with primitives exchanged between DLPM and DLM 65
Table 62 – DLPM state table 66
Table 63 – DLPM functions table 69
Table 64 – Primitives exchanged between DLM and DLS-user 71
Table 65 – Parameters used with primitives exchanged between DLM and DLS-user 72
Table 66 – Primitive exchanged between DLM and DMAC 72
Table 67 – Parameters used with primitives exchanged between DLM and DMAC 73
Table 68 – Primitive exchanged between DLM and DPHY 73
Table 69 – Parameters used with primitives exchanged between DLM and DPHY 73

Table 70 – DLM state table 75
Table 71 – DLM function table 97
Table 72 – DLL constants 107
Table 73 – Type 21 DLL error codes 108

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-21: Data-link layer protocol specification – Type 21 elements

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of 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, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). 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. 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 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 IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

Attention is drawn to the fact that the use of the associated protocol type is restricted by its intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a layer protocol type to be used with other layer protocols of the same type, or in other type combinations explicitly authorized by its intellectual-property-right holders.

NOTE Combinations of protocol types are specified in the IEC 61784-1 series and the IEC 61784-2 series.

IEC 61158-4-21 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial process measurement, control and automation. It is an International Standard.

This third edition cancels and replaces the second edition published in 2019. This edition constitutes a technical revision.

This edition includes the following significant technical changes with respect to the previous edition:

a) changed Table 9, Table 15, Table 30, Table 34, Table 35, Table 43, Table 47, Table 48, Table 49, Table 58, and Table 61;

b) changed Network Control Message Type;

c) miscellaneous editorial corrections.

The text of this International Standard is based on the following documents:

Draft	Report on voting
65C/1202/FDIS	65C/1243/RVD

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

A list of all parts of the IEC 61158 series, published under the general title *Industrial communication networks – Fieldbus specifications*, can be found on the IEC web site.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

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

INTRODUCTION

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

The data-link protocol provides the data-link service by making use of the services available from the physical layer. The primary aim of this document 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 document is concerned, in particular, with the communication and interworking of sensors, effectors and other automation devices. By using this document together with other standards positioned within the OSI or fieldbus reference models, otherwise incompatible systems could work together in any combination.

The International Electrotechnical Commission (IEC) draws attention to the fact that it is claimed that compliance with this document may involve the use of a patent. IEC takes no position concerning the evidence, validity, and scope of this patent right.

The holder of this patent right has assured IEC that s/he is willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC. Information may be obtained from the patent database available at http://patents.iec.ch.

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 4-21: Data-link layer protocol specification – Type 21 elements

1 Scope

1.1 General

The DLL provides basic time-critical data communications between devices in an automated environment. Type 21 provides priority-based cyclic and acyclic data communication using an internal collision-free, full-duplex dual-port Ethernet switch technology. For wide application in various automation applications, Type 21 does not restrict the cyclic/acyclic scheduling policy in the DLL.

1.2 Specifications

This part of IEC 61158 describes:

a) procedures for the timely transfer of data and control information from one data link user entity to a peer user entity, and among the data link entities forming the distributed data link service provider;

b) procedures for giving communication opportunities based on ISO/IEC/IEEE 8802-3 MAC, with provisions for nodes to be added or removed during normal operation;

c) structure of the fieldbus data link protocol data units (DLPDUs) used for the transfer of data and control information by the protocol of this document, and their representation as physical interface data units.

1.3 Procedures

The procedures are defined in terms of:

a) the interactions between peer data link entities (DLEs) through the exchange of fieldbus DLPDUs;

b) the interactions between a data link 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 layer 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 that support time-critical communications services in the data link layer of the OSI or fieldbus reference models, and that require the ability to interconnect in an open systems interconnection environment. Profiles provide a simple multi-attribute means of summarizing implementation's capabilities, and thus its applicability to various time-deterministic communications needs.

1.5 Conformance

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

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements 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.

NOTE All parts of the IEC 61158 series, as well as the IEC 61784-1 series and the IEC 61784-2 series are maintained simultaneously. Cross-references to these documents within the text therefore refer to the editions as dated in this list of normative references.

IEC 61158-3-21:2019, *Industrial Communication Networks – Fieldbus specifications – Part 3-21: Data-link layer service definition – Type 21 elements*

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

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

ISO/IEC/IEEE 8802-3:2021, *Telecommunications and exchange between information technology systems – Requirements for local and metropolitan area networks – Part 3: Standard for Ethernet*

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

3 Terms, definitions, symbols and abbreviated terms

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

- IEC Electropedia: available at https://www.electropedia.org/
- ISO Online browsing platform: available at https://www.iso.org/obp

3.1 Reference model terms and definitions

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

3.1.1	**called-DL-address**	[ISO/IEC 7498-3]
3.1.2	**calling-DL-address**	[ISO/IEC 7498-3]
3.1.3	**centralized multi-end-point-connection**	[ISO/IEC 7498-1]
3.1.4	**correspondent (*N*)-entities** **correspondent DL-entities (N=2)** **correspondent Ph-entities (N=1)**	[ISO/IEC 7498-1]
3.1.5	**demultiplexing**	[ISO/IEC 7498-1]
3.1.6	**DL-address**	[ISO/IEC 7498-3]
3.1.7	**DL-address-mapping**	[ISO/IEC 7498-1]
3.1.8	**DL-connection**	[ISO/IEC 7498-1]
3.1.9	**DL-connection-end-point**	[ISO/IEC 7498-1]
3.1.10	**DL-connection-end-point-identifier**	[ISO/IEC 7498-1]
3.1.11	**DL-connection-mode transmission**	[ISO/IEC 7498-1]
3.1.12	**DL-connectionless-mode transmission**	[ISO/IEC 7498-1]
3.1.13	**DL-data-sink**	[ISO/IEC 7498-1]
3.1.14	**DL-data-source**	[ISO/IEC 7498-1]
3.1.15	**DL-duplex-transmission**	[ISO/IEC 7498-1]
3.1.16	**DL-facility**	[ISO/IEC 7498-1]
3.1.17	**DL-local-view**	[ISO/IEC 7498-3]
3.1.18	**DL-name**	[ISO/IEC 7498-3]
3.1.19	**DL-protocol**	[ISO/IEC 7498-1]
3.1.20	**DL-protocol-connection-identifier**	[ISO/IEC 7498-1]
3.1.21	**DL-protocol-control-information**	[ISO/IEC 7498-1]
3.1.22	**DL-protocol-data-unit**	[ISO/IEC 7498-1]
3.1.23	**DL-protocol-version-identifier**	[ISO/IEC 7498-1]
3.1.24	**DL-relay**	[ISO/IEC 7498-1]
3.1.25	**DL-service-connection-identifier**	[ISO/IEC 7498-1]
3.1.26	**DL-service-data-unit**	[ISO/IEC 7498-1]
3.1.27	**DL-simplex-transmission**	[ISO/IEC 7498-1]
3.1.28	**DL-subsystem**	[ISO/IEC 7498-1]
3.1.29	**DL-user-data**	[ISO/IEC 7498-1]
3.1.30	**flow control**	[ISO/IEC 7498-1]
3.1.31	**layer-management**	[ISO/IEC 7498-1]
3.1.32	**multiplexing**	[ISO/IEC 7498-3]

3.1.33 naming-(addressing)-authority [ISO/IEC 7498-3]

3.1.34 naming-(addressing)-domain [ISO/IEC 7498-3]

3.1.35 naming-(addressing)-subdomain [ISO/IEC 7498-3]

3.1.36 (N)-entity [ISO/IEC 7498-1]
DL-entity
Ph-entity

3.1.37 (N)-interface-data-unit [ISO/IEC 7498-1]
DL-service-data-unit (N=2)
Ph-interface-data-unit (N=1)

3.1.38 (N)-layer [ISO/IEC 7498-1]
DL-layer (N=2)
Ph-layer (N=1)

3.1.39 (N)-service [ISO/IEC 7498-1]
DL-service (N=2)
Ph-service (N=1)

3.1.40 (N)-service-access-point [ISO/IEC 7498-1]
DL-service-access-point (N=2)
Ph-service-access-point (N=1)

3.1.41 (N)-service-access-point-address [ISO/IEC 7498-1]
DL-service-access-point-address (N=2)
Ph-service-access-point-address (N=1)

3.1.42 peer-entities [ISO/IEC 7498-1]

3.1.43 Ph-interface-control-information [ISO/IEC 7498-1]

3.1.44 Ph-interface-data [ISO/IEC 7498-1]

3.1.45 primitive name [ISO/IEC 7498-3]

3.1.46 reassembling [ISO/IEC 7498-1]

3.1.47 recombining [ISO/IEC 7498-1]

3.1.48 reset [ISO/IEC 7498-1]

3.1.49 responding-DL-address [ISO/IEC 7498-3]

3.1.50 routing [ISO/IEC 7498-1]

3.1.51 segmenting [ISO/IEC 7498-1]

3.1.52 sequencing [ISO/IEC 7498-1]

3.1.53 splitting [ISO/IEC 7498-1]

3.1.54 synonymous name [ISO/IEC 7498-3]

3.1.55 systems-management [ISO/IEC 7498-1]

3.2 Service convention terms and definitions

This document 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-protocol-machine

3.2.11 DL-provider-initiated-facility

3.2.12 DL-provider-optional-facility

3.2.13 DL-service-primitive; primitive

3.2.14 DL-service-provider

3.2.15 DL-service-user

3.2.16 DLS-user-optional-facility

3.2.17 indication (primitive); acceptor.deliver (primitive)

3.2.18 multi-peer

3.2.19 request (primitive); requestor.submit (primitive)

3.2.20 requestor

3.2.21 response (primitive); acceptor.submit (primitive)

3.2.22 submit (primitive)

3.2.23 symmetrical service

3.3 Common terms and definitions

For the purposes of this document, the following definitions also apply.

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

3.3.1
active network
network in which data transmission between non-immediately-connected devices is dependent on active elements within those intervening devices that form the connection path

[SOURCE: IEC 61918:2018, 3.1.3]

3.3.2
DL-segment
link
local link
single data link (DL) subnetwork in which any of the connected data link entities (DLEs) may communicate directly, without any intervening data link 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

3.3.3
data-link service access point
DLSAP
distinctive point at which DL-services are provided by a single DLE to a single higher-layer entity

Note 1 to entry: Definition derived from ISO/IEC 7498-1:1994, Clause 5.

3.3.4
DL(SAP) -address
either an individual DLSAP address designating a single DLSAP of a single data link service (DLS) user (DLS-user), or a group DL-address potentially designating multiple DLSAPs, each of a single DLS-user

Note 1 to entry: This terminology was 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

Note 1 to entry: A single DL-entity may have multiple DLSAP-addresses associated with a single DLSAP.

3.3.6
data-link connection endpoint address
DLCEP-address
DL-address that designates either:

a) one peer DL-connection-end-point;

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

3.3.7
DL–entity identifier
address that designates the (single) DLE associated with a single device on a specific local link

3.3.8
device
single DLE as it appears on one local link

3.3.9
end-station
system attached to a network that is an initial source or a final destination of MAC frames transmitted across that network

Note 1 to entry: A network layer router is, from the perspective of the network, an end-station. A switch, in its role of forwarding MAC frames from one link to another, is not an end-station.

[SOURCE: IEC 61784-2-0, 3.1.3]

3.3.10
frame
unit of data transmission on an ISO/IEC/IEEE 8802-3 MAC (Media Access Control) that conveys a protocol data unit (PDU) between MAC service users

[SOURCE: IEEE Std 802.1Q-2018, Modified − Replaced IEEE Std 802-3 by ISO/IEC/IEEE 8802-3]

3.3.11
frame check sequence (FCS) error
error that occurs when the computed frame check sequence value after reception of all the octets in a data link protocol data unit (DLPDU) does not match the expected residual

3.3.12
linear topology
topology where the nodes are connected in series, with two nodes connected to only one other node and all others each connected to two other nodes (that is, connected in the shape of a line)

Note 1 to entry: This topology corresponds to that of an open ring.

[SOURCE: IEC 61918:2018, 3.1.51]

3.3.13
link
transmission path between two adjacent nodes

3.3.14
network management
management functions and services that perform network initialization, configuration, and error handling

3.3.15
node
network entity connected to one or more links

Note 1 to entry: A node may be either a switch, an end-station or an RTE end-station.

[SOURCE: IEC 61784-2-0, 3.1.10]

3.3.16
packet
logical grouping of information used to describe a unit of data at any layer to convey the upper layer user data to its peer layer

Note 1 to entry: A packet is identical to the PDU at each layer in terms of the OSI reference model. A data-link layer packet is a frame.

[SOURCE: IEC 61784-2-0, 3.1.11]

3.3.17
protocol
convention on the data formats, time sequences, and error correction for data exchange in communication systems

3.3.18
real-time
ability of a system to provide a required result in a bounded time

[SOURCE: IEC 61784-2-0, 3.1.12]

3.3.19
real-time communication
transfer of data in real-time

[SOURCE: IEC 61784-2-0, 3.1.13]

3.3.20
Real-time Ethernet
RTE
ISO/IEC/IEEE 8802-3 based network that includes real-time communication

Note 1 to entry: Other communications can be supported, providing that the real-time communication is not compromised.

Note 2 to entry: This definition is dedicated but not limited to ISO/IEC/IEEE 8802-3. It could be applicable to other IEEE 802 specifications, for example IEEE Std 802.11.

[SOURCE: IEC 61784-2-0, 3.1.14]

3.3.21
ring
active network where each node is connected in series to two other nodes

[SOURCE: IEC 61918:2018, 3.1.71]

3.3.22
RTE end device
device with at least one RTE end-station

[SOURCE: IEC 61784-2-0, 3.1.16]

3.3.23
RTE end-station
end-station with RTE capability

[SOURCE: IEC 61784-2-0, 3.1.17]

3.4 Additional Type 21 definitions

3.4.1
device unique identification
unique 8 octet identification to identify a Type 21 device in a network

Note 1 to entry: This ID is a combination of a 6 octet ISO/IEC/IEEE 8802-3 MAC address and 2 octet DL-address.

3.4.2
R-port
port in a communication device that is part of a ring structure

3.5 Common symbols and abbreviations

DL	data link (used as a prefix or adjective)
DLC	data link connection
DLCEP	data link connection endpoint
DLE	data link entity (the local active instance of the DLL)
DLL	data link layer
DLPDU	data link protocol data unit
DLPM	data link protocol machine
DLM	data link management
DLME	data link management entity (the local active instance of DLM)
DLMS	data link management service
DLS	data link service

DLSAP	data link service-access-point
DLSDU	data link service-data-unit
FIFO	first-in, first-out (queuing method)
NMT	network management
OSI	Open Systems Interconnection
Ph-	physical layer (as a prefix)
PHY	physical interface transceiver
PhL	physical layer
RTE	Real-time Ethernet
IEC	International Electrotechnical Commission
IP	Internet Protocol (see IETF RFC 791)
ISO	International Organization for Standardization
MAC	media access control
NRT	non-real-time
PDU	protocol data unit
SAP	service access point
RT	real-time
TCP	Transmission Control Protocol (see IETF RFC 793)
UDP	User Datagram Protocol (see IETF RFC 768)

3.6 Additional Type 21 symbols and abbreviations

EFR	extremely fast recovery
GD	general device
LNM	line network manager
PO	power on
PnP	plug and play
RNM	ring network manager
RNMP	primary ring network manager
RNMS	secondary ring network manager
RNAC	ring network auto configuration
UID	device unique identification
Type 21 NMIB	Type 21 network management information base

4 Overview of the data-link protocol

4.1 General

Type 21 extends Ethernet according to the ISO/IEC/IEEE 8802-3 with mechanisms to transfer data with predictable timing demands typical of high-performance automation. It does not change the basic principles of the Ethernet according to ISO/IEC/IEEE 8802-3 but extends it toward RTE. Thus, it is possible to continue to use standard Ethernet hardware, infrastructure components, or test and measurement equipment, such as network analyzers.

4.2 Overview of medium access control

A Type 21 device requires an integrated switch with two ports (ring ports) connected to the ring. A Type 21 network system is constructed with full-duplex, collision-free Ethernet switching devices as a ring or a line network. Type 21 guarantees collision-free data transmission between two devices linked by a full-duplex Ethernet connection. Thus, the Type 21 data link layer provides reliable, transparent and collision-free data transmission among DLS-users.

A Type 21 connection provides collision-free, full-duplex data communication between two devices. Therefore, a Type 21 device can transmit a frame at any time without restriction of media access rights. Type 21 data link layer does not restrict the scheduling method to use for data link resources, but each device can be individually scheduled by the application program so that Type 21 can be applied flexibly in various applications.

A Type 21 frame is delivered to the destination device in one of the following two ways:

- a frame is initiated by a source device and directly transmitted to the neighboring destination device;
- a frame is initiated by a source device and forwarded to the destination device by intermediate devices.

When a frame is forwarded by the intermediate device, the frame forwarding procedure is processed by the internal hardware switch to minimize the processing time, and it does not affect the performance of Type 21 DLL.

4.3 Service assumed from the physical layer

This document describes the assumed physical layer service (PhS) and the constraints used by the DLE. The physical service is assumed to provide the following service primitives specified by ISO/IEC/IEEE 8802-3:2021, Clause 2.

The assumed primitives of PhS are MA-DATA.request and MA-DATA.indication.

The temporal relationship of the primitives is shown in Figure 1.

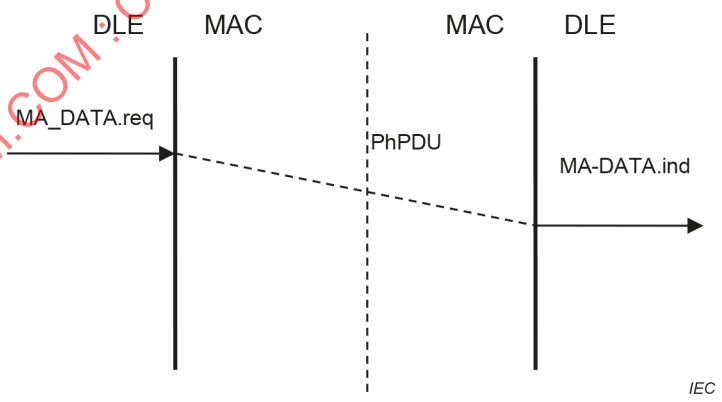


Figure 1 – Interaction of PhS primitives with DLE

The MA-DATA request primitive defines the transfer of data from a MAC client entity to a single peer entity or multiple peer entities in the case of group addresses.

The MA-DATA indication primitive defines the transfer of data from the MAC sublayer entity (through the optional MAC control sublayer, if implemented) to the MAC client entity or entities in the case of group addresses.

4.4 DLL architecture

4.4.1 General

The Type 21 DLL provides reliable and efficient higher-level data transfer service using a full-duplex ring or line topology without any specific access control scheme. The Type 21 DLL is modeled as a combination of control components of the data link protocol machine (DLPM) and the DLL management interface.

The DLPM transmits the data link service data unit (DLSDU) generated by the local DLS-user to the appropriate R-port according to the data link service policy. The DLPM also examines the received frame and delivers the received DLPDU to the appropriate DLS-user.

The DLL management interface provides DLL management functions to maintain the network management information base (NMIB). The NMIB includes local device information, network information and the path table. The DLL is comprised of the components listed in Table 1.

Table 1 – DLL components

Components	Description
DL-protocol machine (DLPM)	Transmits the DLSDU received from local DLS-user through the real-time queue (RT-queue) or non-real-time queue (NRT-queue). Examines the received frame and delivers the DLPDU to the appropriate DLS-user.
DLL management interface (DLM-interface)	Holds the station management variables that belong to the DLL, and manages synchronized changes of the link parameters.

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

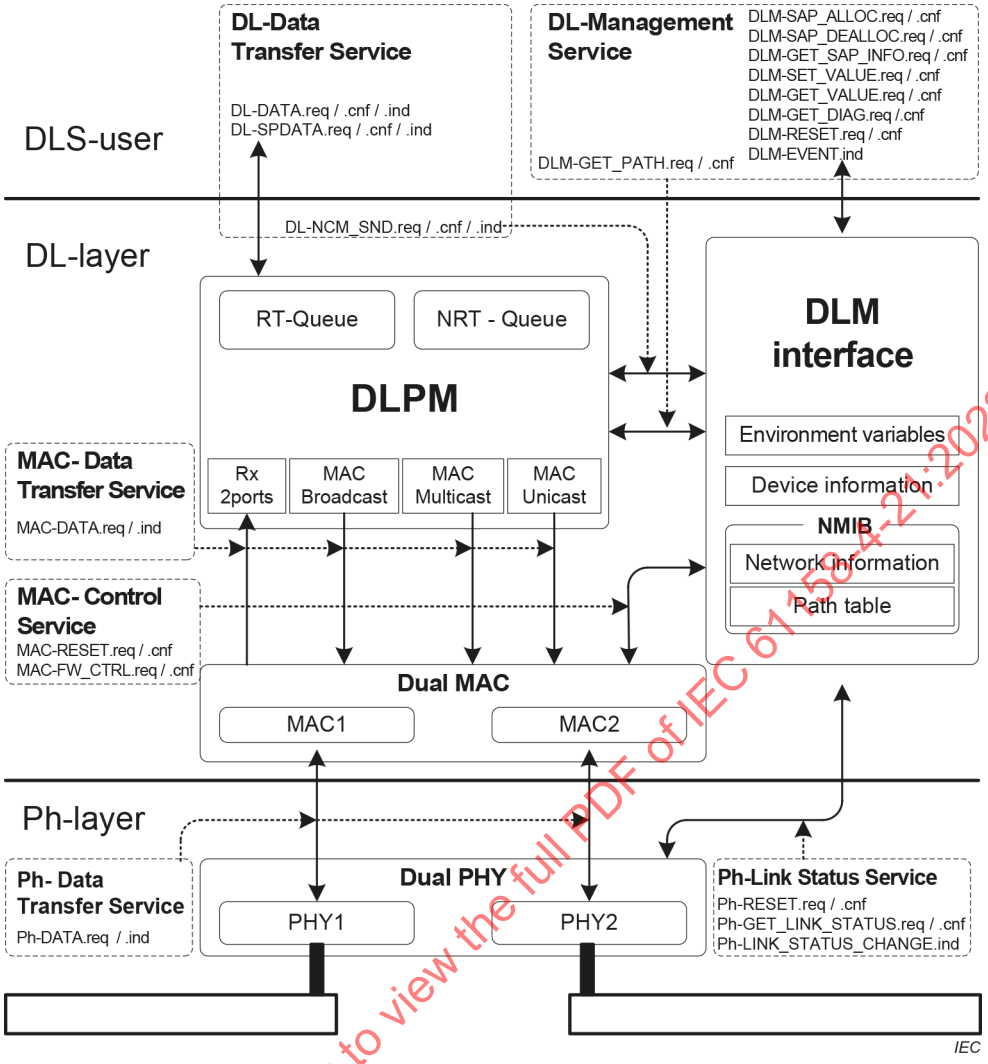


Figure 2 – Data-link layer architecture

4.4.2 DLL management (DLM) interface support function

DLM is one of the key Type 21 features that enables the Plug and Play (PnP), Network Auto Configuration (NAC), and Extremely Fast Recovery (EFR) functions. DLM spontaneously maintains and shares local device information and network-related information with every device on the network.

- Type 21 NMIB management

  The DLM interface maintains the NMIB including the local device information that indicates the physical and logical status of the device, the network information that indicates the current network configuration status, and the path table that indicates the path information to the other devices and their profiles.

- Plug and Play (PnP)

  When a device joins the existing network, it is automatically detected and configured without extra manual configuration of parameters so that the new device can communicate directly with the other devices on the network.

- Network Auto Configuration (NAC)

  Type 21 supports ring and line network topologies. When the network is changed from ring to line or from line to ring topology, the change is automatically detected and broadcast to every device on the network, and then the network information and path table entries are automatically updated.

- Extremely Fast Recovery (EFR)

  When the network is changed from ring to line topology, network information and path information are automatically updated within 10 ms.

- System diagnostic services

  The DLM interface also provides system diagnostic service.

4.5 Data type

4.5.1 General

This document describes the basic Type 21 data types. The data types described in this document are only the normative references to represent Type 21 data type formats. The implementation issues are not covered in this document.

4.5.2 Boolean

Data of basic data type BOOLEAN can have the values TRUE or FALSE. The values are represented as bit sequences of length 1. The value TRUE is represented by a 1 and the value FALSE is represented by a 0.

4.5.3 Unsigned integer

Data of basic data type UNSIGNEDn have values of non-negative integers. The value range is 0,..., 2^{n-1}. The data are represented as bit sequences of length n. The bit sequence

$$b = b_0 \sim b_{n-1}$$

is assigned the value

$$\text{UNSIGNED}n = b_0 2^0 + b_1 2^1 + \cdots + b_{n-1} 2^{n-1}$$

The bit sequence starts on the right with the least significant octet as shown in Table 2.

Table 2 – UNSIGNEDn data type

Octet number	1	2	3	4	5	6	7	8
UNSIGNED8	$b_0..b_7$							
UNSIGNED16	$b_0..b_7$	$b_8..b_{15}$						
UNSIGNED24	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$					
UNSIGNED32	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$				
UNSIGNED40	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$			
UNSIGNED48	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$	$b_{40}..b_{47}$		
UNSIGNED56	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$	$b_{40}..b_{47}$	$b_{48}..b_{55}$	
UNSIGNED64	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$	$b_{40}..b_{47}$	$b_{48}..b_{55}$	$b_{56}..b_{63}$

4.5.4 Signed integer

Data of basic data type INTEGERn has values of integers. The value range is from -2^{n-1} to 2^{n-1}. The data are represented as bit sequences of length n. The bit sequence

$$b = b_0 \sim b_{n-1}$$

is assigned the value

INTEGERn = $b_0 2^0 + b_1 2^1 + \cdots b_{n-2} 2^{n-2}$, when b_{n-1} is 0, and

INTEGERn = -Complement(b) – 1, when b_{n-1} is 0.

The bit sequence starts on the right with the least significant octet as shown in Table 3.

Table 3 – INTEGERn data type

Octet number	1	2	3	4	5	6	7	8
INTEGER8	$b_0..b_7$							
INTEGER16	$b_0..b_7$	$b_8..b_{15}$						
INTEGER24	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$					
INTEGER32	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$				
INTEGER40	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$			
INTEGER48	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$	$b_{40}..b_{47}$		
INTEGER56	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$	$b_{40}..b_{47}$	$b_{48}..b_{55}$	
INTEGER64	$b_0..b_7$	$b_8..b_{15}$	$b_{16}..b_{23}$	$b_{24}..b_{31}$	$b_{32}..b_{39}$	$b_{40}..b_{47}$	$b_{48}..b_{55}$	$b_{56}..b_{63}$

4.5.5 Octet String

The data type OCTET_STRINGn is defined below. The n represents the octet length of the octet string.

ARRAY[n] of UNSINGED8 OCTET_STRINGn

4.5.6 Visible String

The data type VISIBLE_STRINGn is defined below. The admissible values of data of type VISIBLE_CHAR are 0x00 and the range 0x20–0x7E. The data are interpreted as 7-bit coded characters. The n indicates the octet length of the visible string.

UNSINGED8 VISIBLE_CHAR

ARRAY[n] of VISIBLE_CHAR VISIBLE_STRINGn

There is no 0x00 necessary to terminate the string.

4.5.7 Time of day

The data type TIMEOFDAY is defined below. TIMEOFDAY consists of UNSIGNED16 date counting and UNSIGNED32 millisecond fields.

TIMEOFDAY

UNSIGNED32 milliseconds: the count from time 00:00 in milliseconds.

UNSIGNED16 date count: the number of days from 1984–01–01.

4.6 Local parameters and variables

4.6.1 General

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

a) all variables shall be initialized to their default value, or to their minimum permitted value if no default is specified;

b) all counters shall be initialized to zero;

c) all timers shall be initialized to inactive.

DLM can change the values of configuration variables.

Local parameters and variables include DLE configuration parameters, local device information, and NMIB variables. The DLE operational condition is stored in DLE configuration parameters and the DLE configuration parameters are configured locally or remotely according to the application. Local data link information, such as DL–entity identifier and the status of each R-port, are stored in the local device information. NMIB includes network information and the path table. Network topology and the network-related variables are stored in the network information, and the device profile and path information of the other devices on the network are summarized in the path table.

4.6.2 DLE configuration parameters

4.6.2.1 General

These parameters are required to configure the local DLE operation, and they are managed by DLM. Every device on the same network segment has the same DLE configuration parameters. DLM-GET_VALUE service and DLM-SET_VALUE service are used to read or write the DLE configuration parameters. Table 4 shows the list of DLE configuration parameters.

Table 4 – DLE configuration parameters

Parameter	Data type	Default value	Description
Max DL–entity identifier	UNSIGNED16	255	Maximum DL–entity identifier 256 to 65 535: reserved
DLPM scheduling policy	UNSIGNED8	0	0: First-In, First-Out 1: Fixed priority 2: to 255: reserved

4.6.2.2 P(MAX_ADDR): Maximum DL–entity identifier

This variable holds the maximum device address and is set by the DLM. The range of this variable is 1 to 255. The default value of this variable is 255. This variable also indicates the maximum number of path table entries in the NMIB. The values in the range 256 to 65 535 are reserved.

4.6.2.3 P(DLPMSP): data link protocol machine scheduling policy

This variable holds the scheduling policy of the local DLPM, which dictates how to serve the concurrent DLSDUs in the RT-queue. The NRT-queue is not scheduled by the DLPMSP. DLPMSP can have one of the following values:

a) 0: first-in, first-out. The first received frame is served first. In this case, the message priority in the DLSDU is ignored;

b) 1: Fixed priority. The DLSDU with high priority is served first. If many DLSDUs are stacked with the same priority, the frame received first is served first;

c) 2 to 255: reserved.

4.6.3 Queues to support data transfer

4.6.3.1 General

When a DLS-user generates a DLSDU to be transmitted by the DLPM, the DLSDU is encapsulated by the DLPDU. The DLPDU is then stored in the RT-queue or the NRT-queue according to its application and service type. Table 5 shows the queue list for Type 21 data transfer.

Table 5 – Queues to support data transfer

Parameter	Data type	Default value	Description
RT-queue	—	—	Transmit queue to store the real-time data
NRT-queue	—	—	Transmit queue to store the non-real-time data

4.6.3.2 Q(RTQ): RT-queue

The RT-queue stores the real-time DLPDUs generated by a DLS-user to be transmitted by the DLPM. The size of the RT-queue is not restricted in this document and it is considered a detail of the local implementation. The RT-queue is scheduled by the DLPM.

4.6.3.3 Q(NRTQ): NRT-queue

The NRT-queue stores non-real-time DLPDUs generated by a DLS-user. The size of the NRT-queue is not restricted in this document and it is considered a detail of the local implementation. Messages in the RT-queue are transmitted first. Messages in the NRT-queue are transmitted only when the RT-queue is empty.

4.6.4 Variables to support SAP management

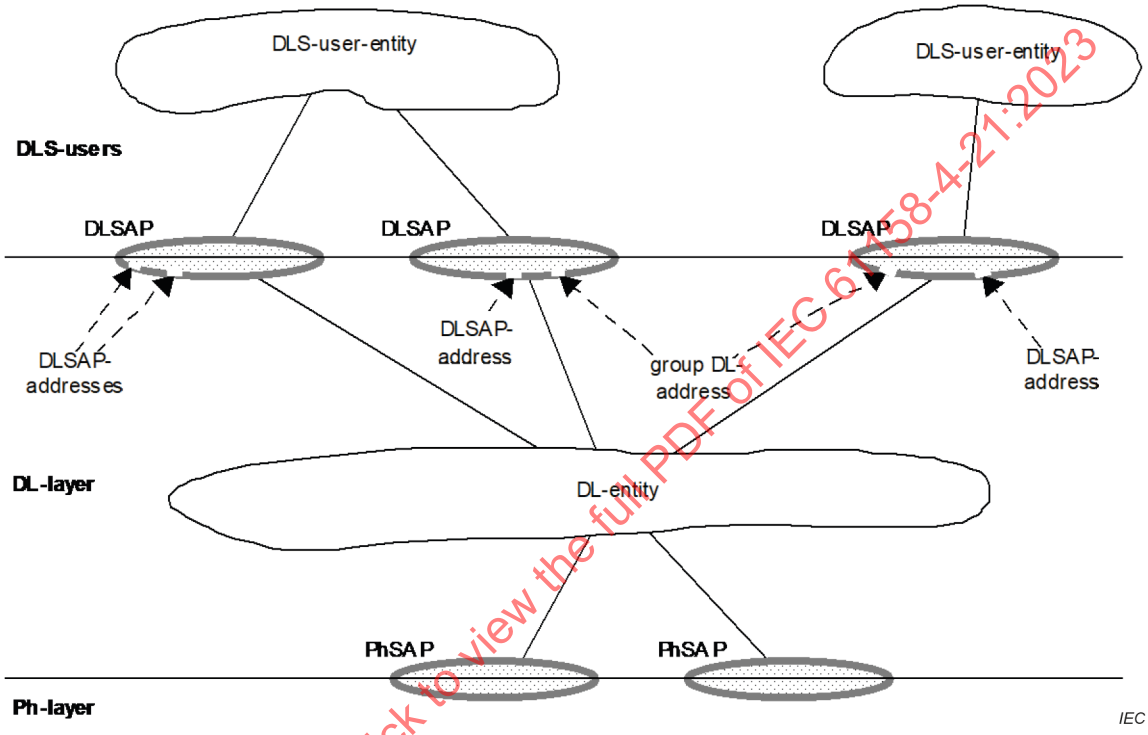
4.6.4.1 General

Allocation and de-allocation of the data link service access point (DLSAP) is managed by the DLM. When a frame is received, the DLPM examines the destination service access point (DSAP) in the DLSDU. If the DSAP is already allocated to a DLS-user, the DLM returns the appropriate DLS-user ID equivalent for the received DSAP address, and the DLPM delivers the received DLSDU to the DLS-user. If the service access point (SAP) is not allocated and no appropriate DLS-user ID is found in the DLM, the received DLSDU is discarded by the DLPM. Therefore, to receive a DLSDU from a certain peer DLS-user, a DLS-user shall first obtain a SAP allocation using the DLM-SAP_ALLOC service. Once the SAP is allocated to a DLS-user, it is used to send and receive data until the SAP is deallocated and returned to the DLM. The deallocated SAP can be used again after reallocation. The SAP address and its appropriate DLS-user ID are stored together and maintained by the DLM. The maximum number of SAP management items is 65 535 but the method to allocate and de-allocate the SAP address is not restricted in this document. Table 6 shows the list of SAP management variables.

Table 6 – Variables to support SAP management

Parameter	Data type	Default value	Description
SAP	UNSIGNED16	—	Service access point
DLS-user ID	UNSIGNED32	—	Numeric identification of the DLS-user that owns the SAP allocation

NOTE The definition of DLSAP, derived from ISO/IEC 7498-1, is explained here to facilitate understanding of the critical distinction between DLSAPs and their DL-addresses (see Figure 3).



NOTE 1 DLSAPs and physical layer service access points (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.

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

A single DLE can have multiple DLSAP-addresses and group DL-addresses associated with a single DLSAP.

4.6.4.2 V(SAP): SAP

This variable holds the SAP. The value for this variable is in the range 0 to 65 535.

4.6.4.3 V(DLS_USER_ID): DLS-user ID

This variable holds the 4-octet numeric identification of the DLS-user who owns the SAP allocation.

4.6.5 Variables to support local device information management

4.6.5.1 General

To maintain the network topology, every device manages a device database including the local device information and the other device information. Table 7 shows the list of device information management variables.

Table 7 – Variables to support device information management

Parameter	Data type	Default value	Description
DL–entity identifier	UNSIGNED16	INVALID_ADDR	Local DL–entity identifier
Device flags	UNSIGNED16	0	Local device flags
Device type	UNSIGNED16	0	Local device type
Hop count	UNSIGNED16	0	Hop count
Device UID	UNSIGNED64	INVALID_UID	Local device unique ID
Device UID for R-port1	UNSIGNED64	INVALID_UID	Device unique ID connected through R-port1
Device UID for R-port2	UNSIGNED64	INVALID_UID	Device unique ID connected through R-port2
MAC address	UNSIGNED48	0	Local device MAC address
Reserved0	UNSIGNED16	0	Reserved
Port information	UNSIGNED16	0	Local device port information
Device state	UNSIGNED8	0	DLM state
Protocol version	UNSIGNED8	0	Local device protocol version
Device description	VISIBLE_STRING[16]	""	Device description string
Reserved1	UNSIGNED32	0	Reserved

4.6.5.2 V(DL_ADDR): DL–entity identifier

This variable holds the DL–entity identifier that designates the (single) DL–entity associated with a single device on a specific local link whose value is constrained to the range 0 to 255. The DL–entity identifier can be provided by hardware settings (for example, rotary switch) or set by software. The DL–entity identifier is defined in Table 8.

Table 8 – DL–entity identifier

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.5.3 V(DEV_FLAG): Device Flag

This variable holds the flags for events that occurred in a local device. When the local DL–entity identifier collision flag is set, EVENT_THIS_ADDR_COLLISION event is generated by the DLM, and then the local DL–entity identifier collision flag is cleared. When the DLM state change flag is set, EVENT_DEV_STATE_CHG is generated by the DLM, and then the DLM state change flag is cleared. Type 21 device flags and event flags are listed in Table 9.

Table 9 – Device flags

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Bit 0	Read/Write	Read	Collision
	Bit 1	Read/Write	Read	Changed
	Bit 2	Read/Write	Read	Frame forward enabled between R-port1 and R-port2
	Bit 3 – Bit 15	—	—	All others reserved

4.6.5.4 V(DLM_STATE): DLM state

This variable holds the DLM state. DLM state is defined as shown in Table 10.

Table 10 – DLM state

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0: INVALID_DLM_STATE 1: standalone state (SA) 2: line network manager state (LNM) 3: general Device state (GD) 4: primary ring network manager state (RNMP) 5: secondary ring network manager state (RNMS) 6 to 255: reserved

4.6.5.5 V(DEV_UID): Device UID

This variable holds the unique 8-octet identification that identifies a Type 21 device in a network. It is a combination of the 6-octet ISO/IEC/IEEE 8802-3 MAC address and the 2-octet DL–entity identifier. The Device UID is defined as shown in Table 11.

Table 11 – Device Unique Identification

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.5.6 V(DEV_UID_RP1): Device UID for R-port1

This variable holds the UID of the device that is linked through the R-port1. The Device UID for R-port1 is defined as shown in Table 12.

Table 12 – Unique identification of device connected to R-port1

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.5.7 V(DEV_UID_RP2): Device UID for R-port2

This variable holds the UID of the device that is linked through the R-port2. The Device-UID for R-port2 is defined as shown in Table 13.

Table 13 – Unique identification of device connected to R-port2

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.5.8 V(MAC_ADDR): MAC address

This variable holds 6-octet ISO/IEC/IEEE 8802-3 Ethernet MAC address of local device. As a Type 21 device has two Ethernet MAC ports, both MAC addresses should be identical. The MAC address is defined as shown in Table 14.

Table 14 – MAC address

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED48	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address

4.6.5.9 V(PORT_INFO): Port information

This variable holds the port information for each R-port. It is defined as shown in Table 15.

Table 15 – Port information

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Bit 0	Read/Write	Read	R-port1 link down
	Bit 1	Read/Write	Read	Received NCM_FAMILY_RES message from R-port1
	Bit 2	Read/Write	Read	A NCM_FAIMLY_RES message has been sent for R-port1
	Bit 3	Read/Write	Read	A NCM_ADV_THIS message has been received from neighbor node of R-port1
	Bit 4	Read/Write	Read	A NCM_MEDIA_LINKED message has been received from neighbor node of R-port1
	Bit 5 – 7	Read/Write	Read	Reserved
	Bit 8	Read/Write	Read	R-port2 link down
	Bit 9	Read/Write	Read	Received NCM_FAMILY_RES message from R-port2
	Bit 10	Read/Write	Read	A NCM_FAMILY_RES message has been sent for R-port2
	Bit 11	Read/Write	Read	A NCM_ADV_THIS message has been received from neighbor node of R-port2
	Bit 12	Read/Write	Read	A NCM_MEDIA_LINKED message has been received from neighbor node of R-port2
	Bit 13 – 15	Read/Write	Read	Reserved

4.6.5.10 V(PROTOCOL_VER): Protocol version

This variable holds the protocol version of local device. It is defined as shown in Table 16.

Table 16 – Protocol version

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Bit 0 – 1	Read	Read	0x00: major version 1 0x01: major version 2 0x02: major version 3 0x03: major version 4
	Bit 2 – 3	Read	Read	0x00: minor version 0 0x01: minor version 1 0x02: minor version 2 0x03: minor version 3
	Bit 4 – 7	—	—	Reserved

4.6.5.11 V(DEV_TYPE): Device type

This variable holds the local device type that represents the general function of the device. The value of this variable is defined as shown in Table 17.

Table 17 – Device type

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Bit 0 – 7	Read	Read/Write	0 to 255: general device type 0: invalid device type 1: programmable logic controller (PLC) 2: motion controller 3: human-machine interface (HMI) 4: industrial personal computer 5: inverter 6: simple I/O 7 to 255: reserved
	Bit 8 – 15	Read	Read/Write	0 to 255: application specific device type

4.6.5.12 V(DEV_DESC): Device description

This variable contains a description of the local device. The maximum length of this variable is 16 octets. It is defined as shown in Table 18.

Table 18 – Device description

Data type	Access type DLM	Access type DLS-user	Value/Description
VISIBLE_STRING[16]	—	Read/Write	Any string defined by a DLS-user set using the set DLL configuration service

4.6.5.13 V(HOP_CNT): Hop count

This variable holds the count of the number of devices between two devices. When the DLM receives NCM_ADV_THIS or NCM_LA DLPDU, the DLM saves the received hop count value in this variable, then increments the hop counts in the received frame by 1 and transmits the frame through the other R-port. In this way, each device builds its own path table with a hop count for R-port1 and a hop count for R-port2. This variable is defined as shown in Table 19.

Table 19 – Hop count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	—	0 to 255: Hop count 256 to 65 535: reserved

4.6.6 Variables and counter to support network information management

4.6.6.1 General

Network information is managed automatically by the DLM. Network information variables and counters are summarized in Table 20.

Table 20 – Variables to support managing network information

Parameter	Data type	Default value	Description
Topology	UNSIGNED8	NET_TPG_SA	Network topology
Collision count	UNSIGNED8	0	DL–entity identifier collision counter between remote devices
Device count	UNSIGNED16	1	Device counter for the network segment
Topology change count	UNSIGNED16	0	Network topology change counter
Network flags	UNSIGNED16	0	Network event flags
Last topology change time	TIMEOFDAY	0	Date and time when the network topology last was changed.
Reserved	UNSIGNED16	0	Reserved
RNMP device UID	UNSIGNED64	INVALID_UID	UID of the RNMP device
RNMS device UID	UNSIGNED64	INVALID_UID	UID of the RNMS device
LNM device UID for R-port1	UNSIGNED64	INVALID_UID	UID of the LNM device in R-port1 direction
LNM device UID for R-port2	UNSIGNED64	INVALID_UID	UID of the LNM device in R-port2 direction

4.6.6.2 V(TPG): Topology

This variable holds the type of network topology. The value of this variable is defined as shown in Table 21.

Table 21 – Topology

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	Network topology 0x01: standalone 0x02: line topology 0x03: ring topology

4.6.6.3 C(COLL_CNT): Collision count

This variable holds the DL–entity identifier collision count for remote devices. The value is incremented by the DLM when a remote DL–entity identifier collision is detected and the value is decremented when the collision is cleared. This variable is defined in Table 22.

Table 22 – Collision count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0 to 255: remote DL–entity identifier collision count

4.6.6.4 C(DEV_CNT): Device Count

This variable holds the total number of devices on the network, to a maximum of 256. It is defined as shown in Table 23.

Table 23 – Device count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0: not used 1 to 256: device count 257 to 65 535: reserved

4.6.6.5 C(TPG_CHG_CNT): Topology change count

This variable holds the topology change count. The value is incremented by the DLM when the network is changed from ring to line or from line to ring topology. It is defined as shown in Table 24.

Table 24 – Topology change count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0 to 65 535: Topology change count

4.6.6.6 V(TPG_CHG_TIME): Last topology change time

This variable holds the date and time when the network topology was last changed. It is defined as shown in Table 25.

Table 25 – Last topology change time

Data type	Access type DLM	Access type DLS-user	Value/Description
TIMEOFDAY	Read/Write	Read	The date and time when the network topology was last changed

4.6.6.7 V(UID_RNMP): RNMP device UID

This variable holds the device UID selected as the RNMP on the network. It is defined as shown in Table 26.

Table 26 – RNMP device UID

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.6.8 V(UID_RNMS): RNMS device UID

This variable holds the UID of the device selected as the RNMS on the network. It is defined as shown in Table 27.

Table 27 – RNMS device UID

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.6.9 V(UID_LNM_RP1): LNM device UID for R-port1

In a Type 21 line network, the two end devices are automatically selected as the LNMs. This variable holds the UID of the device selected as the LNM in the R-port1 direction. It is defined as shown in Table 28.

Table 28 – LNM device UID for R-port1

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.6.10 V(UID_LNM_RP2): LNM device UID for R-port2

In a Type 21 line network, two end devices are automatically selected as the LNMs. This variable holds the UID of the device selected as the LNM in the R-port2 direction. It is defined as shown in Table 29.

Table 29 – LNM device UID for R-port2

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED64	Bit 0 – 47	Read/Write	Read	ISO/IEC/IEEE 8802-3 MAC Address
	Bit 48 – 63	Read/Write	Read	0 to 255: DL–entity identifier 256 to 65 535: reserved

4.6.6.11 V(NET_FLAG): Network flags

This variable holds the event flags to notify the DLMS-user of network events. When any bit in this variable is set, the DLM generates a DLM-event indication service primitive to notify the DLMS-user of the event. After the DLM-event service has completed successfully, the designated bit in this variable is cleared. Network flags are defined in Table 30.

Table 30 – Network flags

Data type	Position	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Bit 0	Read/Write	Read	Network DL–entity identifier Collision Status 0x00: Normal 0x01: Network DL–entity identifier collision detected
	Bit 1 – 15	—	—	Reserved

The bit fields in the network flags are defined as follows:

a) Network DL–entity identifier Collision Status;

This bit is set to TRUE by the DLM when it detects a DL–entity identifier collision on the network. This bit is reset to FALSE when the DL–entity identifier collision is cleared.

4.6.7 Variables and counter to support a device path information management

4.6.7.1 General

The path table is managed by the DLM, and it is made up of table items related to the other Type 21 devices on the network. The variables and counters for a table item are defined in Table 31.

Table 31 – Variables and counter to support managing path information

Parameter	Data type	Default value	Description
DL–entity identifier	UNSIGNED16	INVALID_ADDR	DL–entity identifier
Hop count for R-port1	UNSIGNED16	INVALID_HOP_CNT	Hop count in R-port1 direction
Hop count for R-port2	UNSIGNED16	INVALID_HOP_CNT	Hop count in R-port2 direction
Preferred R-port	UNSIGNED8	INVALID_R_PORT	R-port with the smaller hop count value to the peer device.
Destination R-port	UNSIGNED8	INVALID_R_PORT	Selected R-port for sending a frame to the destination DL–entity identifier.
Device state	UNSIGNED8	0	Peer device's DLM State
MAC address	UNSIGNED48	0	Peer device's ISO/IEC/IEEE 8802-3 MAC address
Port information	UNSIGNED16	0	Peer device's local R-port information
Protocol version	UNSIGNED8	0	Peer device's protocol version
Device type	UNSIGNED16	0	Peer device's application device type
Device description	VISIBLE_STRING[16]	" "	Peer device's description
Device UID	UNSIGNED64	INVALID_UID	Peer device's Device UID
Device UID for R-port1	UNSIGNED64	INVALID_UID	Device UID of the device connected through the peer device's R-port1
Device UID for R-port2	UNSIGNED64	INVALID_UID	Device UID of the device connected through the peer device's R-port2
In net count	UNSIGNED16	0	The number of times that the peer device has joined in the network.
In net time	TIMEOFDAY	0	The date and time when the peer device last joined the network.
Out net count	UNSIGNED16	0	The number of times that the peer device has been disconnected from the network.
Out net time	TIMEOFDAY	0	The date and time when the peer device was last disconnected from the network

4.6.7.2 V(path-DL_ADDR): DL–entity identifier

See 4.6.5.2.

4.6.7.3 C(path-HOP_CNT_RP1): Hop count for R-port1

This variable indicates the frame forwarding counts for sending a frame from the local device to the peer device through the R-port1. It is defined as shown in Table 32.

Table 32 – Hop count for R-port1 direction

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0 to 255: Hop count for R-port1 256 to 65 535: reserved

4.6.7.4 C(path-HOP_CNT_RP2): Hop count for R-port2

This variable indicates the frame forwarding counts for sending a frame from the local device to the peer device through the R-port2. It is defined as shown in Table 33.

Table 33 – Hop count for R-port2 direction

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0 to 255: hop count for R-port2 256 to 65 535: reserved

4.6.7.5 V(path-PREFER_RP): Preferred R-port

This variable holds the preferred R-port for sending a frame from the local device to the peer device without regard for the RNMP or RNMS. This variable is determined as the R-port that has the smaller hop count value for the peer device. If the R-port1 hop count and R-port2 hop count have the same value, R-port1 is selected as the preferred R-port. This variable is defined as shown in Table 34.

Table 34 – Preferred R-port

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0: R-port1 1: R-port2 2 to 255: reserved

4.6.7.6 V(path-DST_RP): Destination R-port

This variable holds the destination R-port for sending a frame from the local device to the peer device. In a line network, this variable has the same value as the Preferred R-port. However, in a ring network, this variable is determined based on the RNMP and RNMS position, because the preferred path can be blocked by the RNMP or RNMS. In this case, the destination R-port is selected as the other R-port. It is defined as shown in Table 35.

Table 35 – Destination R-port

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED8	Read/Write	Read	0: R-port1 1: R-port2 2 to 255: reserved

4.6.7.7 V(path-DEV_STATE): Device State

See 4.6.5.4.

4.6.7.8 V(path-MAC_ADDR): MAC address

See 4.6.5.8.

4.6.7.9 V(path-PORT_INFO): Port information

See 4.6.5.9.

4.6.7.10 V(path-PROTOCOL_VER): Protocol Version

See 4.6.5.10.

4.6.7.11 V(path-DEV_TYPE): Device type

See 4.6.5.11.

4.6.7.12 V(path-DEV_DESC): Device Description

See 4.6.5.12.

4.6.7.13 V(path-DEV_UID): Device UID

See 4.6.5.5.

4.6.7.14 V(path-DEV_UID_RP1): Device UID for R-port1

See 4.6.5.6.

4.6.7.15 V(path-DEV_UID_RP2): Device UID for R-port2

See 4.6.5.7.

4.6.7.16 C(path-IN_NET_CNT): In net count

This variable holds the number of times that the peer device has joined the network. When a line network is merged into an existing network, the variables for the newly joined devices are incremented together. This variable is defined as shown in Table 36.

Table 36 – In net count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0 to 65 535: the number of times that the peer device has joined the network

4.6.7.17 V(path-IN_NET_TIME): In net time

This variable holds the date and time when the peer device last joined in the network. This variable is defined as shown in Table 37.

Table 37 – In net time

Data type	Access type DLM	Access type DLS-user	Value/Description
TIMEOFDAY	Read/Write	Read	The date and time when the device last joined in the network

4.6.7.18 C(path-OUT_NET_CNT): Out net count

This variable holds the number of times that the peer device has been disconnected from the network. When a device or a group of devices are disconnected from the network, the variables for the disconnected devices are incremented together. This variable is defined as shown in Table 38.

Table 38 – Out net count

Data type	Access type DLM	Access type DLS-user	Value/Description
UNSIGNED16	Read/Write	Read	0 to 65 535: the number of times that the device has been disconnected from the network

4.6.7.19 V(path-OUT_NET_TIME): Out net time

This variable holds the date and time when the device was last disconnected from the network. This variable is defined as shown in Table 39.

Table 39 – Out net time

Data type	Access type DLM	Access type DLS-user	Value/Description
TIMEOFDAY	Read/Write	Read	The date and time when the device was last disconnected from the network

4.6.8 Variables, counters, timers, and queues to support path table management

4.6.8.1 Path table

The path table is composed of items related to the other devices on the network. It is managed by the DLM in the form of an array table filled with the path information of each device (see 4.6.7). The maximum size of the path table is a function of MAX_ADDR (see 4.6.2.2) as follows:

Path table: Array[n] of device's path information, n = MAX_ADDR + 1

5 General structure and encoding

5.1 Overview

The DLL and its procedures are necessary to provide services to the DLS-user using the services available from the physical layer. This clause describes the structure and semantics of the management application protocol data unit (MAPDU), the DLPDU, and the procedure commonly used in this document. This portion is identical to and fully compliant with ISO/IEC/IEEE 8802-3:2021.

NOTE In Clause 5, 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. This is sometimes referred to as "little-endian" bit numbering.

5.2 MAPDU structure and encoding

The local MAC sublayer uses the service primitives provided by the physical layer service (PLS) sublayer specified by ISO/IEC/IEEE 8802-3:2021, Clause 2. These service primitives provided by the PLS sublayer are mandatory:

a) MA-DATA request;

b) MA-DATA indication.

5.3 Common MAC frame structure, encoding and elements of procedure

5.3.1 MAC frame structure

5.3.1.1 MAC frame format for the Type 21 DLPDU

The DLPDU for the Type 21 is encapsulated in the data field of a MAC frame as specified by ISO/IEC/IEEE 8802-3:2021, Clause 3. The value of the Length/Type field is $88FE_H$, which is authorized and registered as the protocol identification number by the IEEE Registration Authority, to identify a Type 21 fieldbus frame. Figure 4 shows the Type 21 DLPDU structure.

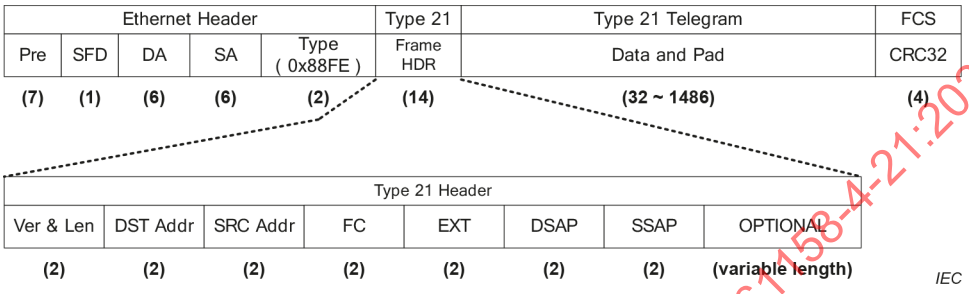


Figure 4 – Common MAC frame format for Type 21 DLPDU

5.3.1.2 MAC frame format for Type 21 fieldbus sporadic DLPDU

The MAC frame format used for Type 21 fieldbus sporadic data transmission is identical to the frame format of Ethernet V2.0 specified by ISO/IEC/IEEE 8802-3:2021, Clause 3 "Media access control frame structure," and the value of the Length/Type field is anything other than 0x88FE. Figure 5 shows the frame format for a Type 21 fieldbus sporadic DLPDU.

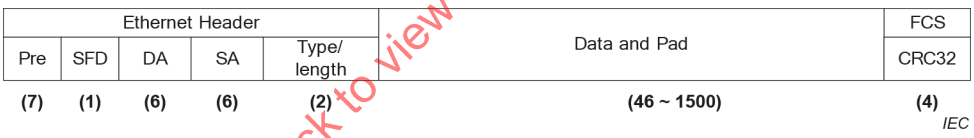


Figure 5 – MAC frame format for other protocols

5.3.2 Elements of the MAC frame

5.3.2.1 General

The elements of the MAC frame are the preamble, the start frame delimiter, the destination MAC address, the source MAC address, the length/type code, and the frame check sequence (FCS), all as specified by ISO/IEC/IEEE 8802-3:2021, Clause 3.

5.3.2.2 Preamble field

The preamble of MAC frame is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3. This is a 7-octet field that is used to allow the physical signaling portion of the circuitry to reach its steady-state synchronization with the receiving frame timing. The preamble pattern is:

"10101010 10101010 10101010 10101010 10101010 10101010 10101010"

The bits are transmitted in order from left to right. The nature of the pattern is such that for Manchester encoding, it appears as a periodic waveform on the medium that enables bit synchronization. It should be noted that the preamble ends with a "0".

5.3.2.3 Start Frame Delimiter

The Start Frame Delimiter (SFD) is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3. The SFD field is the bit pattern sequence "10101011." It immediately follows the preamble pattern and indicates the start of a frame.

5.3.2.4 Destination MAC Address field

The Destination MAC Address field is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3. It specifies the device(s) for which the frame is intended, and can be an individual or multicast (including broadcast) address. The destination MAC address is set to the corresponding DL–entity identifier by the DLM. Type 21 also defines a special MAC address, 00-E0-91-02-05-99 (NCM_MAC_ADDR) for sharing network management information using the DLM services. Every message received through the NCM_MAC_ADDR is delivered to the DLM to update the network management information. The message is not forwarded by the MAC layer but the message is examined and forwarded by the DLM.

5.3.2.5 Source MAC address field

The Source MAC Address field is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3. This field specifies the device sending the frame and is not interpreted by the DLE or the CSMA/CD MAC sublayer.

5.3.2.6 Length/type field

The Length/type field is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3 "Media access control frame structure." To be identified as a Type 21 frame, the value of the Length/type field is set to 0x88FE, which is authorized and registered as the protocol identification number for RTE-Type 21 by the IEEE Registration Authority. Every frame with a value other than 0x88FE is identical to the frame in ISO/IEC/IEEE 8802-3:2021, Clause 3, and is processed as a Type 21 fieldbus sporadic data frame.

5.3.2.7 Frame check sequence

The FCS field is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3.

5.3.3 Elements of the Type 21 DLPDU

5.3.3.1 Version and Length

This field stores the protocol version and the length of a Type 21 telegram or data field. This Version and Length field is specified in Figure 6. The version is represented by 2 bits for the major version and 3 bits for the minor version, and the length is given by 11 bits.

Version and Length															
Version				R	Length										
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

IEC

Figure 6 – Version and Length field

The parts of this field and permissible values are described in Table 40.

Table 40 – Version and Length

Field Name		Position	Value/Description
Length		Bit 0 – 10	Frame length including FCS field
Reserved		Bit 11	Reserved
Version	Minor	Bit 12 – 13	Type 21 Protocol minor version 0x00: minor version 0 0x01: minor version 1 0x02: minor version 2 0x03: minor version 3
	Major	Bit 14 – 15	Type 21 Protocol major version 0x00: major version 1 0x01: major version 2 0x02: major version 3 0x03: major version 4

5.3.3.2 DST_addr

5.3.3.2.1 General

This field indicates the destination DL–entity identifier of the node to which the frame is sent. This value is represented as shown in Figure 7.

Destination DL- address															
DST_Addr															
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

IEC

Figure 7 – DST_addr field

The separate field and its permissible values are described in Table 41.

Table 41 – Destination DL–entity identifier

Field Name	Position	Value/Description
DST_addr	Bit 0 – 15	0xFF: broadcast address 0xFE: network control address (C_NCM_ADDR) 0xFD–0xDE: user-defined multicast address 0xDD: invalid address 0x0100 to 0xFFFF: reserved 0x00 to 0xDC: regular Type 21 DL–entity identifier

5.3.3.2.2 Broadcast address

If the destination DL–entity identifier is 0xFF, the destination MAC address field contains the ISO/IEC/IEEE 8802-3 MAC address.

5.3.3.2.3 Network control address

If the destination DL–entity identifier is 0xFE (C_NCM_ADDR), the destination MAC address field contains C_NCM_MAC_ADDR. However, NCM_LINK_ACTV and NCM_ADV_THIS messages are transmitted using C_NCM_ADDR as the destination DL–entity identifier.

NOTE C_NCM_ADDR cannot be accessed by the DLS-user.

5.3.3.2.4 User-defined multicast address

A user-defined multicast address is used to indicate multiple recipients. However, user-defined multicast addressing is not a mandatory feature in this document. It is designed for use in a special application system that requires multicast communication. Therefore, user-defined multicast addressing is not interoperable between heterogeneous devices. The destination DL–entity identifier range from 0xFD–0xDE is used to specify the user-defined multicast address. However, the method of using the user-defined multicast address is not specified in this document, and is considered a local responsibility. This document does not restrict the use of user-defined multicast addresses, nor is it a mandatory feature.

5.3.3.3 SRC_addr

This field indicates the source DL–entity identifier of the node from which the frame is generated. This value is represented as shown in Figure 8.

Source DL- address															
SRC_Addr															
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

IEC

Figure 8 – SRC_addr field

The separate field and its permissible values are described in Table 42.

Table 42 – Source DL–entity identifier

Field Name	Position	Value/Description
SRC_addr	Bit 0 – 15	Source DL–entity identifier

5.3.3.4 Frame Control (FC)

5.3.3.4.1 General

The Frame Control field indicates the frame control information. This value is represented as shown in Figure 9.

Frame Control															
VoE	RES	PRI		ToS				NCMT							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

key
VoE: Validation of Extension code
RES: Reserved
PRI : Priority
ToS: Type of Service
NCMT: Network control message type

IEC

Figure 9 – Frame Control Field

The separate field and its permissible values are described in Table 43.

Table 43 – Frame control

Field Name	Position	Value/Description
Network Control Message Type (NCMT)	Bit 0 – 7	0x00: reserved 0x01: NCM_FAMILY_REQ 0x02: NCM_FAMILY_RES 0x03: NCM_MEDIA_LINKED 0x04: NCM_ADV_THIS 0x05: NCM_LINE_START 0x06: NCM_RING_START 0x07: NCM_ACK_RNMS 0x08: NCM_CHECK_NET_INTEGRITY 0x09: NCM_CHECK_NEIGHBOR_INTEGRITY 0x0A: NCM_NET_IS_RING 0x0B: NCM_RENEGO_REQ 0x0C to 0xFF:reserved
Type of Service (ToS)	Bit 8 – 11	0x00: Network Control Message (NCM) 0x01: unconfirmed service request 0x02–0x0F: reserved
Priority (PRI)	Bit 12 – 13	0x00: lowest priority … 0x03: highest priority
Reserved	Bit 14	Reserved
Validation of Extension code (VoE)	Bit 15	0x00: EXT Code is invalid 0x01: EXT Code is valid(default)

5.3.3.4.2 Validation of extension code (VoE)

If the frame has the extension field, VoE is set to TRUE; otherwise VoE is set to FALSE.

5.3.3.4.3 Priority

This field indicates the frame priority. This field contains the value of the message priority parameter for the DL service. The highest priority is 0x03 and the lowest is 0x00.

5.3.3.4.4 Type of service (ToS)

This field indicates the type of DL service. A value of 0x00 indicates a network control message among DLMs, and 0x01 indicates the unconfirmed service request among DLS-users.

5.3.3.4.5 Network Control Message Type (NCMT)

5.3.3.4.5.1 General

NCMT field indicates the type of network control message.

5.3.3.4.5.2 NCM_FAMILY_REQ

NCMT: 0x01

This network control message is used to ask the device newly connected through an R-port if it is a Type 21 device. This network control message is transmitted through the newly activated R-port. This message shall not be forwarded to the other port.

5.3.3.4.5.3 NCM_FAMILY_RES

NCMT: 0x02

This network control message is used to confirm whether the recipient is a Type 21 device when the recipient receives the NCM_FAMILY_REQ message from the newly linked device. This message is transmitted through the R-port used to receive the NCM_FAMILY_REQ message. This message shall not be forwarded to the other port.

5.3.3.4.5.4 NCM_MEDIA_LINKED

NCMT: 0x03

This network control message is used to indicate that a new Type 21 link has been established through the R-port. This message is transmitted through the newly activated R-port. The destination DL–entity identifier contains C_NCM_ADDR. When the DLM receives this message, the DLM increments the hop count in the frame, and forwards the frame through the other R-port. This message is discarded by the LNM or the device that generated the message.

5.3.3.4.5.5 NCM_ADV_THIS

NCMT: 0x04

This network control message is used to transmit the recipient's local device information when the recipient receives NCM_MEDIA_LINKED message from a new device on the network. This message is transmitted through the R-port that is used to receive the NCM_MEDIA_LINKED message. The destination DL–entity identifier contains C_NCM_ADDR. When the DLM receives this message, the DLM increments the hop count in the frame, and forwards the frame through the other R-port. This message is discarded by the LNM or the device that generated the message.

5.3.3.4.5.6 NCM_LINE_START

NCMT: 0x05

This network control message is used to broadcast that the network topology has been automatically configured as a line network. This message is initiated by the DLM whose state is changed to LNM when the existing line network is divided into two line networks, or when a link failure is detected in a ring network and the network is reconfigured as a line network. This message is broadcast on the network using the broadcast address.

5.3.3.4.5.7 NCM_RING_START

NCMT: 0x06

This network control message is used to broadcast that the network topology has been automatically configured as a ring network. This message is initiated and broadcast through both R-ports by the DLM whose state is changed to RNMP.

5.3.3.4.5.8 NCM_ACK_RNMS

NCMT: 0x07

This network control message is used by the RNMS device to broadcast that the RNMS has been successfully selected. This message is transmitted from the RNMS to the RNMP.

5.3.3.4.5.9 NCM_CHECK_NET_INTEGRITY

NCMT: 0x08

This network control message is used by RNMS, RNMP and LNM to check the integrity of the network path. In the ring topology, it is a message periodically transmitted/received by two devices, RNMP and RNMS. This message contains the period value of the sender. The receiver sets three times the period value included in the message as the timeout value. When a timeout occurs in the R-port direction, NCM_CHECK_NEIGHBOR_INTEGRITY is broadcast to the R-port.

5.3.3.4.5.10 NCM_CHECK_NEIGHBOR_INTEGRITY

NCMT: 0x09

This network control message is used by RNMS, RNMP and LNM to detect network fault location in the RRP network.

5.3.3.4.5.11 NCM_NET_IS_RING

NCMT: 0x0A

This network control message is used by two LNM devices to broadcast network topology has been changed from line to ring by connecting two LNMs.

5.3.3.4.5.12 NCM_RENEGO_REQ

NCMT: 0x0B

This network control message is used when requesting to restart negotiation between two RRP devices.

5.3.3.5 Extension (EXT)

5.3.3.5.1 General

This field exists when the VoE bit in the frame control field is set to TRUE. The extension field is specified as shown in Figure 10.

<table>
<tr><td colspan="16">Extension Code</td></tr>
<tr><td>G</td><td colspan="7">Extension Type</td><td colspan="8">Extension Length</td></tr>
<tr><td>15</td><td>14</td><td>13</td><td>21</td><td>11</td><td>10</td><td>9</td><td>8</td><td>7</td><td>6</td><td>5</td><td>4</td><td>3</td><td>2</td><td>1</td><td>0</td></tr>
</table>

key
G: Group Mask

IEC

Figure 10 – Extension field

The separate field and its permissible values are described in Table 44.

Table 44 – Extension

Field Name	Position	Value/Description
Extension Length	Bit 0 – 7	0 to 255: the length of extension field
Extension Type	Bit 8 – 14	0: Invalid extension type 1 to 127: reserved for future use
Group Mask Enable	Bit 15	0x0: Group mask is enabled 0x1: Group mask is disabled

5.3.3.5.2 Group Mask Enable

Group Mask Enable is a bit field to specify whether the frame is to be accepted by the peer device or not, when the frame is broadcast or multicast. When the value is set to TRUE, group mask is enabled in the peer device that receives the frame. Otherwise, group mask is disabled in the peer device. When the group mask is enabled, the group mask fields are appended in the option field (see 5.3.3.8).

5.3.3.5.3 Extension Type

This field indicates the type of extension field. The value 0x00 indicates an invalid extension type and the other values are reserved for future use.

5.3.3.5.4 Extension Length

This field indicates the length of the extension field. When Group Mask Enable is set to TRUE and extension type is set to 0x00, the extension length specifies the length of the group mask field. When group mask enable is set to FALSE and extension type is set to a value other than 0x00, extension length specifies the length of the extension field. When group mask enable is set to TRUE and extension type is not set to 0x00, the first two octets specify the length of the group mask field and the next two octets specify the extension type.

5.3.3.6 DSAP

This field indicates the SAP of the DLE to which the DLPDU is sent. The permissible values are in the range 0 to 65 535. The DSAP is specified as shown in Figure 11 and Table 45.

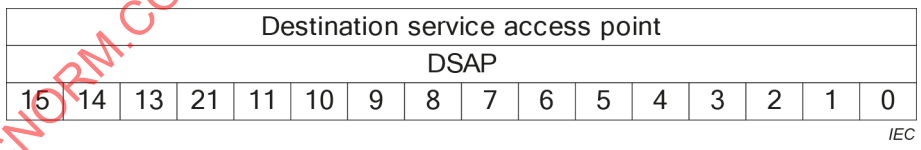


Figure 11 – DSAP field

Table 45 – Destination service access point

Field Name	Position	Value/Description
DSAP	Bit 0 – 15	Service access point of destination DLE

5.3.3.7 SSAP

This field indicates the SAP of the DLE from which the DLPDU is generated. The permissible values are in the range 0 to 65 535. The DSAP is specified as shown in Figure 12 and Table 46.

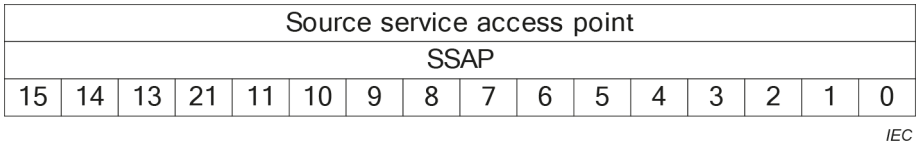


Figure 12 – Source service access point field

Table 46 – source service access point

Field Name	Position	Value/Description
SSAP	Bit 0 – 15	Service access point of source DLE

5.3.3.8 Option

5.3.3.8.1 General

This field indicates the option field when the VoE (see 5.3.3.4.2) is set to TRUE. The option field is used to indicate group mask information or other additional information. The maximum length of option field is limited to 256 octets.

5.3.3.8.2 Length of group mask and extension information

This field indicates the length of group mask and extension information. When group mask enable is set to TRUE and the extension type is not 0x00, the first two octets indicate the length of the group mask field and the next two octets indicate the length of the extension type field. When group mask enable is set to FALSE and the extension type is 0x00, the length of group mask and extension information field is ignored. Figure 13 shows the length of group mask and extension information.

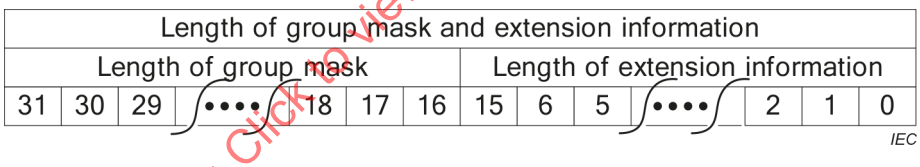


Figure 13 – Length of group mask and extension information

5.3.3.8.3 Group mask

This field uses a bit sequence to indicate the receipt selection of a message. When group mask enable is set to TRUE, the group mask field is appended in 4-octet units, for example, 4, 8, 12 to 32 octets. Each bit indicates the receipt selection of the frame for the corresponding DL–entity identifier. A 1 means TRUE for frame receipt and 0 means FALSE. The first bit indicates the frame receipt option for highest DL–entity identifier. Figure 14 shows the bit sequence order of group mask field when its length is set to 255. When the extension type is set to 0x00, the group mask field is appended to the option field. Otherwise, the group mask field is appended after the length of group mask and extension information fields.

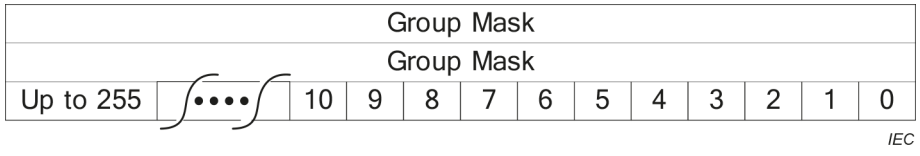


Figure 14 – Group mask option field

5.3.3.8.4 Extension information

This field is reserved for future extension. This field contains 4 octets aligned extension information.

5.3.3.9 Data and pad

This field indicates the data field received from a DLS-user.

5.4 Order of bit transmission

The order of bit transmission is identical to ISO/IEC/IEEE 8802-3:2021, Clause 3. Each octet of the DLPDU with the exception the FCS is transmitted with the low-order bit first.

5.5 Invalid DLPDU

An invalid DLPDU shall be defined as one that meets at least one of the following conditions; this is almost identical to ISO/IEC/IEEE 8802-3:2021, Clause 3:

a) the frame length is inconsistent with a length value specified in the Length/type field. If the Length/type field contains a type value defined by ISO/IEC/IEEE 8802-3:2021, 3.2.6, then the frame length is assumed consistent with this field and should not be considered an invalid DLPDU on this basis;

b) it is not an integral number of octets in length;

c) the bits of the incoming DLPDU excluding the FCS field do not generate a CRC value identical to the one received;

d) it is inconsistent with a F-type value of Type 21 fieldbus DLPDU.

The contents of invalid DLPDUs shall not be passed to the DL-user or DLE. The occurrence of an invalid DLPDU can be communicated to the network management.

Invalid DLPDUs can be ignored, discarded, or used in a private manner by a DL-user other than the RTE DL-user.

NOTE The use of such DLPDUs is beyond the scope of this document.

6 DLPDU structure and procedure

6.1 General

This clause defines the structure, contents, and encoding for each type and format of the DLPDU, along with procedural elements. Subclauses describe the structure, contents, parameters, and encoding of the DLPDU, along with the Type 21-specific part of the DLPDU structure, which is shown in Figure 4. The aspects relating to sending and receiving by DLS-users and their DLEs are also described.

NOTE In Clause 6, 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, and this is sometimes referred to as "little-endian" bit numbering.

6.2 Common DLPDU Field

6.2.1 General

Figure 15 shows the common DLPDU field. The Version and Length field is common to every DLPDU and is not described further for each DLPDU type.

Version and Length															
Version					Length										
15	14	13	21	11	10	9	8	7	6	5	4	3	2	1	0

IEC

Figure 15 – Common DLPDU field

6.2.2 Version

This field indicates the Type 21 protocol version with a 5 bit sequence. The highest 2 bits specify the major version and the lowest 3 bits specify the minor version (see 5.3.3.1).

6.2.3 Length

This field indicates the length of the data field in octets including the FCS field. The permissible values are in the range 12 to 1 498.

6.3 DL-DATA Transfer

6.3.1 DT DLPDU

6.3.1.1 General

The DT DLPDU is used to carry Type 21 data from one device to another.

6.3.1.2 DT DLPDU structure

Figure 16 shows the process of building a DT DLPDU and Figure 17 shows the structure of the DT DLPDU.

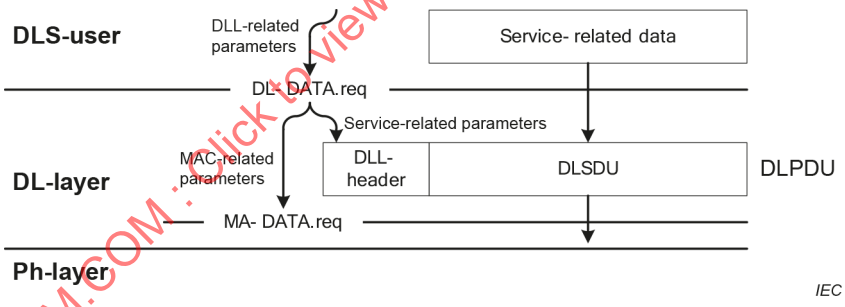


Figure 16 – Building a DT DLPDU

DST Addr	SRC Addr	FC	EXT	DSAP	SSAP	opt	DATA

IEC

Figure 17 – DT DLPDU structure

6.3.1.3 DT DLPDU Parameters

6.3.1.3.1 General

Table 47 shows the list of DT DLPDU parameters.

Table 47 – DT DLPDU parameters

Parameter	Data type	Value/Description
DST_addr	UNSIGNED16	Destination DL–entity identifier [Optional] -Broadcast -Multicast (user-defined) -Unicast
Priority	UNSIGNED8	0x00 to 0x03 0x04 to 0xFF: reserved
DSAP	UNSIGNED16	Destination DLSAP of remote device
SSAP	UNSIGNED16	Source DLSAP
Group mask	—	Intended destination device bit array. Only when broadcast or multicast.
Group mask length	UNSIGNED8	Length of group mask. Only when broadcast or multicast.
Data	—	DLSDU
Data Length	UNSIGNED16	Length of DLSDU
Extension type	UNSIGNED8	0x00: invalid 0x01 to 0x7F
Extension data	—	4 octets aligned Extension data
Extension length	UNSIGNED16	Length of Extension data

6.3.1.3.2 DST_addr

This parameter indicates the destination DL–entity identifier of the DLE(s) for which the DLPDU is intended. This can be an individual or multicast (including broadcast) DL–entity identifier, not its MAC address. The destination address for unicast is explicitly assigned a value between 0x00 and 0xFF. A user-defined multicast address is assigned a value in the range 0xDE to 0xFD. A broadcast address is assigned the value 0xFF. The values 0x0100–0xFFFF are reserved. The value 0xDD indicates an invalid address (see 5.3.3.2).

6.3.1.3.3 Priority

This field indicates the frame priority with a 2 bit sequence. This field contains the value of the message priority parameter of the DL service. The highest priority is 0x03, and 0x00 is the lowest (see 5.3.3.4.3).

6.3.1.3.4 DSAP

See 5.3.3.6.

6.3.1.3.5 SSAP

See 5.3.3.7.

6.3.1.3.6 Group mask

See 5.3.3.8.3.

6.3.1.3.7 Group mask length

See 5.3.3.8.2.

6.3.1.3.8 Data

This data field is the DLSDU.

6.3.1.3.9 Data length

This field indicates the length of the data field in octets. The permissible values are in the range 0 to 1 486.

6.3.1.3.10 Extension type

See 5.3.3.5.3.

6.3.1.3.11 Extension data

This data field is the Extension data.

6.3.1.3.12 Extension length

See 5.3.3.5.4.

6.3.1.4 Sending

When the local DLS-user initiates a DL-DATA service request to transfer a DLSDU to a peer DLS-user, the DLSDU is stored in the RT-queue and transmitted by the DLPM using a DT DLPDU. The group mask option is available when the DL-DATA DLPDU is broadcast or multicast.

Table 48 shows the required data link service primitives and parameters to send a DT DLPDU. To send a DT DLPDU to a peer DLE, the DLPM queries the peer device's path information to the DLM using the DLM-GET_PATH service. Then, the DT DLPDU is transmitted using an MA-DATA service request primitive.

Table 48 – Primitives exchanged between DLS-user and DLE to send a DT DLPDU

Primitive	Source	Associated parameters
DL-DATA request	DLS-user	DST_addr DSAP SSAP Priority Group mask Group mask length Data Data length Extension type Extension data Extension length
DLM-GET_PATH request	DLE (DLPM)	DST_addr
DLM-GET_PATH confirm	DLM	Status R-port MAC address
MA-DATA request	DLE (DLPM)	DLPDU DLPDU length

6.3.1.5 Receiving

When a DT DLPDU is received through the MAC layer, the DLPM checks that it is valid and if it is intended for the local DLS-user. If the DT DLPDU is broadcast or multicast, the DLPM examines the group mask field to check if the DT DLPDU is intended for the local DLS-user. If the DLPDU is intended for the local DLS-user, the DLPM extracts the DLSDU from the received DT DLPDU and delivers the DLSDU to the appropriate DLS-user using a DL-DATA indication primitive. If no DLS-user is registered for the DSAP in the received DT DLPDU, the DLPM discards the DT DLPDU just received.

Table 49 shows the required data link service primitives and parameters to receive a DT DLPDU. When a DT DLPDU is received through the MA-DATA indication service primitive, the DLPM extracts the DLSDU from the received DT DLPDU and delivers the DLSDU to the appropriate DLS-user.

Table 49 – Primitives exchanged between DLS-user and DLEs to receive a DT DLPDU

Primitive	source	Associated parameters
MA-DATA indication	MAC	DLPDU DLPDU length
DL-DATA indication	DLPM	DLSDU DLSDU length SRC DL Address DST DL Address DSAP SSAP Type of Service(ToS) Group receive Group length Group data Extension type Extension length Receive R-port

6.4 DL-SPDATA Transfer

6.4.1 SPDT DLPDU

6.4.1.1 General

SPDT DLPDUs are used to carry non-Type 21 data, such as IP.

6.4.1.2 SPDT DLPDU structure

Figure 18 shows the process of building an SPDT DLPDU and Table 50 shows the structure of the SPDT DLPDU. The Type 21 header is not included in the SPDT DLPDU.

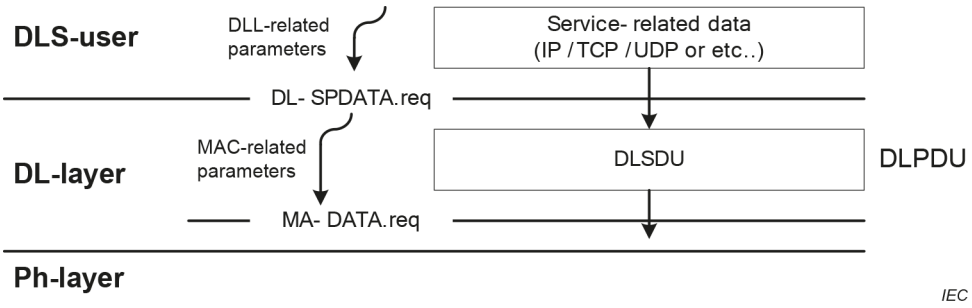


Figure 18 – SPDT DLPDU structure

6.4.1.3 SPDT DLPDU parameters

6.4.1.3.1 General

Table 50 shows the parameter list for the SPDT DLPDU.

Table 50 – SPDT DLPDU Parameters

Parameter	Data type	Value/Description
Data	Octetstring	DLSDU
Data Length	UNSIGNED16	Length of DLSDU

6.4.1.3.2 Data

This field indicates the DLSDU, which is to be transmitted in a ISO/IEC/IEEE 8802-3 Ethernet frame. This DLSDU can be any protocol data unit, such as an IP packet.

6.4.1.3.3 Data length

This field indicates the length of the data field in octets. The permissible values are in the range 0 to 1 514.

6.4.1.4 Sending

When the local DLS-user initiates a DL-SPDATA service request to transfer a DLSDU to the peer DLS-user, the DLSDU is stored in the NRT-queue and transmitted by the DLPM using an SPDT DLPDU. In this case, the DLSDU is transmitted as the SPDT DLPDU without a Type 21 header.

Table 51 shows the required data link service primitives and parameters to send an SPDT DLPDU. The SPDT DLPDU is transmitted by the DLPM using a MA-DATA request service primitive.

Table 51 – Primitive exchanged between DLS-User and DLEs to send an SPDT DLPDU

Primitive	Source	Associated parameters
DL-SPDATA request	DLS-User	Data: DLSDU Data length: length of DLSDU
MA-DATA request	DLPM	DLPDU DLPDU length
MA-DATA confirm	MAC	Status

6.4.1.5 Receiving

When an SPDT DLPDU is received through the MAC layer, the DLPM checks that it is valid and if it is intended for the local DLS-user. If this is the case, the DLPM extracts the DLSDU from the SPDT DLPDU and delivers it to the appropriate DLS-user using a DL-DATA indication primitive.

Table 52 shows the required data link service primitives and parameters to receive an SPDT DLPDU. When an SPDT DLPDU is received through the MA-DATA indication service primitive, the DLPM extracts the DLSDU from the received SPDT DLPDU and delivers the DLSDU to the appropriate local DLS-user.

Table 52 – Primitives exchanged between DLS-user and DLEs to receive an SPDT DLPDU

Primitive	Source	Associated parameters
MA-DATA indication	DLPM	DLPDU DLPDU length
DL-SPDATA indication	MAC	DLSDU DLSDU length

6.5 Network control messages

6.5.1 General

Network control message (NCM) DLPDUs are used to transfer the network control messages among DLMs. Five message types are provided to share the network information.

6.5.2 NCM_LA DLPDU

6.5.2.1 General

NCM_LA DLPDU is used to transfer local device information including user defined information that uses the extension field to the peer device when a link is established. When the DLM detects the change of link status through the Ph-LINK_STATUS_CHANGE indication primitive, the DLM queries the link status information of the physical layer. After that, the DLM generates an NCM_LA DLPDU to send the changed link information to the peer device over the newly established link.

6.5.2.2 NCM_LA DLPDU structure

NCM messages are based on the basic Type 21 DLPDU structure shown in Figure 19. Figure 19 shows the structure of the NCM_LA DLPDU after the Ethernet header.

Figure 19 – NCM_LA DLPDU structure

6.5.2.3 NCM_LA DLPDU parameters

Table 53 shows the parameter list of the NCM_LA DLPDU.

Table 53 – NCM_LA DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	C_NCM_ADDR (see 5.3.3.2)
SRC_addr	UNSIGNED16	Local DL–entity identifier DL_ADDR (see 5.3.3.3)
NCMT	UNSIGNED8	NCM_FAMILY_REQ (see 5.3.3.4.5.2)
DLMDU	—	DL management data unit Local device information (see 4.6.5)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	Newly link-activated R-port

6.5.2.4 Sending

When the DLM detects the change in the link status through the Ph-LINK_STATUS_CHANGE indication primitive, the DLM queries the link status information of the physical layer. If the link is newly established, the DLM generates NCM_LA DLPDU to send the changed link information to the peer device over the newly established link. NCM_LA DLPDU is stored in the RT-queue and transmitted by the DLPM using the MA-DATA request primitive.

6.5.2.5 Receiving

The C_NCM_ADDR is used as the destination DL–entity identifier in the NCM_LA DLPDU. When an NCM_LA DLPDU is received through the MAC layer, the DLM increases the hop count field in the DLPDU and forwards the DLPDU through the other R-port. The NCM_LA DLPDU is discarded by the LNM or the device that generated the DLPDU. NCM_LA DLPDU is received and processed by the DLM as follows:

a) the DLM checks whether the DLPDU is generated by the device itself. If this is the case, and the DLPDU is received through the other R-port, it means the network is configured as a ring network. If this is the case, proceed to step f) and finish the procedure without step g). Otherwise, proceed to step b);

b) set the R-port's linked device type in the PORT_INFO (see 4.6.5.9) to "homogeneous device type";

c) extract the device path information (DPI) from the DLMDU and check for a DL–entity identifier collision using the DPI. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL–entity identifier collision is detected, set the DEV_FLAG (see 4.6.5.3) to "Local DL–entity identifier Collision," and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL–entity identifier collision is detected, set the NET_FLAG (see 4.6.6.11) to "Network DL–entity identifier Collision Status," and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive;

d) increment the HOP_CNT (see 4.6.5.13) in the received DLMDU;

e) forward the modified NCM_LA DLPDU through the other R-port;

f) check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 7.3.3);

g) generate an NCM_AT DLPDU and transmit the DLPDU through the opposite R-port.

6.5.3 NCM_AT DLPDU

6.5.3.1 General

An NCM_AT DLPDU is used to transfer local device information to the other devices on the network when a device receives an NCM_LA DLPDU. This DLPDU is transmitted through the R-port that is used to receive the NCM_LA DLPDU.

6.5.3.2 NCM_AT DLPDU structure

See 6.5.2.2.

6.5.3.3 NCM_AT DLPDU parameters

Table 54 shows the parameter list of NCM_AT DLPDU.

Table 54 – NCM_AT DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	C_NCM_ADDR (see 5.3.3.2.3)
SRC_addr	UNSIGNED16	Local DL–entity identifier DL_ADDR (see 4.6.5.2)
NCMT	UNSIGNED8	NCM_ADV_THIS (see IEC 61158-3-21, 4.4.3.2.4)
DLMDU	—	Local device information (see 4.6.5)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	R-port used to receive the NCM_LA DLPDU

6.5.3.4 Sending

When the DLM receives an NCM_LA DLPDU, the DLM generates an NCM_AT DLPDU for the DLMDU including the local device information. The DLMDU is stored in the RT-queue and transmitted by the DLMP through the R-port through which the NCM_LA DLPDU was received. In this case, C_NCM_ADDR is used as the destination DL–entity identifier.

6.5.3.5 Receiving

The C_NCM_ADDR is used as the destination address in an NCM_AT DLPDU. When an NCM_AT DLPDU is received through the MAC layer, the DLM increments the hop count field in the DLPDU and forwards the DLPDU through the other R-port. The NCM_AT DLPDU is discarded by the LNM or the device that generated the DLPDU. The NCM_AT DLPDU is received and processed by the DLM as follows:

a) the DLM checks whether the DLPDU was generated by the device itself. If the DLPDU was generated locally and received through the other R-port, it means the network is configured as a ring network. In this case, proceed to step f). Otherwise, proceed to step b);

b) set the R-port's linked device type in the PORT_INFO (see 4.6.5.9) to "homogeneous device type";

c) extract the device path information (DPI) (see 4.6.7) from the DLMDU and check for a DL–entity identifier collision using the DPI. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL–entity identifier collision is detected, set the DEV_FLAG (see 4.6.5.3) to "Local DL–entity identifier Collision," and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL–entity identifier collision is detected, set the NET_FLAG (see 4.6.6.11) to "Network DL–entity identifier Collision Status," and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive;

d) increment HOP_CNT (see 4.6.5.13) in the received DLMDU;

e) forward the modified NCM_LA DLPDU through the other R-port;

f) check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 7.3.3).

6.5.4 NCM_LS DLPDU

6.5.4.1 General

The NCM_LS DLPDU is used to indicate that the network is automatically configured as a line topology when a line network is newly established, or that an existing line network is divided into two lines, or that a ring network has been reconfigured as a line network. This DLPDU is generated by the device that is selected as the LNM on the network.

6.5.4.2 NCM_LS DLPDU structure

See 6.5.2.2.

6.5.4.3 NCM_LS DLPDU parameters

Table 55 shows the NCM_LS DLPDU parameter list.

Table 55 – NCM_LS DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	Broadcast address (see 5.3.3.2.2)
SRC_addr	UNSIGNED16	Local DL–entity identifier DL_ADDR (see 4.6.5.2)
NCMT	UNSIGNED8	NCM_LINE_START (see 5.3.3.4.5.4)
DLMDU	—	Local device information (see 4.6.5)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	link-activated R-port

6.5.4.4 Sending

When the state of the DLM is changed to LNM, the DLM generates an NCM_LS DLPDU to inform every device on the network of the change in network topology. The NCM_LS DLPDU is stored in the RT-queue and it is broadcast by the DLPM using the MA-DATA request primitive.

6.5.4.5 Receiving

The broadcast DL–entity identifier is used as the destination address in an NCM_LS DLPDU. Therefore, the NCM_LS DLPDU is directly forwarded by the MAC layer. The NCM_LS DLPDU is processed by the DLM as follows:

a) extract the DPI (see 4.6.7) from the DLMDU and use it to check for a DL–entity identifier collision. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL–entity identifier collision is detected, set the DEV_FLAG (see 4.6.5.3) to "Local DL–entity identifier Collision," and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL–entity identifier collision is detected, set the NET_FLAG (see 4.6.6.11) to "Network DL–entity identifier Collision Status," and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive;

b) check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 7.3.3);

c) if the DLM state is not LNM or RNM, enable the frame forward functions and set "frame forwarding function from R-port1 to R-port2" and "Frame Forwarding Function from R-port2 to R-port1" in the PORT_INFO (see 4.6.5.9) to TRUE.

6.5.5 NCM_RS DLPDU

6.5.5.1 General

The NCM_RS DLPDU is used to indicate that the network is automatically configured as a ring topology. This DLPDU is generated by the device that is selected as the RNMP on the network.

6.5.5.2 NCM_RS DLPDU structure

See 6.5.2.2.

6.5.5.3 NCM_RS DLPDU parameters

Table 56 shows the NCM_RS DLPDU parameter list.

Table 56 – NCM_RS DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	Broadcast address
SRC_addr	UNSIGNED16	Local DL–entity identifier, DL_ADDR (see 4.6.5.2)
NCMT	UNSIGNED8	NCM_RING_START (see IEC 61158-3-21, 4.4.3.2.4)
DLMDU	—	Local device information (see 4.6.5)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	R-port1, R-port2

6.5.5.4 Sending

When the state of the DLM is changed to RNMP, the DLM generates an NCM_RS DLPDU to inform every device on the network of the change in network topology. The NCM_RS DLPDU is stored in the RT-queue and broadcast by the DLPM using the MA-DATA request primitive.

6.5.5.5 Receiving

The broadcast DL–entity identifier is used as the destination address in an NCM_RS DLPDU. Therefore, the NCM_RS DLPDU is directly forwarded by the MAC layer. The NCM_RS DLPDU is processed by the DLM as follows:

a) extract the DPI (see 4.6.7) from the DLMDU and use it to check for a DL–entity identifier collision. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL–entity identifier collision is detected, set the DEV_FLAG (see 4.6.5.3) to "Local DL–entity identifier Collision," and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL–entity identifier collision is detected, set the NET_FLAG (see 4.6.6.11) to "Network DL–entity identifier Collision Status," and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive;

b) check the DLM state trigger event condition using the PORT_INFO and perform the DLM state transition (see 7.3.3);

c) if the DLM state is not RNMP or RNMS, enable the frame forward functions and set "frame forwarding function from R-port1 to R-port2" and "Frame Forwarding Function from R-port2 to R-port1" in PORT_INFO (see 4.6.5.9) to TRUE.

d) when the device is designated as the RNMS by the RNMP, disable the frame forwarding function in the RNMP direction;

e) enter the RNMS state if the frame forwarding control is successfully completed. Otherwise, remain in the GD state;

f) when the DLM enters the RNMS state, send NCM_ACK_RNMS to the RNMP.

6.5.6 NCM_AR_DLPDU

The NCM_AR DLPDU is used to indicate that the designated RNMS has been successfully assigned and is operational. This DLPDU is generated by the device that is selected as the RNMS and transferred to the RNMP.

6.5.7 NCM_AR DLPDU structure

6.5.7.1 General

See 6.5.2.2.

6.5.7.2 NCM_AR DLPDU parameters

Table 57 shows the NCM_AR DLPDU parameter list.

Table 57 – NCM_AR DLPDU parameters

Parameter	Data type	Description
DST_addr	UNSIGNED16	DL–entity identifier of RNMP
SRC_addr	UNSIGNED16	Local DL–entity identifier DL_ADDR (see 4.6.5.2)
NCMT	UNSIGNED8	NCM_ACK_RNMS (see IEC 61158-3-21, 4.4.3.2.4)
DLMDU	—	Local device information (see 4.6.5)
Length	UNSIGNED16	Length of DLMDU
R-port	UNSIGNED8	Destination R-port in the Path table

6.5.7.3 Sending

When the state of the DLM changes from GD to RNMS, the DLM generates an NCM_AR DLPDU to indicate that the designated RNMS is successfully assigned, and unicasts it to the RNMP.

6.5.7.4 Receiving

The NCM_AR DLPDU is unicast from the RNMS to the RNMP. The NCM_RS DLPDU is processed by the DLM in the RNMP device as follows:

a) extract the DPI (see 4.6.7) from the DLMDU and use it to check for a DL–entity identifier collision. If the source address is not registered in the DPI, generate an EVENT_IN_DEVICE event. If a local DL–entity identifier collision is detected, set the DEV_FLAG (see 4.6.5.3) to "Local DL–entity identifier Collision," and generate an EVENT_THIS_ADDR_COLLISION event. If a network DL–entity identifier collision is detected, set the NET_FLAG (see 4.6.6.11) to "Network DL–entity identifier Collision Status," and generate an EVENT_NET_ADDR_COLLISION event. Notification of any generated event is sent to the local DLMS-user using the DLM-EVENT indication service primitive;

b) Disable the frame forwarding function in the RNMS direction.

7 DLE elements of procedure

7.1 Overall structure

The DLL is composed of the control elements of the DLPM, the dual MAC (DMAC), the dual physical interface (DPHY), and the DLL management Interface. The DLPM is the primary control element. It provides the functions for deterministic MAC by coordinating the DMAC and the NCMs for reliable and efficient support both of higher-level connectionless real-time and non-real-time data transfer services. The DLL management interface provides DLL management functions. Figure 20 depicts the overall structure of the DLL.

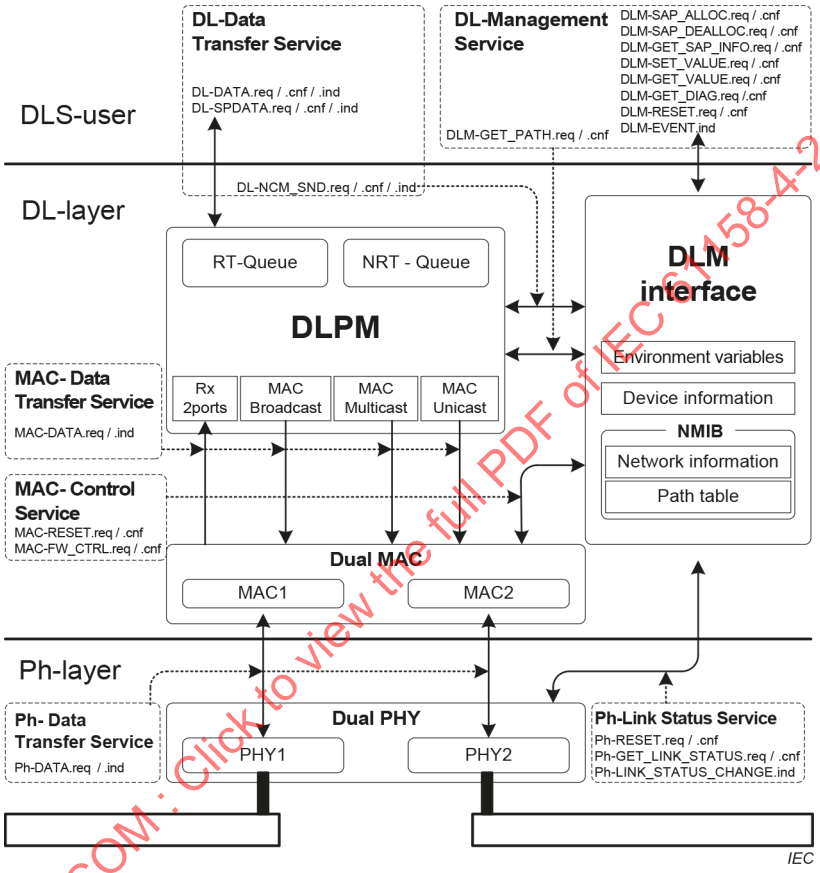


Figure 20 – DLL structure and elements

7.2 DL-protocol machine (DLPM)

7.2.1 Overview

The DLPM maintains two transmitter queues and one receiver queue. When a local DLS-user or DLMS-user generates a message, the message is stored in the RT-queue or the NRT-queue in the form of a DLPDU. The DLPM handles the messages in the RT-queue according to the DLPMSP. The DLPM transmits the message using the MA-DATA request primitive. When a message is received by the MAC layer, the received message is stored in the receiver queue and processed on a first-in, first-out basis by the DLPM.

7.2.2 Primitive definitions

7.2.2.1 Primitives exchanged between DLPM and DLS-user

Table 58 shows the data link service primitives exchanged between the DLPM and the DLS-user.

Table 58 – Primitives exchanged between DLPM and DLS-user

Primitive	Source	Associated parameters	Description
DL-DATA.req	DLS-user	DST_addr DSAP SSAP Priority Group mask Group mask length DLSDU DLSDU length Extension type Extension data Extension length	Transmit request to remote Type 21 device
DL-DATA.cnf	DLPM	Status	Confirmation to calling DLS-user
DL-DATA.ind	DLPM	DST_addr SRC_addr DSAP SSAP Type of Service(ToS) DLSDU DLSDU length Group receive Group length Group data Extension type Extension data Extension length Receive R-port	Receive indication from a remote Type 21 device
DL-SPDATA.req	DLS-user	DLSDU DLSDU length	Sporadic data transmit request to remote Type 21 device
DL-SPDATA.cnf	DLPM	Status	Confirmation to calling DLS-user
DL-SPDATA.ind	DLPM	DLSDU DLSDU length	Sporadic data receive indication from a remote Type 21 device

Table 59 shows the parameters exchanged between the DLPM and the DLS-user.

Table 59 – Parameters exchanged between DLPM and DLS-user

Parameter	Description
DST_addr	Destination DL–entity identifier. C_BROADCAST_ADDR: broadcast address 0xFF User-defined multicast address: 0xFD–0xDE 0-MAX_DEVICE_ADDR: unicast address
DSAP	Destination service access point
SSAP	Source service access point
Priority	Message priority. Permissible values are C_PRI_0 through C_PRI_3 C_PRI_3 is the highest priority.
DLSDU	DLSDU
DLSDU length	Length of DLSDU
Status	This parameter allows the DLMS-user to determine whether the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter is one of: "OK – success – the variable could be updated"; "Failure – the variable does not exist or could not assume the new value"; "Failure – invalid parameters in the request".

NOTE C_NCM_ADDR is not used for DL-DATA and DL-SPDATA services.

7.2.2.2 Primitives exchanged between DLPM and DLM

Table 60 and Table 61 show the data link service primitives and parameters exchanged between the DLPM and the DLM.

Table 60 – Primitives exchanged between DLPM and DLM

Primitive	Source	Associated parameters	Description
DLM-GET_PATH.req	DLPM	DST_addr	Path information request to DLM
DLM-GET_PATH.cnf	DLM	Status R-port MAC address	Confirmation to calling DLPM with R-port and MAC address
DLM-NCM_SND.req	DLM	DST_addr NCMT DLMDU Length R-port	Transmit request for network control message NCM_FAMILY_REQ NCM_FAMILY_RES NCM_MEDIA_LINKED NCM_ADV_THIS NCM_LINE_START NCM_RING_START NCM_ACK_RNMS NCM_RETRY_RNMS
DLM-NCM_SND.cnf	DLPM	Status	Confirmation to calling service user
DLM-NCM_SND.ind	DLPM	DST_addr SRC_addr NCMT DLMDU length R-port	Network control message receive indication from a remote Type 21 device

Table 61 – Parameters used with primitives exchanged between DLPM and DLM

Parameter	Description
DST_addr	Destination DL–entity identifier. C_BROADCAST_ADDR: broadcast address, 0xFF C_NCM_ADDR: Fixed DL–entity identifier for network control message, 0xFE 0-MAX_DEVICE_ADDR: unicast address
SRC_addr	Source DL–entity identifier. "0–MAX_DEVICE_ADDR: unicast address"
R-port	R-port which is used to transmit a frame or to receive a frame.
MAC address	ISO/IEC/IEEE 8802-3 MAC address that corresponds to the Destination DL–entity identifier.
NCMT	Network Control Message Type. NCM_FAMILY_REQ (0x01), NCM_FAMILY_RES (0x02), NCM_MEDIA_LINKED (0x03), NCM_ADV_THIS (0x04), NCM_LINE_START (0x05), NCM_RING_START (0x06), NCM_ACK_RNMS (0x07), NCM_CHECK_NET_INTEGRITY (0x08), NCM_CHECK_NEIGHBOR_INTEGRITY (0x09), NCM_NET_IS_RING (0x0A), NCM_RENEGO_REQ (0x0B)
DLMDU	DL management data unit. The data field of DLM network control message.
Length	Length of DLMDU
Status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of: "OK – success – the variable could be updated"; "Failure – the variable does not exist or could not assume the new value"; "Failure – invalid parameters in the request".

NOTE User-defined multicast addresses are not used for the destination DL–entity identifier.

7.2.3 DLPM state table

Figure 21 depicts the state transition diagram of the DLPM. The state table of the DLPM is shown in Table 62.

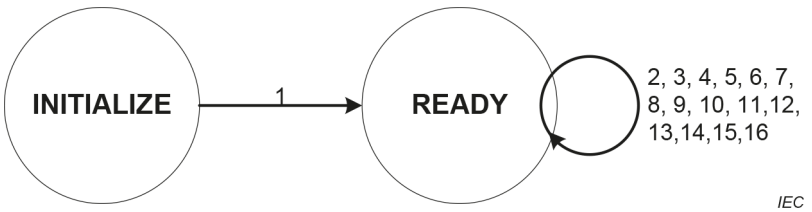


Figure 21 – State transition diagram of the DLPM

Table 62 – DLPM state table

#	Current	Event /Condition =>actions	Next state
1	INITIALIZE	POWER-ON or RESET / =>	READY
2	READY	DL-DATA.req {DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) = "True" && DLM-GET_PATH.req {DST_addr} Status:= DLM-GET_PATH.cfm { D_MAC_addr, R-port } Status ="Success" && CHECK_RTQUEUE( ) <> "Full" => DLPDU:= BUILD_DLPDU(D_MAC_addr, DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length) ENQUEUE_RT(Priority, DLPDU, DLPDU length, R-port) DL-DATA.cfm{Status:= "Success"}	READY
3	READY	DL-DATA.req {DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) <> " True" => DL-DATA.cfm{Status:= "Failure – Invalid parameter"}	READY
4	READY	DL-DATA.req {DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) = " True" && DLM-GET_PATH.ind { D_MAC_addr, R-port} ="Failure" => DL-DATA.cfm{Status:= "Failure – not available destination"}	READY
5	READY	DL-DATA.req { DST_addr, DSAP, SSAP, Priority, DLSDU, DLSDU length } / CHECK_PARA(DLSDU length) = " True" && DLM-GET_PATH.ind { D_MAC_addr, R-port } =" Success" && CHECK_RTQUEUE( ) = "Full" => DL-DATA.cfm{Status:= "Failure – The RT-queue is full"}	READY
6	READY	DL-SPDATA.req {DLSDU, DLSDU length} / CHECK_PARA(DLSDU length) = "True" && CHECK_NRTQUEUE() <> "Full" => DLPDU:= DLSDU R-port:= R-port1 && R-port2 ENQUEUE_NRT(DLPDU, DLPDU length, R-port) DL-SPDATA.cfm {Status:= "Success"}	READY

#	Current	Event /Condition =>actions	Next state
7	READY	DL-SPDATA.req {DLSDU, DLSDU length} / CHECK_PARA(DLSDU length) = "False" => DL- SPDATA.cfm{Status:= "Failure – Invalid requested parameter"}	READY
8	READY	DL-SPDATA.req {DLSDU, DLSDU length} / CHECK_PARA(DLSDU length) = "True" && CHECK_NRTQUEUE( ) = "Full" => DL- SPDATA.cfm{Status:= "Failure – The NRT-queue is full"}	READY
9	READY	DL-NCM_SND.req {DST_addr, NCMT, DLMDU, Length, R-port} / CHECK_PARA(Length) = "True" && CHECK_RTQUEUE( ) <> "Full" => Priority:= C_HIGHEST_PRIORITY DLPDU:= BUILD_DLPDU_NCM(DST_addr , Priority, NCMT , DLMDU, Length) QUEUE_RT(Priority, DLPDU, DLPDU length, R-port) DL-NCM_SND.cfm{Status:= "Success"}	READY
10	READY	DL-NCM_SND.req {DST_addr, NCMT, DLMDU, Length, R-port} / CHECK_PARA(Length) = "False" => DL- NCM_SND.cfm{Status:= "Failure – Invalid parameter"}	READY
11	READY	DL-NCM_SND.req {DST_addr, NCMT, DLMDU, Length, R-port} / CHECK_PARA(Length) = "True" && CHECK_RTQUEUE( ) = "Full" => DL- NCM_SND.cfm{Status:= "Failure – The RT-queue is full"}	READY
12	READY	MA-DATA.ind{DLPDU, DLPDU length, R-port} / CHECK_FRAME_ETH_TYPE(DLPDU) = "Type 21" && CHECK_FRAME_FOR_THIS(DLPDU) = "True" && CHECK_FC(DLPDU) <> "Network control message" => DST_addr:= GET_D_ADDR(DLPDU) SRC_addr:= GET_S_ADDR(DLPDU) DSAP:= GET_DSAP(DLPDU) SSAP:= GET_SSAP(DLPDU) DLSDU:= GET_DLSDU(DLPDU) DLSDU length:= GET_DLSDU(DLPDU length) DLS-user:= GET_DLS_USER(DSAP) dst_dls_user:= SET_DLS_USER(DLS-user) dst_dls_user.DL-DATA.ind{ DST_addr, SRC_addr, SSAP, DLSDU, DLSDU length}	READY

#	Current	Event /Condition =>actions	Next state
13	READY	MA-DATA.ind{DLPDU, DLPDU length, R-port} / CHECK_FRAME_ETH_TYPE(DLPDU) <> "Type 21" => DLSDU:= DLPDU DLSDU length:= DLPDU length DL-SPDATA.ind{DLSDU, DLSDU length}	READY
14	READY	MA-DATA.ind{DLPDU, DLPDU length, R-port} / CHECK_FRAME_ETH_TYPE(DLPDU) = "Type 21" && CHECK_FRAME_FOR_THIS(DLPDU) = "True" && CHECK_FC(DLPDU) = "Network Control Message" => DST_addr:= GET_D_ADDR(DLPDU) SRC_addr:= GET_S_ADDR(DLPDU) DLMDU:= GET_DLSDU(DLPDU) length:= GET_DLSDU(DLPDU length) cmd:= GET_NCM_CMD(DLPDU) DL-NCM_SND.ind{DST_addr, SRC_addr, NCMT, DLMDU, DLMDU length, R-port}	READY
15	READY	New DLPDU from DLS-user / CHECK_RTQUEUE( ) <> "Empty" => DLPDU:= DQUEUE_RT(policy) MA-DATA.req {DLPDU, DLPDU length, R-port}	READY
16	READY	New DLPDU from DLS-user / CHECK_RTQUEUE( ) = "Empty" && / CHECK_NRTQUEUE( ) <> "Empty" => DLPDU:= DQUEUE_NRT( ) MA-DATA.req {DLPDU, DLPDU length, R-port}	READY

7.2.4 DLPM functions

All functions of the DLPM are summarized in Table 63.

Table 63 – DLPM functions table

Function name	Input	Output	Operation
CHECK_PARA	DLSDU length	True/False	Return FALSE if the DLSDU exceeds the C_MAX_USER_DLMDU_SIZE. Otherwise, return TRUE.
CHECK_RTQUEUE	(none)	status	Check the RT-queue condition. The returned status is "Full," "Empty" or "Queued."
BUILD_DLPDU	D_MAC_addr DST_addr DSAP SSAP Priority DLSDU DLSDU length	DLPDU DLPDU length	Build DLPDU and return the DLPDU and its length.
ENQUEUE_RT	Priority DLPDU DLPDU length R-port	(none)	Queues the input data into the tail of the RT-queue.
CHECK_NRTQUEUE	(none)	status	Check that the NRT-Queue condition for DATA is fully queued. The returned status is "Full," "Empty" or "Queued."
ENQUEUE_NRT	DLPDU DLPDU length R-port	(none)	Queues the input data into the NRT-QUEUE on a FIFO basis.
BUILD_DLPDU_NCM	DST_addr Priority NCMT DLMDU DLMDU length	DLPDU DLPDU length	Build NCM DLPDU and return the DLPDU and its length.
DQUEUE_RT	Policy	DLPDU DLPDU length R-port	Get a DLPDU from the RT-queue as specified by the DLPMSP.
DQUEUE_NRT	(none)	DLPDU DLPDU length R-port	Get a DLPDU from the NRT-queue according to the FIFO method.
CHECK_FRAME_ETH_TYPE	DLPDU	status	Check Type 21 EtherType (0x88FE). Return "Type 21" when the type is correct. Otherwise, return "Other."
CHECK_FRAME_FOR_THIS	DLPDU	True/False	Check if the frame has been unicast to local device. Return TRUE if the Destination DL–entity identifier is equal to the Local DL–entity identifier. Otherwise, return FALSE.
CHECK_FC	DLPDU	status	Check if the received frame is a network control message. Return "Network control message" if the FC field indicates NCM. Otherwise, return "Other."
GET_D_ADDR	DLPDU	DST_addr	Return the Destination DL–entity identifier of the DLPDU.

Function name	Input	Output	Operation
GET_S_ADDR	DLPDU	SRC_addr	Return the Source DL–entity identifier of the DLPDU.
GET_DSAP	DLPDU	DSAP	Return the Destination SAP of the DLPDU.
GET_SSAP	DLPDU	SSAP	Return the Source SAP of the DLPDU.
GET_DLSDU	DLPDU	DLSDU	Return DLSDU of the DLPDU.
GET_DLSDU LENGTH	DLPDU length	DLSDU length	Return the DLSDU length.
GET_DLS_USER	DSAP	DLS-user ID	Return the DLS-user ID that owns the DSAP.
SET_DLS_USER	DLS-user ID	DLS-user object	Return the DLS-user object using the DLS-user ID.
GET_NCMT	DLPDU	NCMT	Return the NCMT of the NCM DLPDU

7.3 DLL management Protocol

7.3.1 Overview

This clause describes the interface protocol between the DLM and DLMS-user. This description of the DLL management protocol provides the DLL management services specified in Clause 7 by making use of the services available to the DLMS-user. Full implementation details and matters of local responsibility are not included in this clause.

7.3.2 Primitive definitions

7.3.2.1 Primitive exchanged between DLM and DLS-user

Table 64 summarizes all primitives exchanged between the DLM and the DLS-user.

Table 64 – Primitives exchanged between DLM and DLS-user

Primitive	source	Associated parameters	Description
DLM-RESET.req	DLS-user	<none>	This request primitive causes the DLM to reset the DLE.
DLM-RESET.cnf	DLM	DLM-status	This indicates the status of the reset.
DLM-SET_VALUE.req	DLS-user	Variable name Desired value	This service is used to assign new values to the DLE variables.
DLM-SET_VALUE.cnf	DLM	Status	The DLMS-user receives confirmation that the specified variables have been set to the new values.
DLM-GET_VALUE.req	DLS-user	Variable name	This service is used to read the value of a DLE variable.
DLM-GET_VALUE.cnf	DLM	Status Current value	This service returns the actual value of the specified variable.
DLM-SAP_ALLOC.req	DLS-user	SAP DLS-user ID	This service is used by the DLMS-user to obtain a SAP assignment from the DLM.
DLM-SAP_ALLOC.cnf	DLM	Status	This service returns the result status of DLM-SAP_ALLOC.req.
DLM-SAP_DEALLOC.req	DLS-user	SAP	This service is used by the DLMS-user to release and return the allocated SAP to the DLM.
DLM-SAP_DEALLOC.cnf	DLM	Status	This service returns the result status of DLM-SAP_DEALLOC.req.
DLM-GET_SAP_INFO.req	DLS-user	SAP	This service is used by the DLMS-user to obtain the information of already allocated SAP from the DLM.
DLM-GET_SAP_INFO.cnf	DLM	Status DLS-user ID	This service returns the result status of DLM-GET_SAP_INFO.req, specifically the result status and DLS-user ID.
DLM-GET_DIAG.req	DLS-user	Diag-type Addr	This service is used by the DLMS-user to obtain the diagnostic information from the DLM.
DLM-GET_DIAG.cnf	DLM	Status Diag	This service returns the result status of DLM-GET_SAP_INFO.req, specifically the result status and diagnostic information.
DLM-EVENT.ind	DLM	Event	This service is used to inform the DLMS-user about certain events or errors in the DLL.

The parameters used with the primitives exchanged between the DLM and the DLS-user are described in Table 65.

Table 65 – Parameters used with primitives exchanged between DLM and DLS-user

Parameter	Description
DLM-status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. It it failed, the reason is specified. The value of this parameter can be one of: "OK – successfully completed" "Failure – terminated before completion"
Variable name	This parameter specifies the DLE variable whose value is to be set.
Desired value	This parameter specifies the desired value for the selected variable.
Status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of: "OK – success – the variable could be updated" "Failure – the variable does not exist or could not assume the new value" "Failure – invalid parameters in the request"
Current value	This parameter indicates the current value of the designated variable.
DLS-user ID	This parameter indicates the numeric identification of the local DLS-user. DLS-user ID is unique in a device.
SAP	This parameter indicates the DLSAP.
Diag-type	This parameter indicates the type of diagnostic information. "DIAG_TYPE_L_DEVICE_INFO – local device information" "DIAG_TYPE_R_DEVICE_INFO – remote device information" "DIAG_TYPE_NET_INFO – network information"
Addr	This parameter indicates the DL–entity identifier of the designated node.
Event	This parameter specifies the primitive or composite event being announced. The possible values are defined in the corresponding part of Subclause 8.2.

7.3.2.2 Primitives exchanged between DLM and DLPM

Table 60 summarizes all primitives exchanged between the DLM and the DLPM.

7.3.2.3 Primitives exchanged between DLM and DMAC

Table 66 and Table 67 summarize all primitives and parameters exchanged between the DLM and the DMAC.

Table 66 – Primitive exchanged between DLM and DMAC

Primitive name	Source	Associated parameters	Description
MAC-RESET.req	DLM	<none>	Reset MAC layer.
MAC-FW_CTRL.req	DLM	R-port F_en	This service is used to control the frame forward functions in the MAC layer.
MAC-FW_CTRL.cnf	D-MAC	Status	This service returns the result status of MAC-FW_CTRL.req.

Table 67 – Parameters used with primitives exchanged between DLM and DMAC

Parameter name	Description
R-port	This parameter indicates the Type 21 Ethernet port. Every Type 21 device has two R-ports: R-port1 and R-port2.
F_en	This parameter indicates the frame forward control status between the two R-ports: "Enable - enable the hardware-based frame forward function" "Disable - disable the hardware-based frame forward function"
Status	This parameter allows the DLMS-user to determine whether or not the requested DLMS was provided successfully. If it failed, the reason is specified. The value of this parameter can be one of: "OK – success – the variable could be updated"; "Failure – the variable does not exist or could not assume the new value"; "Failure – invalid parameters in the request".

7.3.2.4 Primitive exchanged between DLM and DPHY

Table 68 and Table 69 summarize all primitives and parameters exchanged between the DLM and the DPHY.

Table 68 – Primitive exchanged between DLM and DPHY

Primitive name	Source	Associated Parameters	Description
Ph-RESET.req	DLM	<none>	Reset physical layer.
Ph-GET_LINK_STATUS.req	DLM	R-port	This service is used to obtain link status information from the physical layer.
Ph-GET_LINK_STATUS.cnf	DPHY	R-port L_status	This service returns the result status of Ph-GET_LINK_STATUS.req
Ph-LINK_STATUS_CHANGE.ind	DPHY	R-port	This service is used to notify the DLM of the link status change event.

Table 69 – Parameters used with primitives exchanged between DLM and DPHY

Parameter name	Description
R-port	This parameter indicates the Type 21 Ethernet port. Every Type 21 device has two R-ports: R-port1 and R-port2.
L_status	"link active – The link is activated and available for communication" "link inactive – The link is not activated and not available for communication"

7.3.3 DLM state table

The DLM is maintained with 6 states: INITIALIZE, SA, LNM, GD, RNMP, and RNMS. The DLM controls the frame forward functions in the MAC layer according to its DLM state. DLM state transition diagram and descriptions are shown in Figure 22 and Table 70.

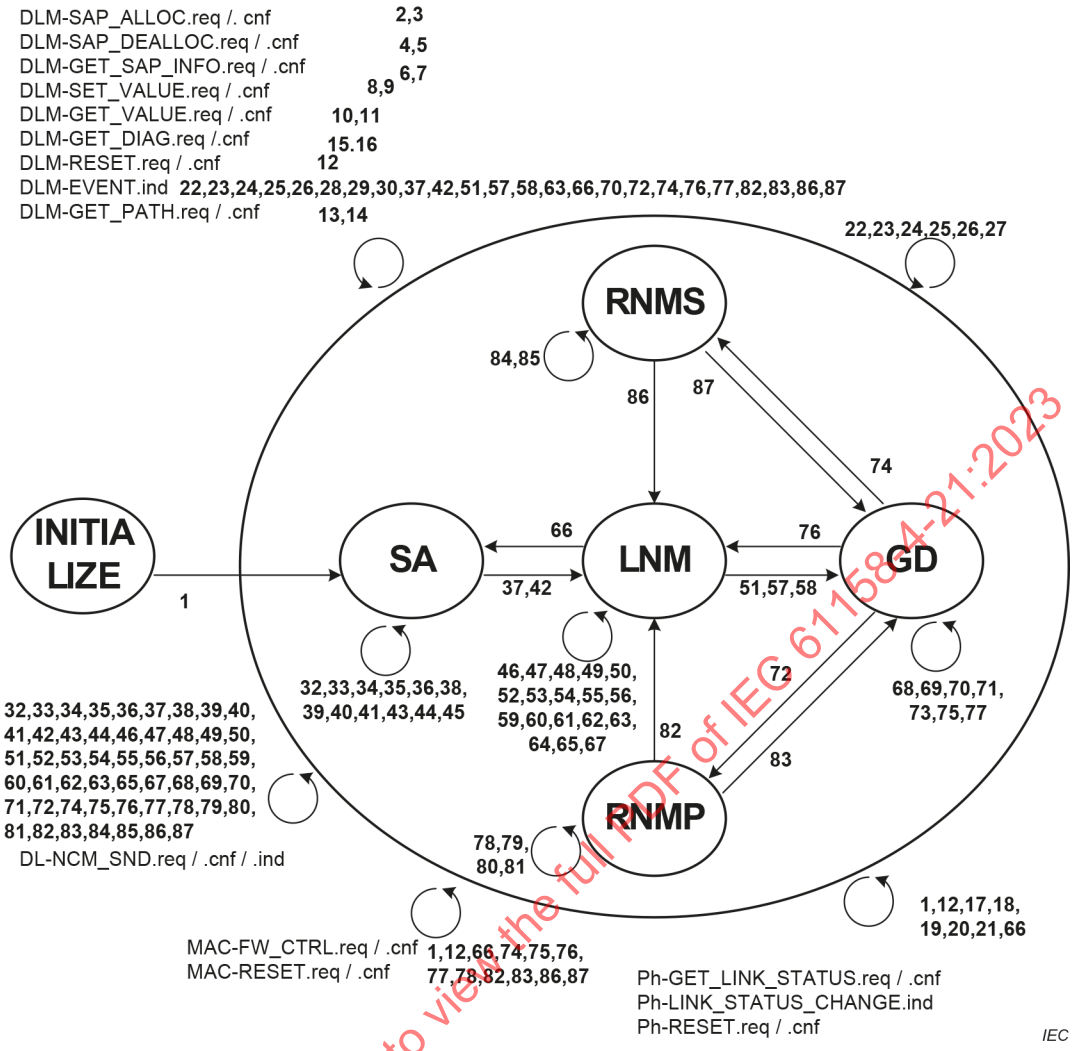


Figure 22 – State transition diagram of DLM

SA (standalone)

This state means that the local initialization procedures have been successfully completed and the device is ready to be linked to the other devices. In the SA state, the DLM tries to find the other devices on the network. When a link is established, the state changes to the LNM state.

LNM (line network manager)

This state means that the local device is located at the end of a line network. The LNM device is linked on the line network with one of its two R-ports. Both frame forward functions are disabled in the LNM device.

GD (general device)

This state means that the local device is connected to the network through both its R-ports. Both frame forward functions are enabled in the GD device. However, the frame forward functions in the GD device are suspended until the device receives NCM_LS DLPDU or NCM_RS DLPDU.

RNMP (primary ring network manager)

This state means that the local device is automatically selected as the primary ring network manager in a ring network. The RNMP device selects one of its neighboring devices as the secondary ring network manager (RNMS) using the NCM_RS DLPDU. The RNMP device disables the frame forward function in the RNMS direction but keeps the frame forward function in the other direction enabled.

RNMS (secondary ring network manager)

This state means that the local device is selected as the secondary ring network manager in a ring network. The RNMS device disables the frame forward function in the RNMP direction but keeps the frame forward function in the other direction enabled.

Table 70 – DLM state table

#	Current	Event / Condition =>actions	Next state
1	INITIALIZE	POWER-ON or RESET / => INIT_ENV_VAR( ) INIT_SAP_INFO( ) INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) MAC-RESET.req { } Ph-RESET.req { } SET_BLOCK_PORT(INVALID_R_PORT) CLEAR_PORT_INFO(R-port1) CLEAR_PORT_INFO(R-port2) FW_CTRL(R-port1, DISABLE) FW_CTRL(R-port2, DISABLE) Generate_Alarm(EVENT_THIS_STATE_CHG) DEV_STATE:=SA STORE_PATH_INFO(DEV_ADDR)	SA
		Events triggered by the DL-Management service are listed below.	
2	Any state	DLM-SAP_ALLOC.req {SAP, DLS-user ID} / CHECK_ALLOC_SAP(SAP) = "True" => ALLOC_SAP(SAP, DLS-user ID) Status:= "success" DLM-SAP_ALLOC.cnf {Status}	Any state
3	Any state	DLM-SAP_ALLOC.req {SAP, DLS-user ID} / CHECK_ALLOC_SAP(SAP) <> "True" => Status:= "Failure – SAP is already allocated to another DLS-user" DLM-SAP_ALLOC.cnf {Status}	Any state

#	Current	Event / Condition =>actions	Next state
4	Any state	DLM-SAP_DEALLOC.req {SAP} / CHECK_DEALLOC_SAP(SAP) = "True" => DEALLOC_SAP(SAP) Status:= "success" DLM-SAP_DEALLOC.cnf {Status}	Any state
5	Any state	DLM-SAP_DEALLOC.req {SAP} / CHECK_DEALLOC_SAP(SAP) <> "True" => Status:= "Failure – SAP is not allocated to a DLS-user" DLM-SAP_DEALLOC.cnf {Status}	Any state
6	Any state	DLM-GET_SAP_INFO.req {SAP} / CHECK_ALLOCEDSAP(SAP) = "True" => DLS-user ID:= GET_USERID_FOR_SAP(SAP) Status:= "Success" DLM-GET_SAP_INFO.cnf {Status, DLS-user ID}	Any state
7	Any state	DLM-GET_SAP_INFO.req {SAP} / CHECK_ALLOCEDSAP(SAP) <> "True" => DLS-user ID:= INVALID_USER_ID Status:= "Failure – SAP is not allocated to a DLS-user" DLM-GET_SAP_INFO.cnf {Status, DLS-user ID}	Any state
8	Any state	DLM-SET_VALUE.req {variable name, desired value} / CHECK_VALUE(variable name, desired value) = "valid" => SET_VALUE(variable name, desired value) Status:= "success" DLM-SET_VALUE.cnf {Status }	Any state
9	Any state	DLM-SET_VALUE.req {variable name, desired value} / CHECK_VALUE(variable name, desired value) <> "valid" => Status:= "Failure – invalid parameters in the request" DLM-SET_VALUE.cnf {Status }	Any state
10	Any state	DLM-GET_VALUE.req {variable name} / CHECK_VAR(variable name) = "valid" => Current value:= GET_CURRENT_VAL(variable name) Status:= "success" DLM-GET_VALUE.cnf {Status , Current value }	Any state

#	Current	Event / Condition =>actions	Next state
11	Any state	DLM-GET_VALUE.req {variable name} / CHECK_VAR(variable name) <> "valid" => Current value:= INVALID_VALUE Status:= "Failure – invalid parameters in the request" DLM-GET_VALUE.cnf {Status , Current value }	Any state
12	Any state	DLM-RESET.req { } / => INIT_ENV_VAR( ) INIT_SAP_INFO( ) INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) MAC-RESET.req { } Ph-RESET.req { } SET_BLOCK_PORT(INVALID_R_PORT) CLEAR_PORT_INFO(R-port1) CLEAR_PORT_INFO(R-port2) Status:= "success" DLM-RESET.cnf {Status} FW_CTRL(R-port1, DISABLE) FW_CTRL(R-port2, DISABLE) Generate_Alarm(EVENT_THIS_STATE_CHG) DEV_STATE:=SA STORE_PATH_INFO(DEV_ADDR)	SA
13	Any state	DLM-GET_PATH.req {Addr} / CHECK_ADDR(Addr) = "valid" => R-port:= GET_DST_PORT(addr) D_MAC_addr:= GET_DST_MAC_ADDR(addr) Status:= "success" DLM-GET_PATH.cnf {Status, R-port, D_MAC_Addr}	Any state
14	Any state	DLM-GET_PATH.req {Addr} / CHECK_ADDR(Addr) <> "valid" => R-port:= INVALID_R_PORT D_MAC_addr:= INVLID_MAC_ADDR Status:= "Failure – invalid parameters in the request" DLM-GET_PATH.cnf {Status, R-port, D_MAC_Addr}	Any state

#	Current	Event / Condition =>actions	Next state
15	Any state	DLM-GET_DIAG.req {Diag-type, addr} / CHECK_DIAG_TYPE(Diag-type, addr) = "True" => Status:= "success" Diag-info:= GET_DIAG_INFO(Diag-type) DLM-GET_DIAG.cnf {Status, Diag-info}	Any state
16	Any state	DLM-GET_DIAG.req {Diag-type, addr} / CHECK_DIAG_TYPE (Diag-type, addr) <> "True" => Status:= "Failure – invalid parameters in the request" DLM-GET_DIAG.cnf {Status, Diag-info}	Any state
		Events triggered by the DLM event generator are listed below.	
17	Any state	Ph_LINK_STATUS_CHANGE.ind {R-port} / => Ph-GET_LINK_STATUS.req {R-port}	Any state
18	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status} / L_status = "link active" && CHECK_NEWLY_LINK_ACTV(R-port) = "True" => NEWLY_LINK_ACTV(R-port)	Any state
19	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status} / L_status = "link active" && CHECK_NEWLY_LINK_ACTV(R-port) <> "True" => (none)	Any state
20	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status} / L_status = "link inactive" && CHECK_NEWLY_LINK_INACTV(R-port) = "True" => NEWLY_LINK_INACTV(R-port)	Any state
21	Any state	Ph-GET_LINK_STATUS.cnf {R-port, L_status} / L_status = "link inactive" && CHECK_NEWLY_LINK_INACTV(R-port) <> "True" => (none)	Any state

#	Current	Event / Condition =>actions	Next state
		Device information-related procedures are listed below.	
22	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_NET_ADDR_COLLISION(DLMDU) <> "True" && CHECK_THIS_ADDR_COLLISION(DLMDU) = "True" && CHECK_NEWLY_IN_DEVICE(DLMDU) = "True" => events:= EVENT_THIS_ADDR_COLLISION | EVENT_IN_DEVICE DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
23	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) = "True" && CHECK_THIS_ADDR_COLLISION(DLMDU) <> "True" && CHECK_NEWLY_IN_DEVICE(DLMDU) = "True" => events:= EVENT_NET_ADDR_COLLISION | EVENT_IN_DEVICE DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
24	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) = "True" && CHECK_THIS_ADDR_COLLISION(DLMDU) <> "True" && CHECK_NEWLY_IN_DEVICE(DLMDU) <> "True" => events:= EVENT_NET_ADDR_COLLISION DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
25	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) <> "True" && CHECK_THIS_ADDR_COLLISION(DLMDU) = "True" && CHECK_NEWLY_IN_DEVICE(DLMDU) <> "True" => events:= EVENT_THIS_ADDR_COLLISION DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
26	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_ NET_ADDR_COLLISION(DLMDU) <> "True" && CHECK_THIS_ADDR_COLLISION(DLMDU) <> "True" && CHECK_NEWLY_IN_DEVICE(DLMDU) = "True" => events:= EVENT_IN_DEVICE DLM-EVENT.ind(events) SAVE_DEV_INFO(DLMDU, length, R-port)	Any state

#	Current	Event / Condition =>actions	Next state
27	Any state	STORE_DEV_INFO(DLMDU, length, R-port) / CHECK_NET_ADDR_COLLISION(DLMDU) <> "True" && CHECK_THIS_ADDR_COLLISION(DLMDU) <> "True" && CHECK_NEWLY_IN_DEVICE(DLMDU) <> "True" => SAVE_DEV_INFO(DLMDU, length, R-port)	Any state
		Path table update procedures are listed below	
28	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) = "valid" && CHECK_THIS_ADDR_COLLISION_CLEAR(UID) <> "True" && CHECK_NET_ADDR_COLLISION_CLEAR(UID) <> "True" => DEL_DEV(UID) Events:= EVENT_OUT_DEVICE DLM-EVENT.ind(Events)	Any state
29	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) = "valid" && CHECK_THIS_ADDR_COLLISION_CLEAR(UID) = "True" && CHECK_NET_ADDR_COLLISION_CLEAR(UID) <> "True" => DEL_DEV(UID) Events:= EVENT_OUT_DEVICE | EVENT_THIS_ADDR_COLLISION_CLEAR DLM-EVENT.ind(Events)	Any state
30	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) = "valid" && CHECK_THIS_ADDR_COLLISION_CLEAR(UID) <> "True" && CHECK_NET_ADDR_COLLISION_CLEAR(UID) = "True" => DEL_DEV(UID) Events:= EVENT_OUT_DEVICE | EVENT_NET_ADDR_COLLISION_CLEAR DLM-EVENT.ind(Events)	Any state
31	Any state	DELETE_DEV_INFO(Device UID) / CHECK_UID(UID) <> "valid" => (<none>)	Any state
		State transitions are listed below.	
32	SA	Phy_Link_Change.Ind(R-port, Link_status) / Link status == PHY_LINK_UP => Set_Block_Port(Toggle_Port(R-port)) Start_Timer(RRP_FamilyReqT, R-port) Clear_FamilyReqRetryCnt(R-port) Family_Frame.Req(R-port)	SA

#	Current	Event / Condition =>actions	Next state
33	SA	Family_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /Chk_Port_Info(R-port, PORT_RCV_FAMILY_CNF) == FALSE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port) Set_Port_Info(R-port, PORT_SNT_FAMILY_RES)	SA
34	SA	Family_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /Chk_Port_Info(R-port,PORT_RCV_FAMILY_CNF) == TRUE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port) Set_Port_Info(R-port, PORT_FAMILY_CFM) DEV_STATE:= LNM RRP_NET_TPG:= NET_TPG_LINE Set_LNM_UID((Toggle_Port(R-port), DEV_UID) MediaLink.Req(R-port) Start_Timer(RRP_MediaLinkedT, R-port) Generate_Alarm(EVENT_THIS_STATE_CHG) Generate_Alarm(EVENT_NET_TPG_CHG)	LNM
35	SA	Timer(RRP_FamilyReqT, R-port) expired / Chk_FamilyReqRetryCnt(R-port, FM_REQ_RETRY_CNT)==FALSE => Increase_FamilyReqRetryCnt(R-port) Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)	SA
36	SA	Timer(RRP_FamilyReqT, R-port) expired / Chk_FamilyReqRetryCnt(R-port, FM_REQ_RETRY_CNT)==TRUE && Chk_LinkStatus(Toggle_Port(R-port), PHY_LINK_UP) == FALSE => Start_Timer(RRP_FamilyReqT, R-port) Family_Frame.Req(R-port)	SA
37	SA	Timer(RRP_FamilyReqT, R-port) expired / Chk_FamilyReqRetryCnt(R-port, FM_REQ_RETRY_CNT)==TRUE && ChkLinkStatue(Toggle_Port(R-port), LINK_UP) == TRUE => BlockPort(R-port) Put_Event(Toggle_Port(R-port), PHY_LINK_UP)	SA

#	Current	Event / Condition =>actions	Next state
38	SA	Family_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /Chk_Port_Info(R-port,PORT_SNT_FAMILY_RES) == FALSE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port)	SA
39	SA	Family_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /Chk_Port_Info(R-port, PORT_SNT_FAMILY_RES) == TRUE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Stop_Timer(RRP_FamilyReqT, R-port) Set_Port_Info(R-port, PORT_FAMILY_CFM) DEV_STATE:= LNM RRP_NET_TPG:= NET_TPG_LINE Set_LNM_UID((Toggle_Port(R-port), DEV_UID) MediaLink.Req(R-port) Start_Timer(RRP_MediaLinkedT, R-port) Generate_Alarm(EVENT_THIS_STATE_CHG) Generate_Alarm(EVENT_NET_TPG_CHG)	LNM
40	SA	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN => Set_Port_Info(R-port, PORT_LINK_DOWN) Set_Block_Port(INVALID_R_PORT) Clear_Port_Info(R-port1) Clear_Port_Info(R-port2) INIT_PATH_INFO( ) STORE_PATH_INFO(DEV_ADDR) Stop_AllTimer(R-port)	SA
41	LNM	"Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_UP => Start_Timer(RRP_FamilyReqT) Family_Frame.Req(R-port)"	LNM

#	Current	Event / Condition =>actions	Next state
42	LNM	Family_Frame.Ind (R-port, rcv-DEV_UID, rcv-DEV_ADDR) / rcv_DEV_UID != Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_RCV_FAMILY_CNF)==TRUE && Chk_Port_Info(Toggle_Port(R-port), PORT_CFM_FAMILY)==TRUE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port) Set_Port_Info(R-port, PORT_CFM_FAMILY) DEV_STATE:= GD Start_Timer(RRP_MediaLinkedT) MediaLinked_Frame.Req(R-port) Generate_Alarm(EVENT_THIS_STATE_CHG)	GD
43	LNM	Family_Frame.Ind (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID != Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_RCV_FAMILY_CNF)==FALSE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port)	LNM
44	LNM	"Family_Frame.Ind (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID != Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_CFM_FAMILY)==TRUE => Put_Event(R-port, PHY_LINK_DOWN)"	LNM
45	LNM	Family_Frame.Ind (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID == Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_RCV_FAMILY_CNF)==TRUE && Chk_Port_Info(Toggle_Port(R-port), PORT_CFM_FAMILY)==TRUE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Family_Frame.Res(R-port) Set_Port_Info(R-port, PORT_CFM_FAMILY) Set_Port_Info(R-port, PORT_CFM_COMPLETE) DEV_STATE:= GD RRP_NET_TPG:= NET_TPG_RING Clear_AllUid( ) Start_Timer(RRP_NetIsRingCheckT, R-Port) RRP_Net_Is_Ring_Frame.Req(Toggle_Port(R-port)) Generate_Alarm(EVENT_THIS_STATE_CHG)	GD

#	Current	Event / Condition =>actions	Next state
46	LNM	Family_Frame.Ind (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID == Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_RCV_FAMILY_CNF)==FALSE && Chk_Port_Info(Toggle_Port(R-port), PORT_CFM_FAMILY)==TRUE => Ignore	LNM
47	LNM	Timer(RRP_FamilyReqT, R-port) expired / => Start_Timer(RRP_FamilyReqT, R-Port) Family_Frame.Req(R-port)	LNM
48	LNM	Family_Frame.Cnf (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID != Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_SNT_FAMILY_RES)==TRUE && Chk_Port_Info(Toggle_Port(R-port), PORT_CFM_FAMILY)==TRUE => Stop_Timer(RRP_FamilyReqT, R-port) Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Set_Port_Info(R-port, PORT_CFM_FAMILY) DEV_STATE:= GD Start_Timer(RRP_MediaLinkedT, R-port) MediaLinked_Frame.Req(R-port) Generate_Alarm(EVENT_THIS_STATE_CHG)	GD
49	LNM	Family_Frame.Cnf (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID != Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_SNT_FAMILY_RES)==FALSE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR)	LNM
50	LNM	Family_Frame.Cnf (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID != Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_CFM_FAMILY)==TRUE => Put_Event(R-port, PHY_LINK_DOWN)	LNM

#	Current	Event / Condition =>actions	Next state
51	LNM	Family_Frame.Cnf (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID == Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_SNT_FAMILY_RES)==TRUE && Chk_Port_Info(Toggle_Port(R-port), PORT_CFM_FAMILY)==TRUE => Stop_Timer(RRP_FamilyReqT, R-port) Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Set_Port_Info(R-port, PORT_CFM_FAMILY) Set_Port_Info(R-port, PORT_CFM_COMPLETE) DEV_STATE:= GD RRP_NET_TPG:= NET_TPG_RING Clear_AllUid( ) Start_Timer(RRP_NetIsRingCheckT, R-Port) RRP_Net_Is_Ring_Frame.Req(Toggle_Port(R-port)) Generate_Alarm(EVENT_THIS_STATE_CHG)	GD
52	LNM	Family_Frame.Cnf (R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv_DEV_UID == Get_LNM_UID(Toggle_Port(R-port)) && Chk_Port_Info(R-port, PORT_SNT_FAMILY_RES)==FALSE && Chk_Port_Info(Toggle_Port(R-port), PORT_CFM_FAMILY)==TRUE => Set_Neighbor_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR)	LNM
53	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_CFM_FAMILY) == FALSE => Ignore	LNM
54	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_RCV_ADV) == TRUE => Set_Port_Info(R-port, PORT_CFM_COMPLETE) Set_Block_Port(INVALID_R_PORT) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	LNM

#	Current	Event / Condition =>actions	Next state
55	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == Get_Neighbor_UID(R-port) && Chk_Port_Info(R-port, PORT_RCV_ADV) == FALSE => Set_Port_Info(R-port, PORT_RCV_ML) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	LNM
56	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != Get_Neighbor_UID(R-port) => Set_Port_Info(R-port, PORT_RCV_ML) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port)	LNM
57	LNM	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == DEV_UID => DEV_STATE:= GD RRP_NET_TPG:= NET_TPG_RING Clear_AllUid( ) Start_Timer(RRP_ChangeRingStateT) Generate_Alarm(EVENT_THIS_STATE_CHG)	GD
58	LNM	Timer(RRP_MediaLinkedT, R-port) expired / => Start_Timer(RRP_MediaLinkedT) MediaLinked_Frame.Req(R-port)	LNM
59	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / Chk_Port_Info(R-port, PORT_CFM_FAMILY) == FALSE => Ignore	LNM
60	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_RCV_ML) == TRUE && rcv-DEV_UID == Get_Neighbor_UID(R-port) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_CFM_COMPLETE) Set_Block_Port(INVALID_R_PORT) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Ind)	LNM

#	Current	Event / Condition =>actions	Next state
61	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && Chk_Port_Info(R-port, PORT_RCV_ML) == FALSE && rcv-DEV_UID == Get_Neighbor_UID(R-port) => Stop_Timer(RRP_MediaLinkedT) Set_Port_Info(R-port, PORT_RCV_ADV) STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Ind)	LNM
62	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && rcv-DEV_UID != Get_Neighbor_UID(R-port) => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, AdvThis_Frame.Ind)	LNM
63	LNM	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / Chk_Port_Info(R-port, PORT_CFM_FAMILY) == TRUE && rcv-DEV_UID == DEV_UID => DEV_STATE:= GD RRP_NET_TPG:= NET_TPG_RING Clear_AllUid( ) Start_Timer(RRP_ChangeRingStateT) Generate_Alarm(EVENT_THIS_STATE_CHG)	GD
64	LNM	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && Chk_Port_Info(R-port, PORT_FAMILY_CFM) == FALSE && R-port == R-port1 => Clear_Port_Info(R-port) DELETE_PATH_INFO(R-port)	LNM

#	Current	Event / Condition =>actions	Next state
65	LNM	Phy_Link_Change.Ind(R-port, Link_status) /Link_status == PHY_LINK_DOWN && Chk_Port_Info(R-port, PORT_FAMILY_CFM) == TRUE => INIT_DEV_INFO( ) INIT_NET_INFO( ) INIT_PATH_INFO( ) Set_Port_Info(R-port, PORT_LINK_DOWN) Set_Block_Port(INVALID_R_PORT) Clear_Port_Info(R-port1) Clear_Port_Info(R-port2) DEV_STATE:= SA STORE_PATH_INFO(DEV_ADDR) Generate_Alarm(EVENT_THIS_STATE_CHG) Generate_Alarm(EVENT_NET_TPG_CHG)	SA
66	LNM	RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / rcv_DEV_UID != Get_Neighbor_UID(Toggle_Port(R-port)) => Store_LNM_UID(R-port, rcv-DEV_UID) STORE_PATH_INFO(rcv-DEV_ADDR) Build_Line_Path( )	LNM
67	LNM	RRP_Line_Start_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) / rcv_DEV_UID == Get_Neighbor_UID(Toggle_Port(R-port)) => Put_Event(Toggle_Port(R-port), PHY_LINK_DOWN)	LNM
68	GD	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == DEV_UID => RRP_NET_TPG:= NET_TPG_RING Clear_AllUid( ) Stop_Timer(RRP_ChangeRingStateT) Start_Timer(RRP_ChangeRingStateT)	GD
69	GD	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != DEV_UID && RRP_NET_TPG == NET_TPG_LINE && (Get_LNM_UID(R-port1) != INVALID_UID) && (Get_LNM_UID(R-port2) != INVALID_UID) => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port) FW_CTRL(R-port1,ENABLE) FW_CTRL(R-port2,ENABLE)	GD

#	Current	Event / Condition =>actions	Next state
70	GD	MediaLinked_Frame.Ind(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != DEV_UID && RRP_NET_TPG == NET_TPG_LINE && (Get_LNM_UID(R-port1) == INVALID_UID || Get_LNM_UID(R-port2) == INVALID_UID) => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port) FW_CTRL(R-port1,DISABLE) FW_CTRL(R-port2,DISABLE)	GD
71	GD	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID == DEV_UID => Clear_AllUid( ) RRP_NET_TPG:= NET_TPG_RING Stop_Timer(RRP_ChangeRingStateT) Start_Timer(RRP_ChangeRingStateT)	GD
72	GD	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != DEV_UID && RRP_NET_TPG == NET_TPG_LINE && (Get_LNM_UID(R-port1) != INVALID_UID) && (Get_LNM_UID(R-port2) != INVALID_UID) => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port) FW_CTRL(R-port1,ENABLE) FW_CTRL(R-port2,ENABLE)	GD
73	GD	AdvThis_Frame.Cnf(R-port, rcv-DEV_UID, rcv-DEV_ADDR) /rcv-DEV_UID != DEV_UID && RRP_NET_TPG == NET_TPG_LINE && (Get_LNM_UID(R-port1) == INVALID_UID || Get_LNM_UID(R-port2) == INVALID_UID) => STORE_PATH_INFO(rcv-DEV_ADDR) Forward_Frame(R-port, MediaLinked_Frame.Ind) AdvThis_Frame.Res(R-port) FW_CTRL(R-port1,DISABLE) FW_CTRL(R-port2,DISABLE)	GD