

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Industrial communication networks – Fieldbus specifications –
Part 6-10: Application layer protocol specification – Type 10 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 6-10: Spécification du protocole de la couche application – Éléments
de type 10**



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2014 IEC, Geneva, Switzerland

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either IEC or IEC's member National Committee in the country of the requester. If you have any questions about IEC copyright or have an enquiry about obtaining additional rights to this publication, please contact the address below or your local IEC member National Committee for further information.

Droits de reproduction réservés. Sauf indication contraire, aucune partie de cette publication ne peut être reproduite ni utilisée sous quelque forme que ce soit et par aucun procédé, électronique ou mécanique, y compris la photocopie et les microfilms, sans l'accord écrit de l'IEC ou du Comité national de l'IEC du pays du demandeur. Si vous avez des questions sur le copyright de l'IEC ou si vous désirez obtenir des droits supplémentaires sur cette publication, utilisez les coordonnées ci-après ou contactez le Comité national de l'IEC de votre pays de résidence.

IEC Central Office
3, rue de Varembe
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
Fax: +41 22 919 03 00
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 corrigenda or an amendment might have been published.

IEC Catalogue - webstore.iec.ch/catalogue

The stand-alone application for consulting the entire bibliographical information on IEC International Standards, Technical Specifications, Technical Reports and other documents. Available for PC, Mac OS, Android Tablets and iPad.

IEC publications search - www.iec.ch/searchpub

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 also once a month by email.

Electropedia - www.electropedia.org

The world's leading online dictionary of electronic and electrical terms containing more than 30 000 terms and definitions in English and French, with equivalent terms in 14 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

More than 55 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

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: csc@iec.ch.

A propos de l'IEC

La Commission Electrotechnique Internationale (IEC) est la première organisation mondiale qui élabore et publie des Normes internationales pour tout ce qui a trait à l'électricité, à l'électronique et aux technologies apparentées.

A propos des publications IEC

Le contenu technique des publications IEC est constamment revu. Veuillez vous assurer que vous possédez l'édition la plus récente, un corrigendum ou amendement peut avoir été publié.

Catalogue IEC - webstore.iec.ch/catalogue

Application autonome pour consulter tous les renseignements bibliographiques sur les Normes internationales, Spécifications techniques, Rapports techniques et autres documents de l'IEC. Disponible pour PC, Mac OS, tablettes Android et iPad.

Recherche de publications IEC - www.iec.ch/searchpub

La recherche avancée permet de trouver des publications IEC en utilisant différents critères (numéro de référence, texte, comité d'études,...). Elle donne aussi des informations sur les projets et les publications remplacées ou retirées.

IEC Just Published - webstore.iec.ch/justpublished

Restez informé sur les nouvelles publications IEC. Just Published détaille les nouvelles publications parues. Disponible en ligne et aussi une fois par mois par email.

Electropedia - www.electropedia.org

Le premier dictionnaire en ligne de termes électroniques et électriques. Il contient plus de 30 000 termes et définitions en anglais et en français, ainsi que les termes équivalents dans 14 langues additionnelles. Egalement appelé Vocabulaire Electrotechnique International (IEV) en ligne.

Glossaire IEC - std.iec.ch/glossary

Plus de 55 000 entrées terminologiques électrotechniques, en anglais et en français, extraites des articles Termes et Définitions des publications IEC parues depuis 2002. Plus certaines entrées antérieures extraites des publications des CE 37, 77, 86 et CISPR de l'IEC.

Service Clients - webstore.iec.ch/csc

Si vous désirez nous donner des commentaires sur cette publication ou si vous avez des questions contactez-nous: csc@iec.ch.

INTERNATIONAL STANDARD

NORME INTERNATIONALE



**Industrial communication networks – Fieldbus specifications –
Part 6-10: Application layer protocol specification – Type 10 elements**

**Réseaux de communication industriels – Spécifications des bus de terrain –
Partie 6-10: Spécification du protocole de la couche application – Eléments
de type 10**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

COMMISSION
ELECTROTECHNIQUE
INTERNATIONALE

PRICE CODE
CODE PRIX

XH

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-1761-0

**Warning! Make sure that you obtained this publication from an authorized distributor.
Attention! Veuillez vous assurer que vous avez obtenu cette publication via un distributeur agréé.**

CONTENTS

FOREWORD.....	28
INTRODUCTION.....	31
1 Scope.....	33
1.1 General.....	33
1.2 Specifications.....	33
1.3 Conformance.....	34
2 Normative references	34
3 Terms, definitions, abbreviations, symbols and conventions	36
3.1 Referenced terms and definitions	36
3.2 Additional terms and definitions for decentralized periphery.....	37
3.3 Abbreviations and symbols.....	45
3.4 Conventions	48
4 Application layer protocol specification for common protocols.....	56
4.1 FAL syntax description	56
4.2 Transfer syntax	59
4.3 Discovery and basic configuration	71
4.4 Precision working time control.....	110
4.5 Time synchronization	185
4.6 Media redundancy.....	185
4.7 Real time cyclic.....	185
4.8 Real time acyclic.....	213
4.9 Fragmentation.....	231
4.10 Remote procedure call	247
4.11 Link layer discovery.....	265
4.12 MAC Bridges.....	276
4.13 Virtual bridges.....	305
4.14 IP suite.....	314
4.15 Domain name system.....	317
4.16 Dynamic host configuration	318
4.17 Simple network management.....	319
4.18 Common DLL Mapping Protocol Machines.....	320
5 Application layer protocol specification for decentralized periphery.....	325
5.1 FAL syntax description	325
5.2 Transfer syntax	343
5.3 FAL protocol state machines	522
5.4 AP-Context state machine	524
5.5 FAL Service Protocol Machines.....	524
5.6 Application Relationship Protocol Machines	544
5.7 DLL Mapping Protocol Machines	663
Annex A (normative) Unified establishing of an AR for all RT classes	664
Annex B (normative) Compatible establishing of an AR.....	670
Annex C (informative) Establishing of a device access AR.....	673
Annex D (informative) Establishing of an AR (accelerated procedure).....	674
Annex E (informative) Establishing of an AR (fast startup procedure).....	677
Annex F (informative) Example of the upload, storage and retrieval procedure	679

Annex G (informative) OSI reference model layers.....	682
Annex H (informative) Overview of the IO controller and the IO device state machines	683
Annex I (informative) Priority regeneration	685
Annex J (informative) Overview of the PTCP synchronization master hierarchy	687
Annex K (informative) Optimization of bandwidth usage	689
Annex L (informative) Time constraints for bandwidth allocation	691
Annex M (informative) Time constraints for the forwarding of a frame	693
Annex N (informative) Principle of dynamic frame packing	694
Annex O (informative) Principle of Fragmentation	697
Annex P (informative) MRPD – Principle of seamless media redundancy.....	700
Annex Q (normative) Principle of a RED_RELAY without forwarding information in PDIRFrameData	702
Annex R (informative) Optimization for fast startup without autonegotiation	704
Annex S (informative) TX-error handling	706
Annex T (informative) Example of a PrmBegin, PrmEnd and ApplRdy sequence	707
Annex U (informative) List of supported MIBs	708
Annex V (informative) Structure and content of BLOB	709
Annex W (normative) LLDP EXT MIB	710
Bibliography.....	727
Figure 1 – Common structure of specific fields	49
Figure 2 – Common structure of specific fields for octet 1 (high)	50
Figure 3 – Common structure of specific fields for octet 2 (low)	50
Figure 4 – Common structure of specific fields for octet 1 (high)	51
Figure 5 – Common structure of specific fields for octet 2	51
Figure 6 – Common structure of specific fields for octet 3	52
Figure 7 – Common structure of specific fields for octet 4 (low)	52
Figure 8 – Coding of the data type BinaryDate	61
Figure 9 – Encoding of TimeOfDay with date indication value	61
Figure 10 – Encoding of TimeOfDay without date indication value	62
Figure 11 – Encoding of TimeDifference with date indication value	62
Figure 12 – Encoding of TimeDifference without date indication value	62
Figure 13 – FastForwardingMulticastMACAdd.....	66
Figure 14 – State transition diagram of DCPUCS	95
Figure 15 – State transition diagram of DCPUCR.....	99
Figure 16 – State transition diagram of DCPMCS.....	102
Figure 17 – State transition diagram of DCPMCR	105
Figure 18 – State transition diagram of DCPHMCS	108
Figure 19 – State transition diagram of DCPHMCR	109
Figure 20 – PTCP_SequenceID value range	115
Figure 21 – Timescale correspondence between PTCP_Time and CycleCounter	118
Figure 22 – Message timestamp point.....	123

Figure 23 – Timer model	123
Figure 24 – Four message timestamps	124
Figure 25 – Line delay protocol with follow up	125
Figure 26 – Line delay protocol without follow up	125
Figure 27 – Line delay measurement	127
Figure 28 – Model parameter for GSDML usage	129
Figure 29 – Bridge delay measurement	130
Figure 30 – Delay accumulation	131
Figure 31 – Worst case accumulated time deviation of synchronization	132
Figure 32 – Scheme for measurement of deviation	132
Figure 33 – Measurement of deviation	132
Figure 34 – PTCP master sending Sync-Frame without Follow Up-Frame	133
Figure 35 – PTCP master sending Sync-Frame with FollowUp-Frame	134
Figure 36 – !FU Sync Slave Forwarding Sync-Frame	135
Figure 37 – FU Sync Slave Forwarding Sync- and FollowUp-Frame	136
Figure 38 – FU Sync Slave Forwarding Sync- and Generating FollowUp-Frame	137
Figure 39 – Principle of the monitoring of the line delay measurement	138
Figure 40 – State transition diagram of DELAY_REQ	140
Figure 41 – State transition diagram of DELAY_RSP	147
Figure 42 – Overview of PTCP	151
Figure 43 – State transition diagram of SYN_BMA	153
Figure 44 – State transition diagram of SYN_MPSM	163
Figure 45 – State transition diagram of SYN_SPSM	169
Figure 46 – State transition diagram of SYNC_RELAY	175
Figure 47 – State transition diagram of SCHEDULER	182
Figure 48 – CycleCounter value range	187
Figure 49 – Structure of the CycleCounter	188
Figure 50 – Optimized CycleCounter setting	189
Figure 51 – SFCRC16 generation rule	192
Figure 52 – SFCycleCounter value range	194
Figure 53 – Basic structure of a PPM with frame structure	196
Figure 54 – Basic structure of a PPM with subframe structure	197
Figure 55 – State transition diagram of PPM	199
Figure 56 – Basic structure of a CPM	203
Figure 57 – State transition diagram of CPM	205
Figure 58 – Addressing scheme of RTA	215
Figure 59 – Structure of the APM	218
Figure 60 – Structure of the APMS	219
Figure 61 – State transition diagram of APMS	221
Figure 62 – Structure of the APMR	226
Figure 63 – State transition diagram of APMR	228
Figure 64 – State transition diagram of FRAG_D	238
Figure 65 – State transition diagram of FRAG_S	241

Figure 66 – State transition diagram of DEFrag	244
Figure 67 – State transition diagram of RTC3PSM	280
Figure 68 – State transition diagram for generating events	284
Figure 69 – State transition diagram of RED_RELAY	285
Figure 70 – Scheme of the DFP_RELAY	289
Figure 71 – Scheme of the DFP_INBOUND and DFP_STORAGE	290
Figure 72 – Scheme of the DFP_OUTBOUND	290
Figure 73 – State transition diagram of DFP_RELAY	291
Figure 74 – State transition diagram of DFP_RELAY_INBOUND	294
Figure 75 – State transition diagram of DFP_RELAY_IN_STORAGE	298
Figure 76 – State transition diagram of DFP_RELAY_OUTBOUND	302
Figure 77 – State transition diagram of MUX	306
Figure 78 – State transition diagram of DEMUX	311
Figure 79 – Structuring of the protocol machines within the DMPPM (bridge)	321
Figure 80 – State transition diagram of LMPM	323
Figure 81 – AlarmSpecifier.SequenceNumber value range	357
Figure 82 – FrameSendOffset vs. duration of a cycle	393
Figure 83 – Severity classification of diagnosis, maintenance and qualified	436
Figure 84 – Calculation principle for a cycle	455
Figure 85 – Calculation principle for the minimum YellowTime	456
Figure 86 – Definition of the reserved interval	462
Figure 87 – Toplevel view to the PLL window	466
Figure 88 – Definition of PLL window	466
Figure 89 – Toplevel view to the time PLL window	469
Figure 90 – Definition of time PLL window	470
Figure 91 – Detection of dropped frames — appear	478
Figure 92 – Detection of dropped frames — disappear	478
Figure 93 – Detection of DFP late error — appear and disappear	485
Figure 94 – MediaRedundancyWatchDog expired — appear and disappear	486
Figure 95 – Relationship among Protocol Machines	523
Figure 96 – State transition diagram of ALPMI	546
Figure 97 – State transition diagram of ALPMR	550
Figure 98 – Scheme of the IO device CM	553
Figure 99 – State transition diagram of the IO device CM	555
Figure 100 – State transition diagram of CMDEV	559
Figure 101 – Scheme of the IO device CM – device access	563
Figure 102 – State transition diagram of CMDEV_DA	565
Figure 103 – State transition diagram of CMSU	569
Figure 104 – State transition diagram of CMIO	574
Figure 105 – State transition diagram of CMWRR	578
Figure 106 – State transition diagram of CMRDR	582
Figure 107 – State transition diagram of CMSM	584
Figure 108 – State transition diagram of CMPBE	588

Figure 109 – State transition diagram of CMDMC	593
Figure 110 – State transition diagram of CMINA	597
Figure 111 – State transition diagram of CMRPC	606
Figure 112 – Scheme of the IO controller CM	612
Figure 113 – State transition diagram of the IO controller CM	613
Figure 114 – State transition diagram of CMCTL	617
Figure 115 – State transition diagram of CTLISM	623
Figure 116 – State transition diagram of CTLIO	626
Figure 117 – State transition diagram of CTRLDI	629
Figure 118 – State transition diagram of CTRLDR	632
Figure 119 – State transition diagram of CTRLPC	636
Figure 120 – State transition diagram of CTLSU	641
Figure 121 – State transition diagram of CTLWRI	646
Figure 122 – State transition diagram of CTLWRR	650
Figure 123 – State transition diagram of CTLPBE	653
Figure 124 – State transition diagram of CTLDINA	658
Figure 125 – Automatic NameOfStation assignment	663
Figure A.1 – Establishing of an AR using RT_CLASS_1, RT_CLASS_2 or RT_CLASS_3 (Initial connection monitoring w/o RT)	665
Figure A.2 – Establishing of an AR using RT_CLASS_1, RT_CLASS_2 or RT_CLASS_3 (Connection monitoring with RT)	666
Figure A.3 – Principle of the data evaluation during startup (RED channel establishment delayed)	667
Figure A.4 – Principle of the data evaluation during startup (RED channel establishment early)	668
Figure A.5 – Principle of the data evaluation during startup (Special case: Isochronous mode application)	669
Figure B.1 – Establishing of an AR using RT_CLASS_3 AR with startup mode “Legacy”	671
Figure B.2 – Establishing of an AR using RT_CLASS_1, 2 or UDP AR with startup mode “Legacy”	672
Figure C.1 – Establishing of a device access AR	673
Figure D.1 – Accelerated establishing of an IOAR without error	675
Figure D.2 – Accelerated establishing of an IOAR with “late” error	676
Figure E.1 – Establishing of an IOAR using fast startup	678
Figure F.1 – Example of upload with storage	680
Figure F.2 – Example of retrieval with storage	681
Figure G.1 – Assignment of the OSI reference model layers	682
Figure H.1 – Overview of the IO controller state machines	683
Figure H.2 – Overview of the IO device state machines	683
Figure H.3 – Overview of the common state machines	684
Figure J.1 – Level model for synchronization master hierarchy	687
Figure J.2 – Two level variant of the synchronization master hierarchy	688
Figure K.1 – Devices build up in a linear structure	689
Figure K.2 – Propagation of frames in linear transmit direction	689
Figure K.3 – Propagation of a frames in receive direction	690

Figure L.1 – Overview of time constraints for bandwidth allocation.....	691
Figure L.2 – Calculation of the length of a RED period.....	691
Figure L.3 – Calculation of the length of a GREEN period.....	692
Figure M.1 – Minimization of bridge delay.....	693
Figure N.1 – Dynamic frame packing.....	694
Figure N.2 – Dynamic frame packing – truncation of outputs.....	695
Figure N.3 – Dynamic frame packing – concatenation of inputs.....	695
Figure N.4 – End node mode.....	696
Figure N.5 – DFPFeed definition.....	696
Figure O.1 – Principle of fragmentation.....	697
Figure O.2 – Protocol elements of fragments.....	697
Figure O.3 – Bandwidth allocation using fragmentation.....	698
Figure O.4 – Guardian for a fragmentation domain.....	698
Figure P.1 – Principle of seamless media redundancy – I/OCR.....	700
Figure P.2 – Principle of seamless media redundancy – MCR.....	701
Figure P.3 – Principle of seamless media redundancy – Line.....	701
Figure Q.1 – Generating the FrameSendOffset for a RED_RELAY without forwarding information in PDIRFrameData.....	702
Figure R.1 – Scheme of a 2-port switch.....	704
Figure R.2 – Scheme of 2-ports.....	704
Figure T.1 – PrmBegin, PrmEnd and ApplRdy procedure.....	707
Table 1 – State machine description elements.....	53
Table 2 – Description of state machine elements.....	53
Table 3 – Conventions used in state machines.....	53
Table 4 – Conventions for services used in state machines.....	54
Table 5 – IEEE 802.3 DLPDU syntax.....	56
Table 6 – IEEE 802.11 DLPDU syntax.....	57
Table 7 – IEEE 802.15.1 DLPDU syntax.....	58
Table 8 – SourceAddress.....	63
Table 9 – DCP_MulticastMACAdd for Identify.....	64
Table 10 – DCP_MulticastMACAdd for Hello.....	64
Table 11 – DCP_MulticastMACAdd.....	64
Table 12 – PTCP_MulticastMACAdd range 1.....	64
Table 13 – PTCP_MulticastMACAdd range 2.....	64
Table 14 – PTCP_MulticastMACAdd range 3.....	65
Table 15 – PTCP_MulticastMACAdd range 4.....	65
Table 16 – PTCP_MulticastMACAdd range 5.....	65
Table 17 – PTCP_MulticastMACAdd range 6.....	65
Table 18 – PTCP_MulticastMACAdd range 7.....	65
Table 19 – PTCP_MulticastMACAdd range 8.....	66
Table 20 – RT_CLASS_3 destination multicast address.....	67
Table 21 – RT_CLASS_3 invalid frame multicast address.....	67

Table 22 – LT (Length/Type).....	67
Table 23 – TagControlInformation.Priority.....	68
Table 24 – FrameID range 1	68
Table 25 – FrameID range 2	68
Table 26 – FrameID range 3	69
Table 27 – FrameID range 4	69
Table 28 – FrameID range 5	69
Table 29 – FrameID range 6	69
Table 30 – FrameID range 7	69
Table 31 – FrameID range 8	70
Table 32 – FrameID range 9	70
Table 33 – FrameID range 10	70
Table 34 – FrameID range 11	71
Table 35 – FrameID range 12	71
Table 36 – FrameID range 13	71
Table 37 – FragmentationFrameID.FragSequence	71
Table 38 – FragmentationFrameID.Constant.....	71
Table 39 – DCP APDU syntax.....	72
Table 40 – DCP substitutions.....	72
Table 41 – ServiceID	75
Table 42 – ServiceType.Selection.....	75
Table 43 – ServiceType.Reserved	76
Table 44 – ServiceType.Selection.....	76
Table 45 – ServiceType.Reserved_1.....	76
Table 46 – ServiceType.Response.....	76
Table 47 – ServiceType.Reserved_2.....	77
Table 48 – ResponseDelayFactor.....	77
Table 49 – List of options.....	78
Table 50 – List of suboptions for option IPOption	79
Table 51 – List of suboptions for option DevicePropertiesOption.....	79
Table 52 – List of suboptions for option DHCPOption.....	79
Table 53 – List of suboptions for option ControlOption	79
Table 54 – List of suboptions for option DeviceInitiativeOption.....	80
Table 55 – List of suboptions for option AllSelectorOption	80
Table 56 – List of suboptions for option ManufacturerSpecificOption.....	80
Table 57 – SuboptionDHCP	81
Table 58 – Coding of DCPBlockLength in conjunction with SuboptionStart.....	82
Table 59 – Coding of DCPBlockLength in conjunction with SuboptionStop	82
Table 60 – Coding of DCPBlockLength in conjunction with SuboptionSignal	83
Table 61 – Coding of DCPBlockLength in conjunction with SuboptionFactoryReset	83
Table 62 – Coding of DCPBlockLength in conjunction with SuboptionResetToFactory	84
Table 63 – Coding of DCPBlockLength in conjunction with SuboptionDeviceInitiative	84

Table 64 – BlockQualifier with options IOption, DevicePropertiesOption, DHCPOption and ManufacturerSpecificOption	85
Table 65 – BlockQualifier with option ControlOption and suboption SuboptionResetToFactory	85
Table 66 – BlockQualifier with all other options and suboptions	86
Table 67 – BlockError	86
Table 68 – BlockInfo for SuboptionIPParameter	87
Table 69 – Bit 1 and Bit 0 of BlockInfo for SuboptionIPParameter	87
Table 70 – Bit 7 of BlockInfo for SuboptionIPParameter	87
Table 71 – BlockInfo for all other suboptions	87
Table 72 – DeviceInitiativeValue	87
Table 73 – SignalValue	88
Table 74 – DeviceRoleDetails	90
Table 75 – IPAddress	90
Table 76 – Subnetmask	90
Table 77 – StandardGateway	92
Table 78 – Remote primitives issued or received by DCPUCS	94
Table 79 – Local primitives issued or received by DCPUCS	95
Table 80 – DCPUCS state table	96
Table 81 – Functions, Macros, Timers and Variables used by the DCPUCS	98
Table 82 – Remote primitives issued or received by DCPUCR	98
Table 83 – Local primitives issued or received by DCPUCR	99
Table 84 – DCPUCR state table	100
Table 85 – Functions, Macros, Timers and Variables used by the DCPUCR	101
Table 86 – Remote primitives issued or received by DCPMCS	102
Table 87 – Local primitives issued or received by DCPMCS	102
Table 88 – DCPMCS state table	103
Table 89 – Functions used by the DCPMCS	104
Table 90 – Remote primitives issued or received by DCPMCR	105
Table 91 – Local primitives issued or received by DCPMCR	105
Table 92 – DCPMCR state table	106
Table 93 – Functions, Macros, Timers and Variables used by the DCPMCR	106
Table 94 – Remote primitives issued or received by DCPHMCS	107
Table 95 – Local primitives issued or received by DCPHMCS	107
Table 96 – DCPHMCS state table	108
Table 97 – Functions, Macros, Timers and Variables used by the DCPHMCS	109
Table 98 – Remote primitives issued or received by DCPHMCR	109
Table 99 – Local primitives issued or received by DCPHMCR	109
Table 100 – DCPHMCR state table	110
Table 101 – Functions, Macros, Timers and Variables used by the DCPHMCR	110
Table 102 – PTCP APDU syntax	111
Table 103 – PTCP substitutions	111
Table 104 – PTCP_TLVHeader.Type	112
Table 105 – PTCP_Delay10ns	113

Table 106 – PTCP_Delay1ns_Byte.Value	113
Table 107 – PTCP_Delay1ns	113
Table 108 – PTCP_Delay1ns_FUP	114
Table 109 – PTCP_SequenceID	114
Table 110 – PTCP_SubType for OUI (=00-0E-CF)	115
Table 111 – PTCP_Seconds	116
Table 112 – PTCP_NanoSeconds	116
Table 113 – PTCP_Flags.LeapSecond	116
Table 114 – Timescale correspondence between MJD, UTC and PTCP_EPOCHNumber	117
Table 115 – Timescale correspondence between PTCP_EPOCHNumber, PTCP_Second, PTCP_Nanosecond, CycleCounter and SendClockFactor	117
Table 116 – PTCP_MasterPriority1.Priority for SyncID == 0 and SyncProperties.Role == 2	119
Table 117 – PTCP_MasterPriority1.Priority for SyncID == 0 and SyncProperties.Role == 1	119
Table 118 – PTCP_MasterPriority1.Level	119
Table 119 – PTCP_MasterPriority2	120
Table 120 – PTCP_ClockClass for SyncID == 0 (working clock synchronization)	120
Table 121 – PTCP_ClockAccuracy	121
Table 122 – PTCP_ClockVariance	121
Table 123 – PTCP_T2PortRxDelay	121
Table 124 – PTCP_T3PortTxDelay	122
Table 125 – PTCP_T2TimeStamp	122
Table 126 – Remote primitives issued or received by DELAY_REQ	139
Table 127 – Local primitives issued or received by DELAY_REQ	139
Table 128 – DELAY_REQ state table	141
Table 129 – Functions, macros, timers and variables used by the DELAY_REQ	144
Table 130 – Remote primitives issued or received by DELAY_RSP	146
Table 131 – Local primitives issued or received by DELAY_RSP	147
Table 132 – DELAY_RSP state table	148
Table 133 – Functions, Macros, Timers and Variables used by the DELAY_RSP	150
Table 134 – Remote primitives issued or received by SYN_BMA	151
Table 135 – Local primitives issued or received by SYN_BMA	152
Table 136 – SYN_BMA state table	154
Table 137 – Functions, Macros, Timers and Variables used by the SYN_BMA	159
Table 138 – Remote primitives issued or received by SYN_MPSM	161
Table 139 – Local primitives issued or received by SYN_MPSM	162
Table 140 – SYN_MPSM state table	164
Table 141 – Functions, Macros, Timers and Variables used by the SYN_MPSM	167
Table 142 – Remote primitives issued or received by SYN_SPSM	168
Table 143 – Local primitives issued or received by SYN_SPSM	168
Table 144 – SYN_SPSM state table	170
Table 145 – Functions, Macros, Timers and Variables used by the SYN_SPSM	173
Table 146 – Truth table for one SyncID for receiving sync and follow up frames	174

Table 147 – Remote primitives issued or received by SYNC_RELAY	175
Table 148 – Local primitives issued or received by SYNC_RELAY	175
Table 149 – SYNC_RELAY state table	176
Table 150 – Functions, Macros, Timers and Variables used by the SYNC_RELAY	178
Table 151 – Truth table for one SyncID for receiving	180
Table 152 – Truth table for one SyncID for transmitting	180
Table 153 – Remote primitives issued or received by SCHEDULER	181
Table 154 – Local primitives issued or received by SCHEDULER	181
Table 155 – SCHEDULER state table	182
Table 156 – Functions, Macros, Timers and Variables used by the SCHEDULER	184
Table 157 – Truth table for RxPeriodChecker of one port	184
Table 158 – Truth table for TxPeriodChecker of one port	184
Table 159 – RTC APDU syntax	185
Table 160 – RTC substitutions	185
Table 161 – CycleCounter Difference	187
Table 162 – DataStatus.State	189
Table 163 – DataStatus.Redundancy	190
Table 164 – DataStatus.DataValid	190
Table 165 – DataStatus.ProviderState	190
Table 166 – DataStatus.StationProblemIndicator	190
Table 167 – DataStatus.Ignore of a frame	191
Table 168 – DataStatus.Ignore of a sub frame	191
Table 169 – TransferStatus for RT_CLASS_3	191
Table 170 – SFPosition.Position	192
Table 171 – SFPosition.Reserved	193
Table 172 – SFDataLength	193
Table 173 – SFCycleCounter Difference	194
Table 174 – IOxS.Extension	195
Table 175 – IOxS.Instance	195
Table 176 – IOxS.DataState	195
Table 177 – Remote primitives issued or received by PPM	197
Table 178 – Local primitives issued or received by PPM	198
Table 179 – PPM state table	200
Table 180 – Functions, Macros, Timers and Variables used by the PPM	201
Table 181 – Truth table used by the PPM for TxOption	202
Table 182 – Remote primitives issued or received by CPM	203
Table 183 – Local primitives issued or received by CPM	204
Table 184 – CPM state table	206
Table 185 – Functions, Macros, Timers and Variables used by the CPM	209
Table 186 – Truth table used by the CPM for RxOption	210
Table 187 – Truth table for one frame using RT_CLASS_x	210
Table 188 – Truth table for one frame using RT_CLASS_UDP	211
Table 189 – Truth table for the C_SDU	211

Table 190 – Truth table for arranging Dht and data	212
Table 191 – Truth table for the subframe – frame check.....	212
Table 192 – Truth table for the subframe – sub frame check	212
Table 193 – Truth table for the subframe – sub frame data check	213
Table 194 – Truth table for the subframe – Dht and data	213
Table 195 – RTA APDU syntax	213
Table 196 – RTA substitutions	214
Table 197 – PDUType.Type	215
Table 198 – PDUType.Version	216
Table 199 – AddFlags.WindowSize	216
Table 200 – AddFlags.TACK	216
Table 201 – SendSeqNum	217
Table 202 – AckSeqNum	217
Table 203 – VarPartLen	217
Table 204 – Remote primitives issued or received by APMS	219
Table 205 – Local primitives issued or received by APMS	220
Table 206 – APMS state table	222
Table 207 – Functions, Macros, Timers and Variables used by the APMS	224
Table 208 – Remote primitives issued or received by APMR	226
Table 209 – Local primitives issued or received by APMR	227
Table 210 – APMR state table	229
Table 211 – Functions, Macros, Timers and Variables used by the APMR	231
Table 212 – TagControlInformation.Priority vs. streams	231
Table 213 – Lower limit of fragments	234
Table 214 – FRAG APDU syntax	235
Table 215 – FRAG substitutions	235
Table 216 – FragDataLength	236
Table 217 – FragStatus.FragmentNumber	236
Table 218 – FragStatus.Reserved	236
Table 219 – FragStatus.MoreFollows	237
Table 220 – Remote primitives issued or received by FRAG_D	237
Table 221 – Local primitives issued or received by FRAG_D	237
Table 222 – FRAG_D state table (dynamic)	238
Table 223 – Functions, Macros, Timers and Variables used by the FRAG_D (dynamic)	240
Table 224 – Remote primitives issued or received by FRAG_S	241
Table 225 – Local primitives issued or received by FRAG_S	241
Table 226 – FRAG_S state table (static)	242
Table 227 – Functions, Macros, Timers and Variables used by the FRAG_S (static)	243
Table 228 – Remote primitives issued or received by DEFRAG	244
Table 229 – Local primitives issued or received by DEFRAG	244
Table 230 – DEFRAG state table	245
Table 231 – Functions, Macros, Timers and Variables used by the DEFRAG	246
Table 232 – Truth table for the DefragGuard – first fragment	246

Table 233 – Truth table for the DefragGuard – next fragment.....	246
Table 234 – Truth table for the DefragGuard – last fragment.....	247
Table 235 – RPC APDU syntax.....	247
Table 236 – RPC substitutions.....	247
Table 237 – RPCVersion.....	248
Table 238 – RPCPacketType	249
Table 239 – RPCFlags.....	249
Table 240 – RPCFlags2.....	249
Table 241 – RPCDRep.Character- and IntegerEncoding	250
Table 242 – RPCDRep Octet 2 – Floating Point Representation	250
Table 243 – RPCObjectUUID.Data4.....	251
Table 244 – RPCObjectUUID for PNIO	251
Table 245 – RPCObjectUUID for PNIO with multiple interfaces	252
Table 246 – RPCInterfaceUUID for PNIO.....	252
Table 247 – RPCInterfaceUUID for the RPC end point mapper.....	253
Table 248 – RPCInterfaceVersion.Major	253
Table 249 – RPCInterfaceVersion.Minor	254
Table 250 – RPCOperationNmb (IO device, controller and supervisor)	254
Table 251 – RPCOperationNmb for endpoint mapper.....	254
Table 252 – RPCDataRepresentationUUID – defined values.....	257
Table 253 – RPCInquiryType	258
Table 254 – RPCEPMapStatus	260
Table 255 – Values of NCAFaultStatus	262
Table 256 – Values of NCAResjectStatus	263
Table 257 – Remote primitives issued or received by RPC	264
Table 258 – Local primitives issued or received by RPC	264
Table 259 – LLDP APDU syntax	265
Table 260 – LLDP substitutions	266
Table 261 – LLDP_ChassisID in conjunction with MultipleInterfaceMode.NameOfDevice and NameOfStation	267
Table 262 – LLDP_PortID in conjunction with MultipleInterfaceMode.NameOfDevice	267
Table 263 – LLDP_PNIO_SubType	268
Table 264 – PTCP_PortRxDelayLocal.....	268
Table 265 – PTCP_PortRxDelayRemote	268
Table 266 – PTCP_PortTxDelayLocal	268
Table 267 – PTCP_PortTxDelayRemote	269
Table 268 – CableDelayLocal	269
Table 269 – RTClass2_PortStatus.State with ARProperties.StartupMode == Legacy	269
Table 270 – RTClass2_PortStatus.State with ARProperties.StartupMode == Advanced	269
Table 271 – RTClass3_PortStatus.State	270
Table 272 – RTClass3_PortStatus.Fragmentation.....	270
Table 273 – RTClass3_PortStatus.PreambleLength.....	270
Table 274 – Truth table for shortening of the preamble	271

Table 275 – RTClass3_PortStatus.Optimized.....	271
Table 276 – MRRT_PortStatus.State	272
Table 277 – IRDataUUID	272
Table 278 – LLDP_RedOrangePeriodBegin.Offset	272
Table 279 – LLDP_RedOrangePeriodBegin.Valid.....	273
Table 280 – LLDP_OrangePeriodBegin.Offset	273
Table 281 – LLDP_OrangePeriodBegin.Valid with ARProperties.StartupMode == Legacy	273
Table 282 – LLDP_OrangePeriodBegin.Valid with ARProperties.StartupMode == Advanced	273
Table 283 – LLDP_GreenPeriodBegin.Offset	274
Table 284 – LLDP_GreenPeriodBegin.Valid	274
Table 285 – LLDP_LengthOfPeriod.Length	274
Table 286 – LLDP_LengthOfPeriod.Valid	274
Table 287 – Unicast FDB entries	277
Table 288 – Multicast FDB entries	277
Table 289 – Broadcast FDB entry	278
Table 290 – Remote primitives issued or received by MAC_RELAY	279
Table 291 – Local primitives issued or received by MAC_RELAY	279
Table 292 – Functions, Macros, Timers and Variables used by the MAC_RELAY	279
Table 293 – Remote primitives issued or received by RTC3PSM	280
Table 294 – Local primitives issued or received by RTC3PSM	280
Table 295 – RTC3PSM state table	281
Table 296 – Functions, Macros, Timers and Variables used by the RTC3PSM	281
Table 297 – Truth table for the RTC3PSM	283
Table 298 – RXBeginEndAssignment and TXBeginEndAssignment.....	283
Table 299 – Event function table.....	284
Table 300 – Remote primitives issued or received by RED_RELAY	285
Table 301 – Local primitives issued or received by RED_RELAY	285
Table 302 – RED_RELAY state table	286
Table 303 – Functions, Macros, Timers and Variables used by the RED_RELAY	287
Table 304 – Truth table for the RedGuard with full check	288
Table 305 – Truth table for the RedGuard with reduced check	288
Table 306 – Truth table for the RedGuard with minimal check.....	288
Table 307 – Remote primitives issued or received by DFP_RELAY	291
Table 308 – Local primitives issued or received by DFP_RELAY	291
Table 309 – DFP_RELAY state table	292
Table 310 – Functions, Macros, Timers and Variables used by the DFP_RELAY	293
Table 311 – Truth table for the DFPGuard	293
Table 312 – Remote primitives issued or received by DFP_RELAY_INBOUND	293
Table 313 – Local primitives issued or received by DFP_RELAY_INBOUND	294
Table 314 – DFP_RELAY_INBOUND state table.....	295
Table 315 – Functions, Macros, Timers and Variables used by the DFP_RELAY_INBOUND	295

Table 316 – Truth table for the InboundGuard – frame check	296
Table 317 – Truth table for the InboundGuard – sub frame check	296
Table 318 – Truth table for the InboundGuard – sub frame data check.....	296
Table 319 – Truth table for the InboundGuard – full check	296
Table 320 – Remote primitives issued or received by DFP_RELAY_IN_STORAGE	297
Table 321 – Local primitives issued or received by DFP_RELAY_IN_STORAGE.....	297
Table 322 – DFP_RELAY_IN_STORAGE state table	298
Table 323 – Functions, Macros, Timers and Variables used by the DFP_RELAY_IN_STORAGE	300
Table 324 – Remote primitives issued or received by DFP_RELAY_OUTBOUND	301
Table 325 – Local primitives issued or received by DFP_RELAY_OUTBOUND	301
Table 326 – DFP_RELAY_OUTBOUND state table	303
Table 327 – Functions, Macros, Timers and Variables used by the DFP_RELAY_OUTBOUND.....	304
Table 328 – Truth table for the OutboundGuard – frame check	304
Table 329 – Truth table for the OutboundGuard – sub frame check	304
Table 330 – Remote primitives issued or received by MUX.....	305
Table 331 – Local primitives issued or received by MUX.....	306
Table 332 – MUX state table.....	307
Table 333 – Functions, Macros, Timers and Variables used by MUX.....	308
Table 334 – Truth table for FrameSizeFits	309
Table 335 – Truth table for StateChecker.....	309
Table 336 – Remote primitives issued or received by DEMUX	310
Table 337 – Local primitives issued or received by DEMUX	310
Table 338 – DEMUX state table	312
Table 339 – Functions, Macros, Timers and Variables used by the DEMUX	313
Table 340 – IP/UDP APDU syntax	314
Table 341 – IP/UDP substitutions	314
Table 342 – UDP_SrcPort.....	315
Table 343 – UDP_DstPort.....	315
Table 344 – IP_DstIPAddress	316
Table 345 – IP Multicast DstIPAddress according to RFC 2365	316
Table 346 – Remote primitives issued or received by DNS	317
Table 347 – Local primitives issued or received by DNS	318
Table 348 – Functions, Macros, Timers and Variables used by the DNS	318
Table 349 – Remote primitives issued or received by DHCP	318
Table 350 – Local primitives issued or received by machines.....	319
Table 351 – Functions, Macros, Timers and Variables used by the DHCP.....	319
Table 352 – Enterprise number	319
Table 353 – Remote primitives issued or received by LMPM	321
Table 354 – Local primitives issued or received by LMPM	323
Table 355 – LMPM state table	324
Table 356 – Functions, Macros, Timers and Variables used by the LMPM	325
Table 357 – IO APDU substitutions	326

Table 358 – BlockType	343
Table 359 – BlockVersionHigh	354
Table 360 – BlockVersionLow	355
Table 361 – AlarmType	355
Table 362 – AlarmSpecifier.SequenceNumber	356
Table 363 – AlarmSpecifier.SequenceNumber Difference	357
Table 364 – AlarmSpecifier.ChannelDiagnosis	358
Table 365 – AlarmSpecifier.ManufacturerSpecificDiagnosis	358
Table 366 – AlarmSpecifier.SubmoduleDiagnosisState	358
Table 367 – AlarmSpecifier.ARDiagnosticsState	359
Table 368 – API	359
Table 369 – SlotNumber	359
Table 370 – SubslotNumber	360
Table 371 – Expression 1 (subslot specific)	361
Table 372 – Expression 2 (slot specific)	361
Table 373 – Expression 3 (AR specific)	361
Table 374 – Expression 4 (API specific)	361
Table 375 – Expression 5 (device specific)	362
Table 376 – Grouping of DiagnosisData	362
Table 377 – Index (user specific)	362
Table 378 – Index (subslot specific)	363
Table 379 – Index (slot specific)	364
Table 380 – Index (AR specific)	365
Table 381 – Index (API specific)	366
Table 382 – Index (device specific)	367
Table 383 – ARType	368
Table 384 – IOCRMulticastMACAdd using RT_CLASS_UDP	369
Table 385 – IOCRMulticastMACAdd using RT_CLASS_2 or RT_CLASS_3	369
Table 386 – Type 10 OUI	369
Table 387 – ARProperties.State	370
Table 388 – ARProperties.SupervisorTakeoverAllowed	370
Table 389 – ARProperties.ParameterizationServer	370
Table 390 – ARProperties.DeviceAccess	371
Table 391 – ARProperties.CompanionAR	371
Table 392 – ARProperties.AcknowledgeCompanionAR	371
Table 393 – ARProperties.StartupMode	371
Table 394 – ARProperties.PullModuleAlarmAllowed	372
Table 395 – IOCRProperties.RTClass	372
Table 396 – IOCRTagHeader.IOCRVLANID	373
Table 397 – IOCRTagHeader.IOUserPriority	373
Table 398 – IOCRType	373
Table 399 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=0	374

Table 400 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=1 or ARProperties.StartupMode:=1	374
Table 401 – CMInitiatorTriggerTimeoutFactor	374
Table 402 – LengthIOCS	375
Table 403 – LengthIOPS	376
Table 404 – AlarmCRProperties.Priority	376
Table 405 – AlarmCRProperties.Transport	376
Table 406 – AlarmCRTagHeaderHigh.AlarmCRVLANID	377
Table 407 – AlarmCRTagHeaderHigh.AlarmUserPriority	377
Table 408 – AlarmCRTagHeaderLow.AlarmCRVLANID	377
Table 409 – AlarmCRTagHeaderLow.AlarmUserPriority	378
Table 410 – AlarmSequenceNumber	378
Table 411 – AlarmCRType	378
Table 412 – RTATimeoutFactor	379
Table 413 – RTARetries	379
Table 414 – AddressResolutionProperties.Protocol	380
Table 415 – AddressResolutionProperties.Factor	380
Table 416 – MCITimeoutFactor	380
Table 417 – VendorIDLow	381
Table 418 – VendorIDHigh	382
Table 419 – ModuleIdentNumber	382
Table 420 – SubmoduleIdentNumber	382
Table 421 – ARUUID	383
Table 422 – ARUUID in conjunction with ARType:= IOCARSR	384
Table 423 – TargetARUUID	384
Table 424 – ActualLocalTimeStamp	384
Table 425 – LocalTimeStamp	385
Table 426 – NumberOfLogEntries	385
Table 427 – EntryDetail	385
Table 428 – AdditionalValue1 and AdditionalValue2	385
Table 429 – ControlBlockProperties in conjunction with ControlCommand.ApplicationReady with ARProperties.StartupMode:=1	386
Table 430 – ControlBlockProperties in conjunction with ControlCommand.ApplicationReady with ARProperties.StartupMode:=0	386
Table 431 – ControlBlockProperties in conjunction with the other values of the field ControlCommand	386
Table 432 – ControlCommand.PrmEnd	386
Table 433 – ControlCommand.ApplicationReady	386
Table 434 – ControlCommand.Release	387
Table 435 – ControlCommand.Done	387
Table 436 – ControlCommand.ReadyForCompanion	387
Table 437 – ControlCommand.ReadyForRT_CLASS_3	387
Table 438 – ControlCommand.PrmBegin	387
Table 439 – DataDescription.Type	388

Table 440 – Values of DataLength	388
Table 441 – Values of SendClockFactor	389
Table 442 – Values of ReductionRatio for RT_CLASS_1 and RT_CLASS_2	390
Table 443 – Values of ReductionRatio for RT_CLASS_3	390
Table 444 – Values of ReductionRatio for RT_CLASS_UDP	390
Table 445 – Values of Phase	391
Table 446 – Values of Sequence	391
Table 447 – DataHoldFactor of a frame	392
Table 448 – DataHoldFactor of a Subframe	392
Table 449 – Values of FrameSendOffset	392
Table 450 – ModuleState	394
Table 451 – SubmoduleState.AddInfo	394
Table 452 – SubmoduleState.QualifiedInfo	394
Table 453 – SubmoduleState.MaintenanceRequired	395
Table 454 – SubmoduleState.MaintenanceDemanded	395
Table 455 – SubmoduleState.DiagInfo	395
Table 456 – SubmoduleState.ARInfo	395
Table 457 – SubmoduleState.IdentInfo	396
Table 458 – SubmoduleState.FormatIndicator	396
Table 459 – SubmoduleState.Detail	396
Table 460 – SubmoduleProperties.Type	397
Table 461 – SubmoduleProperties.SharedInput	397
Table 462 – SubmoduleProperties.ReduceInputSubmoduleDataLength	397
Table 463 – SubmoduleProperties.ReduceOutputSubmoduleDataLength	398
Table 464 – SubmoduleProperties.DiscardIOXS	398
Table 465 – SubstitutionMode	398
Table 466 – SubstituteActiveFlag	399
Table 467 – InitiatorUDPRTPort	399
Table 468 – ResponderUDPRTPort	399
Table 469 – InitiatorRPCServerPort	400
Table 470 – ResponderRPCServerPort	400
Table 471 – MaxAlarmDataLength	401
Table 472 – APStructureIdentifier with API := 0	401
Table 473 – APStructureIdentifier with API != 0	401
Table 474 – ExtendedIdentificationVersionHigh	402
Table 475 – ExtendedIdentificationVersionLow	402
Table 476 – Values of ErrorCode for negative responses	402
Table 477 – Values of ErrorDecode	403
Table 478 – Coding of ErrorCode1 with ErrorDecode PNIO	404
Table 479 – Coding of ErrorCode2 with ErrorDecode PNIO	405
Table 480 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 1)	405
Table 481 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 2 – alarm acknowledge)	408

Table 482 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 3 – machines)	409
Table 483 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 4 – IO controller)	410
Table 484 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 5 – IO device)	411
Table 485 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 6 – abort reasons)	412
Table 486 – Values of ErrorCode2 for ErrorCode1 = RPC	415
Table 487 – Coding of ErrorCode1 for ErrorDecode with the value ManufacturerSpecific	415
Table 488 – Coding of ErrorCode2 for ErrorDecode with the value ManufacturerSpecific	415
Table 489 – IM_Hardware_Revision	416
Table 490 – IM_SWRevision_Functional_Enhancement	416
Table 491 – IM_SWRevision_Bug_Fix	416
Table 492 – IM_SWRevision_Internal_Change	416
Table 493 – IM_Revision_Counter	417
Table 494 – IM_Profile_ID	417
Table 495 – IM_Profile_Specific_Type in conjunction with IM_Profile_ID := 0x0000	417
Table 496 – IM_Profile_Specific_Type in conjunction with IM_Profile_ID range 0x0001 – 0xF6FF	418
Table 497 – IM_Version_Major	418
Table 498 – IM_Version_Minor	418
Table 499 – IM_Supported_I&M1	418
Table 500 – IM_Date	420
Table 501 – UserStructureIdentifier	421
Table 502 – ChannelErrorType	422
Table 503 – ChannelNumber	423
Table 504 – ChannelProperties.Type	424
Table 505 – ChannelProperties.Maintenance	425
Table 506 – Valid combinations within ChannelProperties	425
Table 507 – Valid combinations for Alarmnotification and RecordDataRead(DiagnosisData)	426
Table 508 – ChannelProperties.Specifier	427
Table 509 – ChannelProperties.Direction	427
Table 510 – ExtChannelErrorType	427
Table 511 – Allowed combinations of ChannelErrorType, ExtChannelErrorType, and ExtChannelAddValue	428
Table 512 – ExtChannelErrorType for ChannelErrorType 0 – 0xFF	428
Table 513 – ExtChannelErrorType for ChannelErrorType 0x0100 – 0x7FFF	428
Table 514 – ExtChannelErrorType for ChannelErrorType “Data transmission impossible”	428
Table 515 – ExtChannelErrorType for ChannelErrorType “Remote mismatch”	429
Table 516 – ExtChannelErrorType for ChannelErrorType “Media redundancy mismatch”	429

Table 517 – ExtChannelErrorType for ChannelErrorType “Sync mismatch” and for ChannelErrorType “Time mismatch”	430
Table 518 – ExtChannelErrorType for ChannelErrorType “Isochronous mode mismatch”	430
Table 519 – ExtChannelErrorType for ChannelErrorType “Multicast CR mismatch”	430
Table 520 – ExtChannelErrorType for ChannelErrorType “Fiber optic mismatch”	430
Table 521 – ExtChannelErrorType for ChannelErrorType “Network component function mismatch”	431
Table 522 – ExtChannelErrorType for ChannelErrorType “Dynamic Frame Packing function mismatch”	431
Table 523 – ExtChannelErrorType for ChannelErrorType “Media redundancy with planned duplication mismatch”	432
Table 524 – ExtChannelErrorType for ChannelErrorType “System redundancy mismatch”	432
Table 525 – ExtChannelErrorType for ChannelErrorType “Multiple interface mismatch”	433
Table 526 – ExtChannelErrorType for ChannelErrorType “Nested diagnosis indication”	433
Table 527 – Values for ExtChannelAddValue	433
Table 528 – Values for Accumulative Info	434
Table 529 – Values for “Fiber optic mismatch” – “Power Budget”	434
Table 530 – Values for “Network component function mismatch” – “Frame dropped”	434
Table 531 – Values for “Remote mismatch” – “Peer CableDelay mismatch”	435
Table 532 – Values for “Multiple interface mismatch” – “Conflicting MultipleInterfaceMode.NameOfDevice mode”	435
Table 533 – Values for QualifiedChannelQualifier	435
Table 534 – Values for MaintenanceStatus	436
Table 535 – URRecordIndex	437
Table 536 – URRecordLength	437
Table 537 – MultipleInterfaceMode.NameOfDevice	438
Table 538 – LineDelay.Value with LineDelay.FormatIndicator == 0	439
Table 539 – LineDelay.Value with LineDelay.FormatIndicator == 1	439
Table 540 – LineDelay.FormatIndicator	439
Table 541 – RxPort	440
Table 542 – NumberOfTxPortGroups	440
Table 543 – TxPortEntry	441
Table 544 – FrameDetails.SyncFrame in conjunction with FrameDataProperties.ForwardingMode := “Absolute mode”	442
Table 545 – FrameDetails.SyncFrame in conjunction with FrameDataProperties.ForwardingMode := “Relative mode”	442
Table 546 – FrameDetails.MeaningFrameSendOffset	442
Table 547 – FrameDetails.MediaRedundancyWatchDog	443
Table 548 – FrameDataProperties.ForwardingMode	443
Table 549 – FrameDataProperties.FastForwardingMulticastMACAdd	443
Table 550 – FrameDataProperties.FragmentationMode	443
Table 551 – MAUType	444
Table 552 – Valid combinations between MAUType and LinkState	447
Table 553 – CheckSyncMode.CableDelay	447

Table 554 – CheckSyncMode.SyncMaster	448
Table 555 – MAUTypeMode.Check	448
Table 556 – DomainBoundaryIngress	448
Table 557 – DomainBoundaryEgress	449
Table 558 – DomainBoundaryAnnounce	449
Table 559 – MulticastBoundary	449
Table 560 – PeerToPeerBoundary	450
Table 561 – DCPBoundary	450
Table 562 – PreambleLength.Length	451
Table 563 – LinkState.Link	451
Table 564 – LinkState.Port	451
Table 565 – MediaType	452
Table 566 – MaxBridgeDelay	452
Table 567 – NumberOfPorts	452
Table 568 – MaxPortTxDelay	453
Table 569 – MaxPortRxDelay	453
Table 570 – MaxLineRxDelay	453
Table 571 – YellowTime	454
Table 572 – StartOfRedFrameID	456
Table 573 – EndOfRedFrameID	457
Table 574 – Dependencies of StartOfRedFrameID and EndOfRedFrameID	457
Table 575 – NumberOfAssignments	457
Table 576 – NumberOfPhases	458
Table 577 – AssignedValueForReservedBegin	458
Table 578 – AssignedValueForOrangeBegin	458
Table 579 – AssignedValueForReservedEnd	459
Table 580 – Values of RedOrangePeriodBegin	459
Table 581 – Dependencies of RedOrangePeriodBegin, OrangePeriodBegin and GreenPeriodBegin	459
Table 582 – Values of OrangePeriodBegin with ARProperties.StartupMode == Legacy	460
Table 583 – Values of OrangePeriodBegin with ARProperties.StartupMode == Advanced	460
Table 584 – Values of GreenPeriodBegin	460
Table 585 – EtherType	460
Table 586 – SyncProperties.Role	461
Table 587 – SyncProperties.SyncID	461
Table 588 – ReservedIntervalBegin with ARProperties.StartupMode == Legacy	461
Table 589 – ReservedIntervalBegin with ARProperties.StartupMode == Advanced	461
Table 590 – ReservedIntervalEnd with ARProperties.StartupMode == Legacy	462
Table 591 – ReservedIntervalEnd with ARProperties.StartupMode == Advanced	462
Table 592 – Dependencies of ReservedIntervalBegin and ReservedIntervalEnd	462
Table 593 – SyncSendFactor	463
Table 594 – PTCPTimeoutFactor	463
Table 595 – PTCPTakeoverTimeoutFactor	464

Table 596 – PTCPMasterStartupTime	465
Table 597 – PLLWindow	465
Table 598 – TimeIObase	467
Table 599 – TimeDataCycle	467
Table 600 – TimeIOInput	467
Table 601 – TimeIOOutput	468
Table 602 – TimeIOInputValid	468
Table 603 – TimeIOOutputValid	468
Table 604 – ControllerApplicationCycleFactor	468
Table 605 – TimePLLWindow	469
Table 606 – TimeMasterPriority1	470
Table 607 – TimeMasterPriority2	470
Table 608 – MRP_RingState	471
Table 609 – MRP_DomainUUID	471
Table 610 – MRP_LengthDomainName	471
Table 611 – MRP_Role	472
Table 612 – MRP_Version	472
Table 613 – MRP_Prio	472
Table 614 – MRP_TOPchgT	472
Table 615 – MRP_TOPNRmax	473
Table 616 – MRP_TSTshortT	473
Table 617 – MRP_TSTdefaultT	473
Table 618 – MRP_TSTNRmax	474
Table 619 – MRP_LNKdownT	474
Table 620 – MRP_LNKupT	474
Table 621 – MRP_LNKNRmax	475
Table 622 – MRP_CheckMediaRedundancyManager	475
Table 623 – MRP_Check.MRP_DomainUUID	475
Table 624 – VendorBlockType	476
Table 625 – FiberOpticType	476
Table 626 – FiberOpticCableType	476
Table 627 – FiberOpticPowerBudgetType.Value	477
Table 628 – FiberOpticPowerBudgetType.CheckEnable	477
Table 629 – NCDropBudgetType.Value	477
Table 630 – NCDropBudgetType.CheckEnable	478
Table 631 – FSHelloMode.Mode	479
Table 632 – FSHelloInterval	480
Table 633 – FSHelloRetry	480
Table 634 – FSHelloDelay	480
Table 635 – FSPParameterMode.Mode	481
Table 636 – FSPParameterUUID	481
Table 637 – NumberOfSubframeBlocks	481
Table 638 – SFIOCRProperties.DistributedWatchDogFactor	482

Table 639 – SFIOCRProperties.RestartFactorForDistributedWD	482
Table 640 – SFIOCRProperties.DFPMODE	483
Table 641 – SFIOCRProperties.DFPDirection	483
Table 642 – SFIOCRProperties.DFPRedundantPathLayout.....	483
Table 643 – SFIOCRProperties.SFCRC16	484
Table 644 – SubframeData.Position	484
Table 645 – SubframeData.DataLength	484
Table 646 – Event function table	485
Table 647 – SubframeOffset	485
Table 648 – Event function table	486
Table 649 – SCFEntry.....	487
Table 650 – ACCommunicationProperties.DFP	488
Table 651 – ACCommunicationProperties.RTC3	488
Table 652 – ACCommunicationProperties.RTCUDP	488
Table 653 – ACMinDeviceInterval	489
Table 654 – FromOffsetData	489
Table 655 – NextOffsetData	489
Table 656 – TotalSize	490
Table 657 – RedundancyInfo.EndPoint1	490
Table 658 – RedundancyInfo.EndPoint2	490
Table 659 – Valid combination of RedundancyInfo.EndPoint1 and RedundancyInfo.EndPoint2	490
Table 660 – SRProperties.InputValidOnBackupAR	491
Table 661 – SRProperties.ActivateRedundancyAlarm	491
Table 662 – RedundancyDataHoldFactor	491
Table 663 – NumberOfEntries	492
Table 664 – ArgsLength check	492
Table 665 – ARBlockReq – request check	493
Table 666 – IOCRBlockReq – request check	494
Table 667 – AlarmCRBlockReq – request check	498
Table 668 – ExpectedSubmoduleBlockReq – request check	498
Table 669 – PrmServerBlock – request check	500
Table 670 – MCRBlockReq – request check	500
Table 671 – ARRPCBlockReq – request check	501
Table 672 – IRInfoBlock – request check	501
Table 673 – SRInfoBlock – request check	502
Table 674 – ArgsLength check	502
Table 675 – ARBlockRes – response check	503
Table 676 – IOCRBlockRes – response check	503
Table 677 – AlarmCRBlockRes – response check	504
Table 678 – ModuleDiffBlock – response check	505
Table 679 – ARServerBlockRes – response check	506
Table 680 – ArgsLength check	506

Table 681 – ControlBlockConnect(PrmEnd) – request check.....	507
Table 682 – ControlBlockPlug(PrmEnd) – request check	507
Table 683 – ControlBlockConnect(PrmBegin) – request check	508
Table 684 – SubmoduleListBlock – request check	508
Table 685 – ArgsLength check.....	508
Table 686 – ControlBlockConnect – response check.....	509
Table 687 – ControlBlockPlug – response check.....	510
Table 688 – ControlBlockConnect(PrmBegin) – response check	510
Table 689 – ArgsLength check.....	511
Table 690 – ControlBlockConnect(AppIRdy) – request check	511
Table 691 – ControlBlockPlug(AppIRdy) – request check.....	512
Table 692 – ArgsLength check.....	512
Table 693 – ControlBlockConnect – response check.....	513
Table 694 – ControlBlockPlug – response check.....	514
Table 695 – ArgsLength check.....	514
Table 696 – ReleaseBlock – request check.....	515
Table 697 – ArgsLength check.....	515
Table 698 – ReleaseBlock – response check	516
Table 699 – ArgsLength check.....	516
Table 700 – IODWriteReqHeader – request check	517
Table 701 – ArgsLength check.....	517
Table 702 – IODWriteResHeader – response check.....	518
Table 703 – ArgsLength check.....	518
Table 704 – ArgsLength check.....	519
Table 705 – ArgsLength check.....	520
Table 706 – IODReadReqHeader – request check	520
Table 707 – RecordDataReadQuery – request check	521
Table 708 – ArgsLength check.....	521
Table 709 – IODReadResHeader – response check.....	522
Table 710 – Primitives issued by AP-Context (FAL user) to FSPMDEV	525
Table 711 – Primitives issued by FSPMDEV to AP-Context (FAL user)	527
Table 712 – Functions, Macros, Timers and Variables used by the AP-Context (FAL user) to FSPMDEV	530
Table 713 – Functions, Macros, Timers and Variables used by the FSPMDEV to AP-Context (FAL user)	531
Table 714 – Primitives issued by AP-Context (FAL user) to FSPMCTL.....	534
Table 715 – Primitives issued by FSPMCTL to AP-Context (FAL user).....	536
Table 716 – Functions, Macros, Timers and Variables used by AP-Context (FAL user) to FSPMCTL	540
Table 717 – Functions, Macros, Timers and Variables used by FSPMCTL to AP-Context (FAL user)	541
Table 718 – Remote primitives issued or received by ALPMI	544
Table 719 – Local primitives issued or received by ALPMI	545
Table 720 – ALPMI state table	546

Table 721 – Functions, Macros, Timers and Variables used by ALPMI	548
Table 722 – Remote primitives issued or received by ALPMR	548
Table 723 – Local primitives issued or received by ALPMR	549
Table 724 – ALPMR state table	551
Table 725 – Functions, Macros, Timers and Variables used by ALPMR	553
Table 726 – Remote primitives issued or received by CMDEV	556
Table 727 – Local primitives issued or received by CMDEV	558
Table 728 – CMDEV state table	560
Table 729 – Functions, Macros, Timers and Variables used by CMDEV	563
Table 730 – Remote primitives issued or received by CMDEV_DA	564
Table 731 – Local primitives issued or received by CMDEV_DA	565
Table 732 – CMDEV_DA state table	566
Table 733 – Functions, Macros, Timers and Variables used by CMDEV(DA)	567
Table 734 – Remote primitives issued or received by CMSU	567
Table 735 – Local primitives issued or received by CMSU	567
Table 736 – CMSU state table	570
Table 737 – Functions, Macros, Timers and Variables used by the CMSU	572
Table 738 – Remote primitives issued or received by CMIO	573
Table 739 – Local primitives issued or received by CMIO	573
Table 740 – CMIO state table	574
Table 741 – Functions used by the CMIO	576
Table 742 – Remote primitives issued or received by CMWRR	576
Table 743 – Local primitives issued or received by CMWRR	577
Table 744 – CMWRR state table	579
Table 745 – Functions, Macros, Timers and Variables used by CMWRR	580
Table 746 – Remote primitives issued or received by CMRDR	581
Table 747 – Local primitives issued or received by CMRDR	581
Table 748 – CMRDR state table	582
Table 749 – Functions, Macros, Timers and Variables used by CMRDR	583
Table 750 – Remote primitives issued or received by CMSM	583
Table 751 – Local primitives issued or received by CMSM	584
Table 752 – CMSM state table	585
Table 753 – Functions, Macros, Timers and Variables used by the CMSM	586
Table 754 – Remote primitives received by CMPBE	586
Table 755 – Local primitives issued or received by CMPBE	587
Table 756 – CMPBE state table	589
Table 757 – Functions, Macros, Timers and Variables used by the CMPBE	591
Table 758 – Remote primitives issued or received by CMDMC	591
Table 759 – Local primitives issued or received by CMDMC	591
Table 760 – CMDMC state table	594
Table 761 – Functions, Macros, Timers and Variables used by the CMDMC	596
Table 762 – Remote primitives issued or received by CMINA	596
Table 763 – Local primitives issued or received by CMINA	597

Table 764 – CMINA state table	598
Table 765 – Functions, Macros, Timers and Variables used by the CMINA	603
Table 766 – Return values of CheckAPDU	603
Table 767 – Remote primitives issued or received by CMRPC	604
Table 768 – Local primitives issued or received by CMRPC	606
Table 769 – CMRPC state table	606
Table 770 – Functions, Macros, Timers and Variables used by the CMRPC	610
Table 771 – Return values of CheckRPC	611
Table 772 – Remote primitives issued or received by CMCTL	614
Table 773 – Local primitives issued or received by CMCTL	615
Table 774 – CMCTL state table	618
Table 775 – Functions, Macros, Timers and Variables used by the CMCTL	621
Table 776 – Remote primitives issued or received by CTLSM	622
Table 777 – Local primitives issued or received by CTLSM	623
Table 778 – CTLSM state table	623
Table 779 – Functions, Macros, Timers and Variables used by the CTLSM	624
Table 780 – Remote primitives issued or received by CTLIO	625
Table 781 – Local primitives issued or received by CTLIO	625
Table 782 – CTLIO state table	626
Table 783 – Functions, Macros, Timers and Variables used by the CTLIO	628
Table 784 – Remote primitives received by CTLRDI	628
Table 785 – Local primitives issued or received by CTLRDI	629
Table 786 – CTLRDI state table	630
Table 787 – Functions, Macros, Timers and Variables used by CTLRDI	631
Table 788 – Remote Primitives received by CTLRDR	631
Table 789 – Local primitives issued or received by CTLRDR	632
Table 790 – CTLRDR state table	632
Table 791 – Functions, Macros, Timers and Variables used by CTLRDR	633
Table 792 – Remote primitives received by CTLRPC	633
Table 793 – Local primitives issued or received by CTLRPC	636
Table 794 – CTLRPC state table	637
Table 795 – Functions, Macros, Timers and Variables used by the CTLRPC	639
Table 796 – Remote primitives issued or received by CTLSU	639
Table 797 – Local Primitives issued or received by CTLSU	640
Table 798 – CTLSU state table	642
Table 799 – Functions, Macros, Timers and Variables used by the CTLSU	644
Table 800 – Remote primitives issued or received by CTLWRI	645
Table 801 – Local primitives issued or received by CTLWRI	645
Table 802 – CTLWRI state table	647
Table 803 – Functions, Macros, Timers and Variables used by CTLWRI	648
Table 804 – Remote primitives issued or received by CTLWRR	649
Table 805 – Local primitives issued or received by CTLWRR	649
Table 806 – CTLWRR state table	650

Table 807 – Functions, Macros, Timers and Variables used by CTLWRR.....	651
Table 808 – Remote primitives issued or received by CTLPBE	651
Table 809 – Local primitives issued or received by CTLPBE	652
Table 810 – CTLPBE state table	654
Table 811 – Functions, Macros, Timers and Variables used by CTLPBE.....	656
Table 812 – Remote primitives issued or received by CTLDINA	656
Table 813 – Local primitives issued or received by CTLDINA.....	657
Table 814 – CTLDINA state table.....	659
Table 815 – Functions, Macros, Timers and Variables used by the CTLDINA.....	662
Table A.1 – Examples for the AR establishing.....	664
Table B.1 – Examples for compatible AR establishing.....	670
Table I.1 – Priority regeneration and queue usage	685
Table I.2 – Priority regeneration – variant 1	685
Table I.3 – Priority regeneration – variant 2	686
Table R.1 – Truth table	705
Table S.1 – TX-error	706
Table U.1 – List of supported MIBs	708
Table V.1 – Content of archive.....	709

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 6-10: Application layer protocol specification –
Type 10 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.

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 IEC 61784-1 and IEC 61784-2.

International Standard IEC 61158-6-10 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This third edition cancels and replaces the second edition published in 2010. This edition constitutes a technical revision.

The main changes with respect to the previous edition are listed below:

- Corrections and Improvements

- Change from MRP integration to MRP reference
- Integration of dynamic frame packing
- Integration of peer to peer fragmentation
- Integration of fast forwarding
- Integration of shared RT_CLASS_3 ARs
- Integration of vendor specific blocks for the connect
- Integration of generic POF diagnosis
- Integration of autoconfiguration
- Integration of seamless media redundancy MRPD
- Integration of the System redundancy basic functionality
- Integration of the Configure in run basic functionality
- Integration of multiple interface support
- Integration of port statistic for error tracking
- Integration of controller to controller communication basic functionality
- Optimization of RT_CLASS_3 startup and forwarding
- Optimization of isochronous startup
- Optimization of the startup time from power down
- Removal of MRRT
- Removal of distributed automation
- Update of the LLDP-EXT-MIB

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

FDIS	Report on voting
65C/764/FDIS	65C/774/RVD

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

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

A list of all the 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 publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

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

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

Withdrawing
IECNORM.COM : Click to view the full PDF of IEC 61158-6-10:2014

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 application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) 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:

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

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

NOTE Attention is drawn to the fact that use of the associated protocol type(s) is restricted by its (their) intellectual-property-right holder(s). In all cases, the commitment to limited release of intellectual-property-rights made by the holder(s) of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in type combinations as specified explicitly in the IEC 61784 series. Use of the protocol type(s) in other combinations may require permission of their respective intellectual-property-right holder(s).

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 concerning Type 10 elements and possibly other types given in this standard as follows:

The following patent rights for Type 10 have been announced by [SI]:

WO publication	Title (WO)
WO 02/043336 EP 1388238	System and method for parallel transfer of real-time critical and non-real-time critical data via switchable data networks, particularly Ethernet
WO 02/076033 EP 1368935	Synchronous clocked communication system with decentralized input/output modules and methods for integrating decentralized input/output modules in such a system
WO 03/028258 EP 1430628	Method for synchronizing nodes of a communication system
WO 03/028259 EP 1430627	Communications system and method for synchronizing a communications cycle
WO 04/030284 EP 1540895	Method for permanent redundant transmission of data telegrams in communication systems
EP 1558002	Method for assigning an IP address to a device
EP 1318630	Matrices for controlling the device specific data transfer rates on a field bus

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

The holders of these patent rights have assured the IEC that they are willing to negotiate licences either free of charge or under reasonable and non-discriminatory terms and

conditions with applicants throughout the world. In this respect, the statement of the holders of these patent rights is registered with IEC. Information may be obtained from:

[SI]: Siemens AG
CT IP L&T
Otto-Hahn-Ring 6
D-81739 Munich
Germany

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

ISO (www.iso.org/patents) and IEC (<http://patents.iec.ch>) maintain on-line data bases of patents relevant to their standards. Users are encouraged to consult the data bases for the most up to date information concerning patents.

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-10: Application layer protocol specification – Type 10 elements

1 Scope

1.1 General

The Fieldbus Application Layer (FAL) provides user programs with a means to access the fieldbus communication environment. In this respect, the FAL can be viewed as a “window between corresponding application programs.”

This standard provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 10 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This standard defines in an abstract way the externally visible behavior provided by the Type 10 fieldbus application layer in terms of

- a) the abstract syntax defining the application layer protocol data units conveyed between communicating application entities,
- b) the transfer syntax defining the application layer protocol data units conveyed between communicating application entities,
- c) the application context state machine defining the application service behavior visible between communicating application entities, and
- d) the application relationship state machines defining the communication behavior visible between communicating application entities.

The purpose of this standard is to define the protocol provided to

- a) define the wire-representation of the service primitives defined in IEC 61158-5-10 and
- b) define the externally visible behavior associated with their transfer.

This standard specifies the protocol of the Type 10 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI Application Layer Structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this standard is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-10.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of protocols standardized in IEC 61158-6.

1.3 Conformance

This standard does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems. Conformance is achieved through implementation of this application layer protocol specification.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. 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 IEC 61784-1 and IEC 61784-2 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 61158-1:2014, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-2:2014, *Industrial communication networks – Fieldbus specifications – Part 2: Physical layer specification and service definition*

IEC 61158-5-10:2014, *Industrial communication networks – Fieldbus specifications – Application layer service definition – Type 10 elements*

IEC 61158-6-3:2014, *Industrial communication networks – Fieldbus specifications – Part 6-3: Application layer protocol specification – Type 3 elements*

IEC 61588, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 62439-2, *Industrial communication networks – High availability automation networks – Part 2: Media Redundancy Protocol (MRP)*

ISO/IEC 646:1991, *Information technology – ISO 7-bit coded character set for information interchange*

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

ISO/IEC 8822, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation*

ISO/IEC 9545, *Information technology – Open Systems Interconnection – Application Layer structure*

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

ISO 8601, *Data elements and interchange formats – Information interchange – Representation of dates and times*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, available at <<http://www.ieee.org>>

IEEE 802, *IEEE Standard for Local and metropolitan area networks: Overview and Architecture*, available at <<http://www.ieee.org>>

IEEE 802.1AB-2005, *IEEE Standard for Local and metropolitan area networks: Station and Media Access Control Connectivity Discovery*, available at <<http://www.ieee.org>>

IEEE 802.1AS, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – IEEE standard for Local and metropolitan area networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, available at <<http://www.ieee.org>>

IEEE 802.1D, *IEEE Standard for Local and metropolitan area networks – Media access control (MAC) Bridges*, available at <<http://www.ieee.org>>

IEEE 802.1Q, *IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*, available at <<http://www.ieee.org>>

IEEE 802.3, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, available at <<http://www.ieee.org>>

IEEE 802.11, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, available at <<http://www.ieee.org>>

IEEE 802.15.1-2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*, available at <<http://www.ieee.org>>

IETF RFC 768, *User Datagram Protocol*, available at <<http://www.ietf.org>>

IETF RFC 791, *Internet Protocol*, available at <<http://www.ietf.org>>

IETF RFC 792, *Internet Control Message Protocol*, available at <<http://www.ietf.org>>

IETF RFC 826, *An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*, available at <<http://www.ietf.org>>

IETF RFC 1034, *Domain names – concepts and facilities*, available at <<http://www.ietf.org>>

IETF RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, available at <<http://www.ietf.org>>

IETF RFC 2131, *Dynamic Host Configuration Protocol*, available at <<http://www.ietf.org>>

IETF RFC 2132, *DHCP Options and BOOTP Vendor Extensions*, available at <<http://www.ietf.org>>

IETF RFC 2236, *Internet Group Management Protocol, Version 2*, available at <<http://www.ietf.org>>

IETF RFC 2365, *Administratively Scoped IP Multicast*, available at <<http://www.ietf.org>>

IETF RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, available at <<http://www.ietf.org>>

IETF RFC 2674, *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions*, available at <<http://www.ietf.org>>

IETF RFC 2863, *The Interfaces Group MIB*, available at <<http://www.ietf.org>>

IETF RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*, available at <<http://www.ietf.org>>

IETF RFC 3621, *Power Ethernet MIB*, available at <<http://www.ietf.org>>

IETF RFC 4361, *Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)*, available at <<http://www.ietf.org>>

IETF RFC 4363, *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering, and Virtual LAN Extensions*, available at <<http://www.ietf.org>>

IETF RFC 4836, *Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)*, available at <<http://www.ietf.org>>

IETF RFC 5735, *Special-Use IPv4 Addresses*, available at <<http://www.ietf.org>>

IETF RFC 5890, *Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework*, available at <<http://www.ietf.org>>

The Open Group — Publication C706, *Technical standard DCE1.1: Remote Procedure Call*, available at <<http://www.opengroup.org/onlinepubs/9629399/toc.htm>>

3 Terms, definitions, abbreviations, symbols and conventions

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

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 7498-1 apply:

- a) application entity
- b) application process
- c) application protocol data unit
- d) application service element
- e) application entity invocation
- f) application process invocation
- g) application transaction
- h) real open system

- i) transfer syntax

3.1.2 ISO/IEC 8822 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8822 apply:

- a) abstract syntax
- b) presentation context

3.1.3 ISO/IEC 8824-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824 apply:

- a) object identifier
- b) type

3.1.4 ISO/IEC 9545 terms

For the purposes of this document, the following terms as defined in ISO/IEC 9545 apply:

- a) application-association
- b) application-context
- c) application context name
- d) application-entity-invocation
- e) application-entity-type
- f) application-process-invocation
- g) application-process-type
- h) application-service-element
- i) application control service element

3.2 Additional terms and definitions for decentralized periphery

3.2.1

alarm

activation of an event that shows a critical state

3.2.2

alarm ack

acknowledgment of an event that shows a critical state

3.2.3

alarm data object

object(s) which represent critical states referenced by device/API/slot/subslot/alarm type

3.2.4

allocate

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

3.2.5

application

function or data structure for which data is consumed or produced

3.2.6

application layer interoperability

capability of application entities to perform coordinated and cooperative operations using the services of the FAL

3.2.7

application objects

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

3.2.8

application process

part of a distributed application on a network, which is located on one device and unambiguously addressed

3.2.9

application process identifier

distinguishes multiple application processes used in a device

3.2.10

application process object

component of an application process that is identifiable and accessible through an FAL application relationship

Note 1 to entry: Application process object definitions are composed of a set of values for the attributes of their class (see the definition for Application Process Object Class Definition). Application process object definitions may be accessed remotely using the services of the FAL Object Management ASE. FAL Object Management services can be used to load or update object definitions, to read object definitions and to dynamically create and delete application objects and their corresponding definitions.

3.2.11

application process object class

class of application process objects defined in terms of the set of their network-accessible attributes and services

3.2.12

application relationship

cooperative association between two or more application-entity-invocations for the purpose of exchange of information and coordination of their joint operation. This relationship is activated either by the exchange of application-protocol-data-units or as a result of preconfiguration activities

3.2.13

application relationship application service element

application-service-element that provides the exclusive means for establishing and terminating all application relationships

3.2.14

application relationship endpoint

context and behavior of an application relationship as seen and maintained by one of the application processes involved in the application relationship

Note 1 to entry: Each application process involved in the application relationship maintains its own application relationship endpoint.

3.2.15

attribute

description of an externally visible characteristic or feature of an object

Note 1 to entry: The attributes of an object contain information about variable portions of an object. Typically, they provide status information or govern the operation of an object. Attributes may also affect the behavior of an object. Attributes are divided into class attributes and instance attributes.

3.2.16

backup

status of the IO AR, which indicates that it, is in the standby state

3.2.17**behavior**

indication of how an object responds to particular events

3.2.18**channel**

representation of a single physical or logical link of an input or output application object of a server to the process in order to support addressing of diagnosis information

Note 1 to entry: The channel typically represents a single connector or clamp as a real interface of a module or submodule. This reference is used to identify points of failure within diagnosis PDUs.

3.2.19**channel diagnosis**

information concerning a specific element of an input or output application object, provided for maintenance purposes

EXAMPLE line break

3.2.20**class**

set of objects, all of which represent the same kind of system component

Note 1 to entry: A class is a generalization of an object; a template for defining variables and methods. All objects in a class are identical in form and behavior, but usually contain different data in their attributes.

3.2.21**class attributes**

attribute that is shared by all objects within the same class

3.2.22**class code**

unique identifier assigned to each object class

3.2.23**class specific service**

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

Note 1 to entry: A class specific object is unique to the object class which defines it.

3.2.24**clear**

status of the IO controller, which indicates that the control algorithm is currently not running

3.2.25**client**

- a) object which uses the services of another (server) object to perform a task
- b) initiator of a PDU to which a server reacts

3.2.26**common profile**

collection of device independent information and functionality providing consistency between all devices

3.2.27**communication data object**

object(s) which are parameter of communication relationships and referenced by device/ slot/ subslot/ index

3.2.28

configuration check

comparison of the expected IO-Data object structuring of the client with the real IO-Data object structuring to the server in the start-up phase

3.2.29

configuration fault

unacceptable difference between the expected IO-Data object structuring and the real IO-Data object structuring, as detected by the server

3.2.30

configuration identifier

representation of a portion of IO Data of a single input- and/or output-module of a server

3.2.31

consume

act of receiving data from a provider

3.2.32

consumer

node or sink receiving data from a provider

3.2.33

context management

network-accessible information (communication objects) that supports managing the operation of the fieldbus system, including the application layer

Note 1 to entry: Managing includes functions such as controlling, monitoring and diagnosing.

3.2.34

conveyance path

unidirectional flow of APDUs across an application relationship

3.2.35

cyclic

repetitive in a regular manner

3.2.36

data consistency

means for coherent transmission and access of the input- or output-data object between and within client and server

3.2.37

device

physical hardware connected to the link

Note 1 to entry: A device may contain more than one node.

3.2.38

device ID

vendor assigned device type identification

3.2.39

device profile

collection of device dependent information and functionality providing consistency between similar devices of the same device type

3.2.40**diagnosis data object**

object(s) which contains diagnosis information referenced by device/API/slot/subslot/index

3.2.41**diagnosis information**

all data available at the server for maintenance purposes

3.2.42**dynamic reconfiguration**

change of IO data objects without interruption of an established application relationship and continuous updating of non-changed IO data objects

3.2.43**endpoint**

one of the communicating entities involved in a connection

3.2.44**engineering**

abstract term that characterizes the client application or device responsible for configuring an automation system

3.2.45**error**

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

3.2.46**error class**

general grouping for related error definitions and corresponding error codes

3.2.47**error code**

identification of a specific type of error within an error class

3.2.48**event**

instance of a change of conditions

3.2.49**extended channel diagnosis**

information concerning a specific element of a specific application object, provided for maintenance purposes

EXAMPLE Link Fail

3.2.50**frame**

unit of data transfer on a link

3.2.51**identification data object**

object(s) that contain information about device, module and submodule manufacturer and type referenced by device/API/slot/subslot/index

3.2.52**implicit AR endpoint**

AR endpoint that is defined locally within a device without use of the create service

3.2.53

inbound

input communication relation from IO device to IO controller

3.2.54

index

address of a record data object within an application process

3.2.55

instance

the actual physical occurrence of an object within a class that identifies one of many objects within the same object class

3.2.56

instance attributes

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

3.2.57

instantiated

object that has been created in a device

3.2.58

invocation

act of using a service or other resource of an application process

Note 1 to entry: Each invocation represents a separate thread of control that may be described by its context. Once the service completes, or use of the resource is released, the invocation ceases to exist. For service invocations, a service that has been initiated but not yet completed is referred to as an outstanding service invocation. Also for service invocations, an Invoke ID may be used to unambiguously identify the service invocation and differentiate it from other outstanding service invocations.

3.2.59

IO controller

controlling device, which acts as client for several IO devices (field devices)

Note 1 to entry: This is usually a programmable controller or a distributed control system.

3.2.60

IO data object

object designated to be transferred cyclically for the purpose of processing and referenced by device/API/slot/subslot

3.2.61

IO device

field device which acts as server for IO operation

3.2.62

IO parameter server

server for application parameter of IO devices (client)

Note 1 to entry: This is usually a device to backup parameter data and to log online changes of device parameter.

3.2.63

IO subsystem

subsystem composed of one IO controller and all its associated IO devices

3.2.64

IO supervisor

engineering device which manages commissioning and diagnosis of an IO system

3.2.65**IO system**

system composed of all its IO subsystems

Note 1 to entry: As an example a Programmable Logic Controller with more than one IO controller (network interface) controls one IO system composed of an IO subsystems for each IO controller.

3.2.66**Isochronous mode**

IO system operating tightly synchronized with a jitter of less than 1 μ s

3.2.67**member**

piece of an attribute that is structured as an element of an array

3.2.68**message**

synonym for frame

3.2.69**method**

synonym for an operational service which is provided by the server ASE and invoked by a client

3.2.70**module**

hardware or logical component of a physical device

3.2.71**network**

set of nodes connected by some type of communication medium, including any intervening repeaters, bridges, routers and lower-layer gateways

3.2.72**object**

abstract representation of a particular component within a device, usually a collection of related data (in the form of variables) and methods (procedures) for operating on that data that have clearly defined interface and behavior

3.2.73**object specific service**

service unique to the object class which defines it

3.2.74**operate**

status of the IO controller that indicates that the control algorithm is currently running

3.2.75**outbound**

output communication relation from an IO controller to an IO device

3.2.76**packet**

frame

3.2.77**peer**

role of an AR endpoint in which it is capable of acting as both client and server

3.2.78

physical device

automation or other network device

3.2.79

point-to-point connection

connection that exists between exactly two application objects

3.2.80

primary

status of the IO AR that indicates that it is in the operating state

Note 1 to entry: Besides a primary IO AR a backup IO AR may exist. In example used for redundancy and dynamic reconfiguration of IO data.

3.2.81

provider

node or source sending data to one or many consumer

3.2.82

PTCP subdomain

certain amount of DTEs with synchronized clocks

3.2.83

qualified channel diagnosis

information concerning a specific element of a specific application object, provided for maintenance purposes with a parameterized severity

EXAMPLE Overload

3.2.84

record data object

object(s) which are already pre-processed and transferred acyclically for the purpose of information or further processing and referenced by device/API/slot/subslot/index

3.2.85

resource

processing or information capability

3.2.86

run

status of the IO controller which indicates that the control algorithm is currently operating

3.2.87

server

- a) role of an AREP in which it returns a confirmed service response APDU to the client that initiated the request
- b) object which provides services to another (client) object

3.2.88

service

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

3.2.89

slot

address of a structural unit within an IO device

Note 1 to entry: Within a modular device, a slot typically addresses a physical module. Within compact devices, a slot typically addresses a logical function or virtual module.

3.2.90

stop

status of the IO controller which indicates that the control algorithm is currently not running

3.2.91

submodule

hardware or logical component of a module

3.2.92

subslot

address of a structural unit within a slot

Note 1 to entry: A subslot may address a physical interface for submodules within a module. Generally, a subslot is a second level to structure data within a device.

3.2.93

vendor ID

central administrative number used as manufacturer identification

3.3 Abbreviations and symbols

3.3.1 Abbreviations and symbols for media redundancy

MRC	Media Redundancy Client
MRM	Media Redundancy Manager
MRP	Media Redundancy Protocol
MRPD	Media Redundancy with Planned Duplication of frames

NOTE The Media Redundancy with Planned Duplication of frames is done by the sender using different paths for the adjunctive frames.

3.3.2 Abbreviations and symbols for decentralized periphery

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
ALPMI	Alarm Protocol Machine Initiator
ALPMR	Alarm Protocol Machine Responder
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
APM	Acyclic Protocol Machine
APMR	Acyclic Protocol Machine Receiver
APMS	Acyclic Protocol Machine Sender
APO	Application Object
AR	Application Relationship
AREP	Application Relationship End Point
ARP	Address Resolution Protocol
ARPM	Address Resolution Protocol Machine
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
BLOB	Binary Large Object
BMA	Best-Master-Algorithm Protocol Machine
BMC	Best Master Clock
CLRPC	Connectionless Remote Procedure Call
CL-RPC	Connectionless Remote Procedure Call
CM	Context Management
CMCTL	Context Management Protocol Machine Controller

CMDEV	Context Management Protocol Machine Device
CMDEV_DA	Context Management Device Access Protocol Machine Device
CMDMC	Discovery Multicast Communication Protocol Machine
CMINA	Context Management IP and Name Assignment Protocol Machine
CMIO	Context Management Input Output Protocol Machine Device
CMPBE	Context Management Prm Begin End Protocol Machine Device
CMRDR	Context Management Read Record Responder Protocol Machine Device
CMRPC	Context Management RPC Device Protocol Machine
CMSM	Context Management Surveillance Protocol Machine Device
CMSU	Context Management Startup Protocol Machine Device
CMWRR	Context Management Write Record Responder Protocol Machine Device
CPM	Consumer protocol machine for cyclic transmitted data
CR	Communication Relationship
CREP	Communication Relationship End Point
CT	Cut Through switching
CTLDINA	Context Management Discovery, IP and Name Assignment
CTLIO	Context Management Input Output Protocol Machine Controller
CTLPBE	Context Management Prm Begin End Protocol Machine Controller
CTLRDI	Context Management Read Record Initiator Protocol Machine Controller
CTLRDR	Context Management Read Record Responder Protocol Machine Controller
CTLRPC	Context Management RPC Protocol Machine Controller
CTLSM	Context Management Surveillance Protocol Machine Controller
CTLSU	Context Management Startup Protocol Machine Controller
CTLWRI	Context Management Write Record Initiator Protocol Machine Controller
CTLWRR	Context Management Write Record Responder Protocol Machine Controller
DCE	OSF Distributed Computing Environment
DCP	Discovery and basic Configuration Protocol
DCPHMCS	DCP Hello Multicast Sender Protocol Machine
DCPHMRC	DCP Hello Multicast Receiver Protocol Machine
DCPMCR	DCP Multicast Receiver Protocol Machine
DCPMCS	DCP Multicast Sender Protocol Machine
DCPUCR	DCP Unicast Receiver Protocol Machine
DCPUCS	DCP Unicast Sender Protocol Machine
DEFRAG	Defragmentation Protocol Machine
DELAY_REQ	Line Delay Request Protocol Machine
DELAY_RSP	Line Delay Response Protocol Machine
DEMUX	Network Access Receiver
DFP	Dynamic Frame Packing
DFP_RELAY	DFP Protocol Machine
DFP_RELAY_IN_STORAGE	DFP Inbound Storage Protocol Machine
DFP_RELAY_INBOUND	DFP Inbound Protocol Machine
DFP_RELAY_OUTBOUND	DFP Outbound Protocol Machine
DHCP	Dynamic Host Configuration Protocol
DIM	Device Interface Module
DLC	Data Link Connection
DLL	Data Link Layer
DLPDU	Data Link-Protocol Data Unit
DLSDU	Data Link-service-data-unit

DMPM	DLL Mapping Protocol Machines
DNS	Domain Name Service
DTE	Date Terminal Equipment
FAL	Fieldbus Application Layer
FDB	Filtering Data Base
FIFO	First In First Out
FRAG	Fragmentation Protocol Machine
FSPM	FAL Service Protocol Machines
FSPMCTL	FSPM Controller
FSPMDEV	FSPM Device
GBSM	Get Best Sync Master Protocol Machine
GSDML	General Station Description Markup Language
I&M	Identification and Maintenance Profile
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ID	Identifier
IEC	International Electrotechnical Commission
IFW	RT_CLASS_3 Forwarding Protocol Machine
IOC	Input Output Controller
IOCR	IO Communication Relation
IOCS	Input Output Object Consumer Status
IOD	Input Output Device
IOMCR	IO Multicast Communication Relation
IOPS	Input Output Object Provider Status
IOS	Input Output Supervisor
IP	Internet Protocol
IR	Isochronous Relay
IRT	Isochronous Real Time Protocol
ISO	International Organization for Standardization
IsoM	Isochronous Mode
LED	Light Emitting Diode
LLDP	Link Layer Discovery Protocol
LME	Layer Management Entity
LMPM	Data Link Layer Mapping Protocol Machine
lsb	Least Significant Bit
LT	Length/Type
MAC	Media Access Control
MAC_RELAY	Forwarding Protocol Machine
MCPM	Multicast Consumer protocol machine for cyclic transmitted data
MCR	Multicast communication relation
MPPM	Provider protocol machine for cyclic transmitted data (PPM) using Multicast communication relation
msb	most significant bit
MUX	Network Access Sender
NAP	Network Access Point
NCA	Network Computing Architecture
OSF	Open Software Foundation
OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PL	Physical Layer
PPM	Provider protocol machine for cyclic transmitted data
PTCP	Precision Transparent Clock Protocol
QoS	Quality of Service
RED_RELAY	Realtime Class 3 Forwarding Protocol Machine
RPC	Remote Procedure Call
RSM	Best Remote Sync Master Protocol Machine
RT	Real Time Protocol
RT_CLASS_UDP	Realtime Class UDP (supports Layer 3 transport)
RT_CLASS_1	Realtime Class 1 (Layer 2 transport)
RT_CLASS_2	Realtime Class 2 (Layer 2 transport)
RT_CLASS_3	Realtime Class 3 (Layer 2 transport)

RTA	Real Time Protocol Acyclic
RTC	Real Time Protocol Cyclic
RTC3PSM	Realtime Class 3 Port State Machine
S&F	Store and Forward
SCHEDULER	Scheduler Protocol Machine
SDU	Service Data Unit
SYN_BMA	Best-Master-Algorithm Protocol Machine
SYN_BMA_GBSM	Get Best Sync Master
SYN_BMA_RSM	Best Remote Sync Master
SYN_MPSM	PTCP Master Protocol Machine
SYN_SPSM	PTCP Slave Protocol Machine
SYNC_RELAY	Sync Relay Protocol Machine
TLV	Type Length Value (coding rule)
UDP	User Datagram Protocol
UUID	Universal Unique Identifier
VLAN	Virtual Local Area Network

3.3.3 Abbreviations and symbols for services

cnf	Confirmation
ind	Indication
req	Request
rsp	Response

3.4 Conventions

3.4.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate clause. Each ASE specification is composed of three parts: its class definitions, its services and its protocol specification. The first two are contained in IEC 61158-5-10. The protocol specification for each of the ASEs is defined in this standard.

The class definitions define the attributes of the classes supported by each ASE. The attributes are accessible from instances of the class using the Management ASE services specified in IEC 61158-5-10 standard. The service specification defines the services that are provided by the ASE.

This standard uses the descriptive conventions given in ISO/IEC 10731.

3.4.2 Conventions for decentralized periphery

3.4.2.1 Abstract syntax conventions

3.4.2.1.1 PDUs are described as octets or groups of octets

- Groups of octets separated by a comma appear in the order they are transferred. If optional octets are not present the following octets appear without a gap.
- If octets or groups of octets are grouped within "{ }" the order is arbitrary.
- If octets or groups of octets are marked with "*" they may appear more than once. If it is used within a "{ }" section they may appear mixed with other octets or group of octets of this section.
- Octets can be grouped or values can be assigned within "()"
- If octets or groups of octets are grouped within "[]" the group can be omitted. This doesn't imply that the support of these octets is optional.
- Complex APDUs may be built by means of substitutions (sub-structures)
- Exclusive selections of octets or groups of octets are separated by "^"

NOTE 1 The formal PDU example

AP_PDU = Octet1, OctetGroup1, [Octet2], [Octet3], {[OctGroup2*], OctetGroup3 ^ Octet4}

According to this the following variants are valid on the wire (non exhaustive):

Variant 1: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2, OctetGroup3

Variant 2: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup3

Variant 3: Octet1, OctetGroup1, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup3, OctetGroup2

Variant 4: Octet1, OctetGroup1, OctetGroup2, OctetGroup3, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup2

Variant 5: Octet1, OctetGroup1, Octet3, Octet4

NOTE 2 The arbitrary order implies that groups of octets are characterised by a special header that is described within the coding rules.

NOTE 3 The APDU syntax for RTA- and RTC-PDU implies that according to the maximum DLSDU an APDU does not exceed 1 440 octets in total.

NOTE 4 The APDU syntax for CL-RPC implies that an IO controller supports a minimal ASDU size of 4 096 octets in total and does not exceed $2^{32}-64$ octets in total. The minimal ASDU size is derived from the expected size of configuration, parameter and diagnosis data of an enhanced IO device.

3.4.2.2 Convention for the encoding of reserved bits and octets

The term “reserved” may be used to describe bits in octets or whole octets. All bits or octets that are reserved should be set to zero at the sending side and shall not be tested at the receiving side except it is explicitly stated or if the reserved bits or octets are checked by a state machine.

The term “reserved” may also be used to indicate that certain values within the range of a parameter are reserved for future extensions. In this case the reserved values should not be used at the sending side and shall not be tested at the receiving side except it is explicitly stated or if the reserved values are checked by a state machine.

3.4.2.3 Conventions for the common codings of specific field octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the order according to Figure 1.

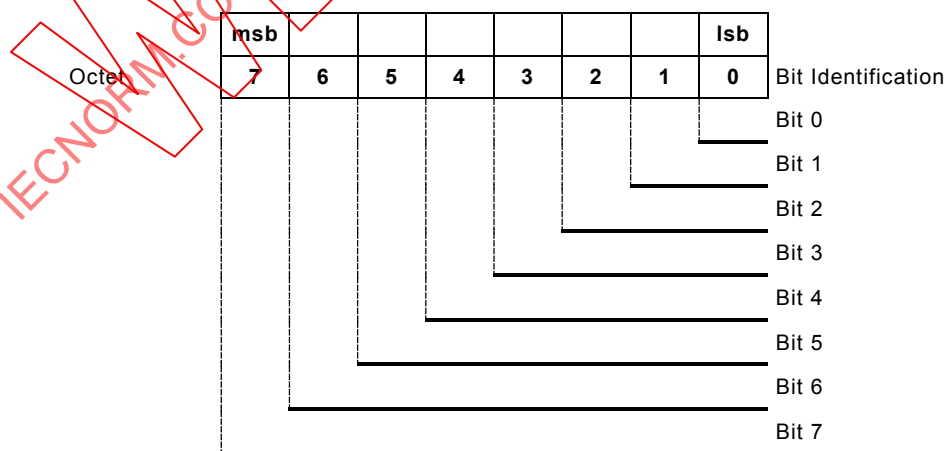


Figure 1 – Common structure of specific fields

Several bit may be grouped as group of bit. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value

shall only be valid for the grouped bits and can only represent the whole octet if all 8 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bit.

EXAMPLE 1 Description and relation for the specific field octet

Bit 0: reserved.

Bit 1-3: Reason_Code The decimal value 2 for the Reason_Code means general error.

Bit 4-7: shall always set to one.

The octet that is constructed according to the description above looks as follows:

(msb) Bit 7 = 1,

Bit 6 = 1,

Bit 5 = 1,

Bit 4 = 1,

Bit 3 = 0,

Bit 2 = 1,

Bit 1 = 0,

(lsb) Bit 0 = 0.

The bit combination "0-1-0" for Bit 1-3 equals the decimal value 2.

3.4.2.4 Conventions for the common codings of specific field consisting of two subsequent octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the order according to Figure 2 and Figure 3.

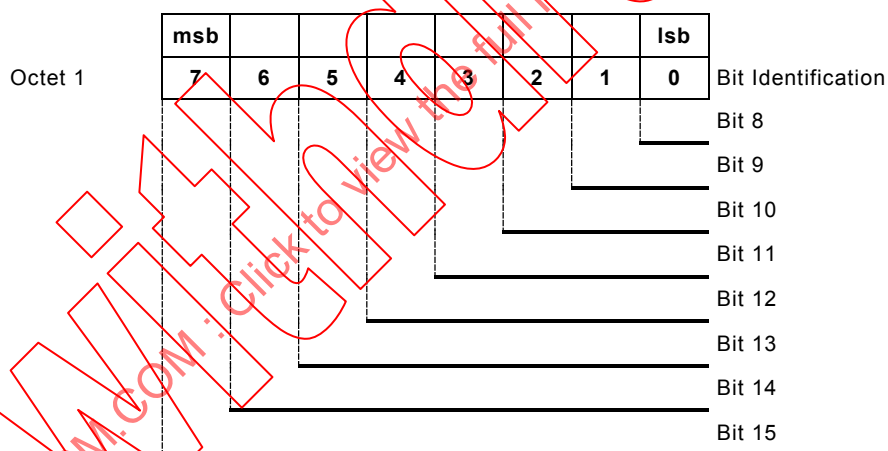


Figure 2 – Common structure of specific fields for octet 1 (high)

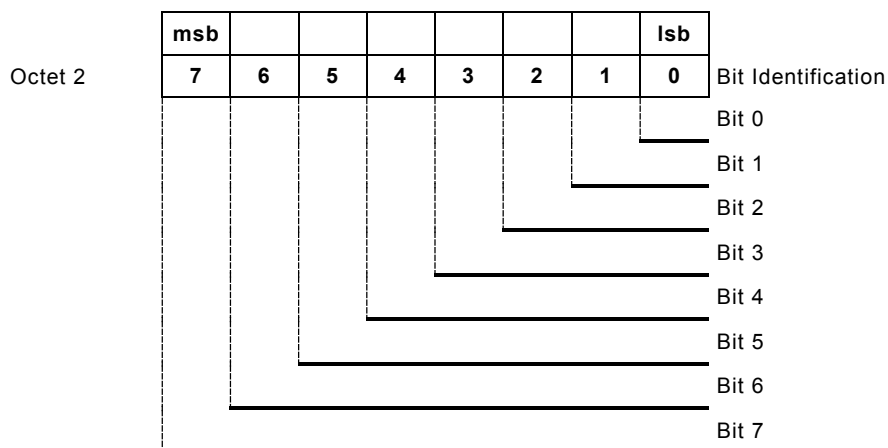


Figure 3 – Common structure of specific fields for octet 2 (low)

Several bits may be grouped as group of bit. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 16 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bit.

3.4.2.5 Conventions for the common coding of specific field consisting of four subsequent octets

APDUs may contain specific fields that carry information in a primitive and condensed way. These fields shall be coded in the order according to Figure 4, Figure 5, Figure 6 and Figure 7.

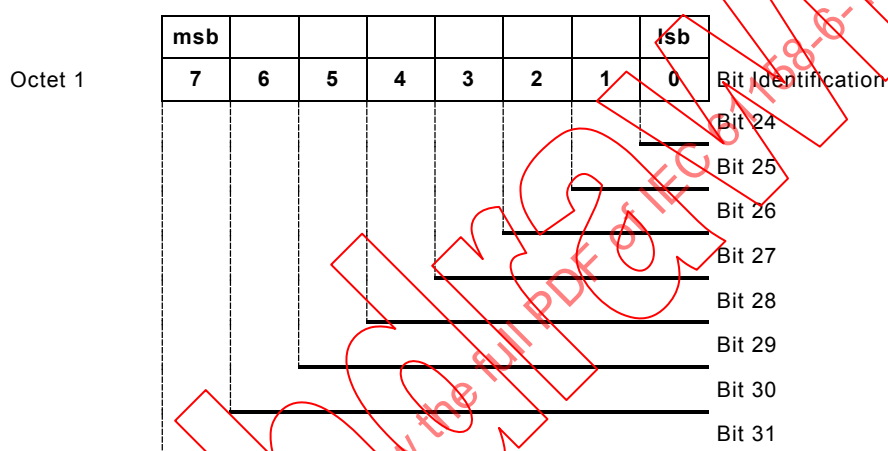


Figure 4 – Common structure of specific fields for octet 1 (high)

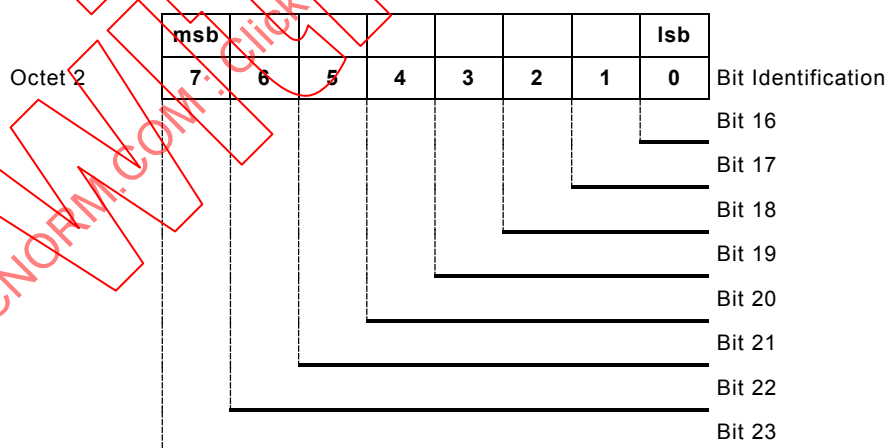


Figure 5 – Common structure of specific fields for octet 2

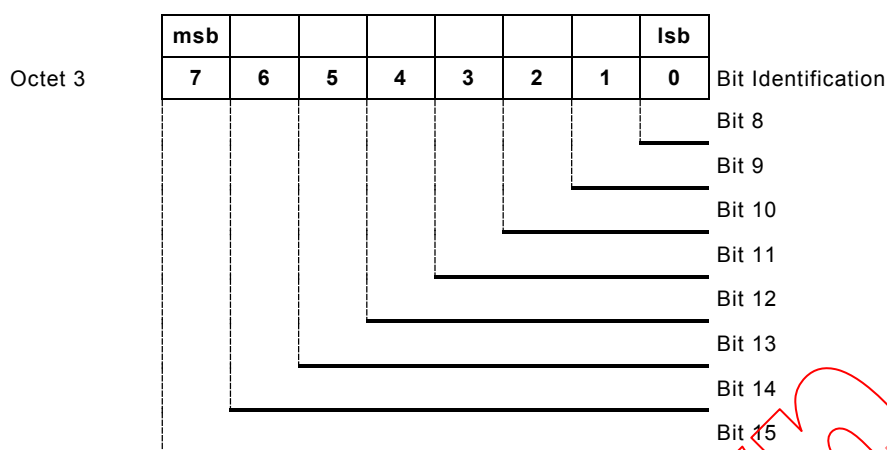


Figure 6 – Common structure of specific fields for octet 3

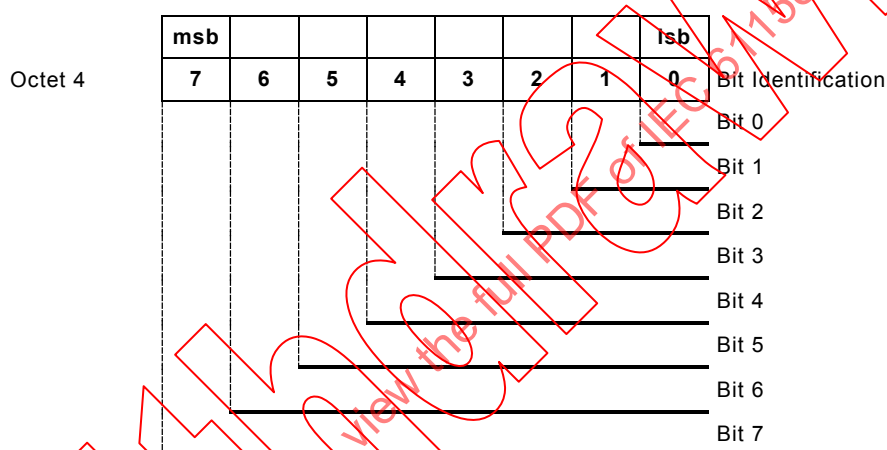


Figure 7 – Common structure of specific fields for octet 4 (low)

Bits may be grouped as group of bits. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 32 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bits.

3.4.3 Conventions used in state machines

The protocol sequences are described by means of State Machines.

In state diagrams states are represented as boxes state transitions are shown as arrows. Names of states and transitions of the state diagram correspond to the names in the textual listing of the state transitions.

The initialization of all parameters (e.g. default values) of a state machine will be realized automatically during the instantiation of the machine (new instance). Implicit transition from POWER-ON to first state calling the INIT service. If it is necessary to have different instances of one machine type, different parameters are set by the INIT service.

The textual listing of the state transitions is structured as follows, see also Table 1.

- The first row contains the name of the transition.
- The second row defines the current state.
- The third row contains an optional event followed by Conditions starting with a “/” as first line character and finally followed by the Actions starting with a “=>” as first line character.
- The last row contains the next state.

If the event occurs and the conditions are fulfilled the transition fires, e.g. the actions are executed and the next state is entered.

The layout of a Machine description is shown in Table 1. The meaning of the elements of a State Machine Description is shown in Table 2.

Table 1 – State machine description elements

#	Current state	Event or condition => action	Next state

Table 2 – Description of state machine elements

Description element	Meaning
Current state	Name of the given states.
Next state	
#	Name or number of the state transition.
Event	Name or description of the event.
/Condition	Boolean expression. The preceding “/” is not part of the condition.
=> Action	List of assignments and service or function invocations. The preceding “=>” is not part of the action.

The conventions used in the state machines are shown in Table 3.

Table 3 – Conventions used in state machines

Convention	Meaning
:=	Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.
xxx	A parameter name. Example: Identifier := reason means value of a ‘reason’ parameter is assigned to a parameter called ‘Identifier.’
“xxx”	Indicates fixed value. Example: Identifier := “abc” means value “abc” is assigned to a parameter named ‘Identifier.’
= or ==	A logical condition to indicate an item on the left is equal to an item on the right.
<	A logical condition to indicate an item on the left is less than the item on the right.
>	A logical condition to indicate an item on the left is greater than the item on the right.
<> or !=	A logical condition to indicate an item on the left is not equal to an item on the right.
>>	A semantic condition with the meaning “newer”

Convention	Meaning
<<	A semantic condition with the meaning “older”
&&	Logical “AND”
	Logical “OR”
<=	A logical condition to indicate an item on the left is equal or less than the item on the right.
>=	A logical condition to indicate an item on the left is equal or greater than the item on the right.
for (Identifier := start_value to end_value) actions endfor	This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start_value to end_value.
If (condition) actions else actions	This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition. The parts beginning with “else” can be omitted if there is no action if the condition is not fulfilled.

The conventions used for service in the state machines are shown in Table 4.

Table 4 – Conventions for services used in state machines

Convention	Meaning
xxx.req ()	request according to ISO/IEC 7498-1 invokes the service and passes any required parameter parameters are omitted and listed within the state machine description
xxx.ind ()	indication according to ISO/IEC 7498-1 advises the activation of a requested service
xxx.rsp (+), xxx.cnf (+)	response according to ISO/IEC 7498-1 may positively acknowledge or complete an action previously invoked by a request primitive confirmation according to ISO/IEC 7498-1 primitive returned to the requestor to positively acknowledge or complete an action previously invoked by a request primitive parameters are omitted and listed within the state machine description
xxx.rsp (-), xxx.cnf (-)	response according to ISO/IEC 7498-1 may negatively acknowledge or complete an action previously invoked by a request primitive confirmation according to ISO/IEC 7498-1 primitive returned to the requestor to negatively acknowledge or complete an action previously invoked by a request primitive parameters are omitted and listed within the state machine description
xxx_req (), xxx_cnf (+), xxx_cnf (-)	this construct is used for the description of local and confirmed services, which are transferred between state machines parameters are listed within the state machine description and can be delivered inside the brackets optionally
xxx_ind ()	this construct is used for the description of local services and indicates the reception of this service parameters are listed within the state machine description and can be delivered inside the brackets optionally
xxx ()	this construct is used for the description of state machine internal functions (local) parameters are listed within the state machine description and can be delivered inside the brackets optionally

Convention	Meaning
(), (+), (-)	replacement of the associated parameter list, described in the primitives definition except local services and functions can deliver parameters inside the brackets if the result is not necessary for further executions, the sign + and - can be omitted

Readers are strongly recommended to refer to the clauses for the AREP and CREP attribute definitions, the local functions and the FAL-PDU definitions to understand protocol machines. It is assumed that readers have sufficient knowledge of these definitions and they are used without further explanations.

In addition the following description elements are used:

Wildcard in names

name_XXX: "XXX" is used as wildcard string for all names beginning with "name".

others: is used as wildcard string for all events which may appear in this state excluding the explicitly stated events.

The typical use of a wildcard is an Event. In this context there are as many state transitions as possible events for this wildcard exists.

Wildcard in state names

"ANY" is used as wildcard string for a current state to indicates all states of this state table.

"SAME" is used as wildcard string for a next state to indicate that the transition remains in the same state. Only to be used in combination with "ANY".

Conditional Macro

<CONDITIONAL -MACRO-NAME>

<

CONDITION1: macrobody1

CONDITION2: macrobody2

...

>

CONDITION1, CONDITION2, etc. define all possible cases for the conditional macro. The "CONDITIONAL -MACRO-NAME" acts as place holder for the "macrocode" depending on the result of the condition.

Replacement Macro

<REPLACEMENT-MACRO-NAME>

<

XXX= Name1 : macrobody1

XXX= Name2 : macrobody2

...

>

Name1, Name2, etc. define all possible cases for the replacement macro. The "REPLACEMENT-MACRO-NAME" acts as place holder for the "macrocode" depending on the value of the current use of the wildcard XXX.

EXAMPLE

<SERVICE_REQ_PARA>

<

XXX=Read: Para1, Para2

XXX=Write: Para1, Para2, Para3

>

when called in

MSAB_XXX.req(<SERVICE_REQ_PARA>)

will result in

MSAB_Read.req(Para1, Para2) or MSAB_Write.req(Para1, Para2, Para3)

4 Application layer protocol specification for common protocols

4.1 FAL syntax description

4.1.1 DLPDU abstract syntax reference

4.1.1.1 General

Table 5, Table 6 and Table 7 give an outline of the abstract syntax of the DLPDU according to IEEE 802.3, IEEE 802.11 and IEEE 802.15.1.

4.1.1.2 IEEE 802.3

The encoding and decoding of the fields in Table 5 shall be according to IEEE 802.3 for the DLPDU.

Table 5 – IEEE 802.3 DLPDU syntax

DLPDU name	DLPDU structure
DLPDU	Preamble ^a , StartFrameDelimiter, DestinationAddress, SourceAddress, DLSDU ^b , DLPDU_Padding* ^c , FrameCheckSequence
DLSDU	[VLAN] ^d , LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^e ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU ^e ^ FRAG-PDU
VLAN	LT(=0x8100), TagControlInformation
NOTE: According to IEEE 802.3 the DLPDUs have a minimum length of 64 octets (excluded Preamble, Start Frame Delimiter).	
^a The field contains at 7 octets but should be decreased to at least 1 octets according to this standard.	
^b The minimum DLSDU size is 2 octets.	
^c The number of padding octets shall be in the range of 0..46 depending on the DLSDU size. See the minimum frame size in IEEE 802.3	
^d The VLAN field of a RTC DLPDU should be encoded by the sender, but shall be omitted for RT_CLASS_3. It may be omitted by the conveyance of the frame. The RED_RELAY may accept RT_CLASS_3 frames with VLAN. The decoder shall accept DLPDUs with or without VLAN field. Only one VLAN shall be supported, more VLANs may be supported.	
^e The VLAN field shall be omitted except for the PTCP-AnnouncePDU.	

4.1.1.3 IEEE 802.11

The encoding and decoding of the fields in Table 6 shall be according to IEEE 802.11 for the DLPDU.

Table 6 – IEEE 802.11 DLPDU syntax

DLPDU name	DLPDU structure
DLPDU	DLPDU_1 ^ DLPDU_2 ^ DLPDU_3 ^ DLPDU_4
DLPDU_1	FrameControl, DestinationAddress, SourceAddress, BSSID, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_2	FrameControl, DestinationAddress, BSSID, SourceAddress, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_3	FrameControl, BSSID, SourceAddress, DestinationAddress, SequenceControl, [QoSControl], DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_4	FrameControl, ReceiverAddress, TransmitterAddress, DestinationAddress, SequenceControl, SourceAddress, [QoSControl], DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLSDU	DSAP(=0xaa), SSAP(=0xaa), CTRL(=0x03), OUI(=0), [VLAN] ^b , LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU
NOTE 1 For definition of FrameControl see IEEE 802.11 NOTE 2 For definition of BSSID see IEEE 802.11 NOTE 3 For definition of SequenceControl see IEEE 802.11 NOTE 4 For definition of QoSControl see IEEE 802.11 NOTE 5 For definition of ReceiverAddress see IEEE 802.11 NOTE 6 For definition of TransmitterAddress see IEEE 802.11 NOTE 7 The IEEE 802.11 supports the use of VLANs in conjunction with FrameClassifier Type 2. For definition of FrameClassifier see IEEE 802.11 NOTE 8 For definition of DSAP see IEEE 802.11 NOTE 9 For definition of SSAP see IEEE 802.11 NOTE 10 For definition of CTRL see IEEE 802.11 NOTE 11 For definition of OUI in this context see IEEE 802.11	
^a The number of padding octets shall be in the range of 0...46 depending on the DLSDU size. See the minimum frame size in IEEE 802.3. ^b The VLAN field of a RTC DLPDU should be encoded by the sender, but omitted for RT_CLASS_3. It may be omitted by the conveyance of the frame, but shall be omitted by the RED_RELAY. The decoder shall accept DLPDUs with or without VLAN fields.	

4.1.1.4 IEEE 802.15.1

The encoding and decoding of the fields in Table 7 shall be according to IEEE 802.15.1 for the DLPDU.

Table 7 – IEEE 802.15.1 DLPDU syntax

DLPDU name	DLPDU structure
DLPDUHEADER	Access Code, Packet Header, Payload Header, L2CAP Header, BNEPType(0)
DLPDU	DLPDUHEADER, DestinationAddress, SourceAddress, DLSDU ^a , FrameCheckSequence
DLSDU	[VLAN] ^b , LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^c ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU ^c
VLAN	LT(=0x8100), TagControlInformation
NOTE 1 For definition of Access Code see IEEE 802.15.1 NOTE 2 For definition of Packet Header see IEEE 802.15.1 NOTE 3 For definition of Payload Header see IEEE 802.15.1 NOTE 4 For definition of L2CAP Header see IEEE 802.15.1 NOTE 5 For definition of BNEPType see IEEE 802.15.1	
^a The minimum DLSDU size is 2 octets. ^b The VLAN field of a RTC DLPDU should be encoded by the sender, but omitted for RT_CLASS_3. It may be omitted by the conveyance of the frame, but shall be omitted by the RED_RELAY. The decoder shall accept DLPDUs with or without VLAN fields. ^c The VLAN field should be omitted.	

4.1.2 Data types

4.1.2.1 Notation for the Boolean type

Boolean ::= BOOLEAN

-- TRUE if the value is non-zero.

-- FALSE if the value is zero.

4.1.2.2 Notation for the Integer type

Integer8 ::= INTEGER (-128..+127) -- range $-2^7 \leq I \leq 2^7-1$

Integer16 ::= INTEGER (-32 768..+32 767) -- range $-2^{15} \leq I \leq 2^{15}-1$

Integer32 ::= INTEGER -- range $-2^{31} \leq I \leq 2^{31}-1$

Integer64 ::= INTEGER -- range $-2^{63} \leq I \leq 2^{63}-1$

4.1.2.3 Notation for the Unsigned type

Unsigned8 ::= INTEGER (0..255) -- range $0 \leq I \leq 2^8-1$

Unsigned16 ::= INTEGER (0..65 535) -- range $0 \leq I \leq 2^{16}-1$

Unsigned32 ::= INTEGER -- range $0 \leq I \leq 2^{32}-1$

Unsigned64 ::= INTEGER -- range $0 \leq I \leq 2^{64}-1$

4.1.2.4 Notation for the Floating Point type

Float32 ::= BIT STRING SIZE (4) -- IEEE 754 Single precision

Float64 ::= BIT STRING SIZE (8) -- IEEE 754 Double precision

4.1.2.5 Notation for the OctetString type

OctetString ::= OCTET STRING -- For generic use

4.1.2.6 Notation for VisibleString type

VisibleString ::= VISIBLE STRING -- ISO/IEC 646 – International Reference Version without the “del” (coding 0x7F) character

4.1.2.7 Notation for BinaryDate type

BinaryDate ::= OCTET STRING SIZE (7) -- ISO 8601 date and time formats are recommended

4.1.2.8 Notation for TimeOfDay type

TimeOfDay with date indication ::= OCTET STRING SIZE (6) -- ISO 8601 date and time formats are recommended

TimeOfDay without date indication ::= OCTET STRING SIZE (4) -- ISO 8601 date and time formats are recommended

4.1.2.9 Notation for TimeDifference type

TimeDifference with date indication ::= OCTET STRING SIZE (6) -- ISO 8601 date and time formats are recommended

TimeDifference without date indication ::= OCTET STRING SIZE (4) -- ISO 8601 date and time formats are recommended

4.2 Transfer syntax**4.2.1 Coding of basic data types****4.2.1.1 General Encoding**

- The encoding of values shall be big endian if not explicitly otherwise stated.

4.2.1.2 Encoding of a Boolean value

- The encoding of a Boolean value shall be primitive. The ContentsOctets shall consist of a single octet.
- If the Boolean value is FALSE, the ContentsOctets shall be 0 (zero). If the Boolean value is TRUE, the ContentsOctets shall be 0xff.

4.2.1.3 Encoding of an Integer value

- The encoding of a fixed-length Integer value of Integer8, Integer16, Integer32 and Integer64 types shall be primitive and the ContentsOctets shall consist of exactly one, two, four, or eight octets, respectively.
- The ContentsOctets shall be a two's complement binary number equal to the integer value and consist of bits 7 to 0 of the first octet, followed by bits 7 to 0 of the second octet, followed by bits 7 to 0 of each octet in turn up to and including the last octet of the ContentsOctets.

NOTE The value of a two's complement binary number is derived by numbering the bits in the ContentsOctets, starting with bit 0 of the last octet as bit zero and ending the numbering with bit 7 of the first octet. Each bit is assigned a numerical value of 2^N , where N is its position in the above numbering sequence. The value of the two's complement binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one, excluding bit 7 of the first octet and then reducing this value by the numerical value assigned to bit 7 of the first octet if that bit is set to one.

4.2.1.4 Encoding of an Unsigned value

- The encoding of a fixed-length Unsigned value of Unsigned8, Unsigned16, Unsigned32 and Unsigned64 types shall be primitive and the ContentsOctets shall consist of exactly one, two, four, or eight octets, respectively.
- The ContentsOctets shall be a binary number equal to the Unsigned value and consist of bits 7 to 0 of the first octet, followed by bits 7 to 0 of the second octet, followed by bits 7 to 0 of each octet in turn up to and including the last octet of the ContentsOctets.

NOTE The value of a binary number is derived by numbering the bits in the ContentsOctets, starting with bit 0 of the last octet as bit zero and ending the numbering with bit 7 of the first octet. Each bit is assigned a numerical value of 2^N , where N is its position in the above numbering sequence. The value of the binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one.

4.2.1.5 Encoding of a Floating-Point value

- The encoding of a fixed-length floating-point value of Float32 and Float64 types shall be primitive and the ContentsOctets shall consist of exactly four or eight octets, respectively.
- The ContentsOctets shall contain floating-point values defined in conformance with IEEE 754.
For Float32 the sign is encoded in bit 7 of the first octet. It is followed by the exponent starting from bit 6 of the first octet and then the mantissa starting from bit 6 of the second octet.
For Float64 the sign is encoded in bit 7 of the first octet. It is followed by the exponent starting from bit 6 of the first octet and then the mantissa starting from bit 3 of the second octet.

4.2.1.6 Encoding of a VisibleString value

- The encoding of a variable length VisibleString value shall be primitive.
- There is no Length field; the length is encoded implicitly.
- The ContentsOctets shall be a sequence of octets. The leftmost string element is encoded in the first octet, followed by the second octet, followed by each octet in turn up to and including the last octet as rightmost of the ContentsOctets.

4.2.1.7 Encoding of an OctetString value

- The encoding of a variable length OctetString value shall be primitive.
- There is no Length field; the length is encoded implicitly.

- The ContentsOctets shall be a sequence of octets. The leftmost string element is encoded in the first octet, followed by second octet, followed by each octet in turn up to and including the last octet as rightmost of the ContentsOctets.

4.2.1.8 Encoding of a BinaryDate value

- The encoding of a BinaryDate value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the BinaryDate value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 8.

bits octets	7	6	5	4	3	2	1	0	
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	0...59 999 ms
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
3	0	0	2^5	2^4	2^3	2^2	2^1	2^0	0...59 min
4	DST	0	0	2^4	2^3	2^2	2^1	2^0	0...23 hours DST: Daylight saving time
5	day of week			day of month			2^1	2^0	1...7 day of week 1...31 day of month
6	2^2	2^1	2^0	2^4	2^3	2^2	2^1	2^0	
7	0	0	2^5	2^4	2^3	2^2	2^1	2^0	1...12 months
	0	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0 ... 50 years 2000 to 2050 51 ... 99 years 1951 to 1999
msb					lsb				

Figure 8 – Coding of the data type BinaryDate

4.2.1.9 Encoding of a TimeOfDay with and without date indication value

- The encoding of a TimeOfDay with and without date indication value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the TimeOfDay with and without date indication value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 9 and Figure 10.

bits octets	7	6	5	4	3	2	1	0	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of Milliseconds since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	number of days since 1984-01-01 T 00:00 Z only with date indication
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
msb					lsb				

Figure 9 – Encoding of TimeOfDay with date indication value

bits octets	7	6	5	4	3	2	1	0	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Number of Milliseconds since midnight
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
msb					lsb				

Figure 10 – Encoding of TimeOfDay without date indication value

4.2.1.10 Encoding of a TimeDifference with and without date indication value

- The encoding of a TimeDifference with and without date indication value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the TimeDifference with and without date indication value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 11 and Figure 12.

bits octets	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Milliseconds
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Days
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
msb					lsb				

Figure 11 – Encoding of TimeDifference with date indication value

bits octets	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Milliseconds
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
msb					lsb				

Figure 12 – Encoding of TimeDifference without date indication value

4.2.1.11 Encoding of a NULL value

- The encoding of a NULL value shall be primitive.
- The ContentsOctet shall be omitted.

4.2.1.12 Encoding of a NIL value

- The encoding of a NIL value shall be primitive.
- Each ContentsOctet shall be set to zero.

4.2.1.13 Encoding of an UUID

- The encoding of this data type shall be primitive and the ContentsOctets shall consist of exactly sixteen octets.
- For DCE RPC V1.1 the ContentsOctets shall be encoded according to Publication C706 of The Open Group.

4.2.2 Coding section related to common basic fields**4.2.2.1 Overview**

The common fields of 4.1.1 are part of the RTC-PDU and RTA-PDU.

4.2.2.2 Coding of the DLPDU field SourceAddress

This field shall be coded as data type OctetString[6]. The value of the field SourceAddress shall be according to the 48-bit universal LAN MAC addresses of IEEE 802, IEEE 802.1D and to Table 8.

Table 8 – SourceAddress

PDU	Meaning
RTC-PDU, RTA-PDU, UDP-RTC-PDU, UDP-RTA-PDU, CL-RPC-PDU, ICMP-PDU, DCP-PDU	The interface MAC address is used
LLDP-PDU, PTCP-PDU	The port MAC address is used

NOTE 1 The port MAC address is used to avoid wrong learning of the Filtering Data Base of the connected devices when ports are in the Port State BLOCKED.

NOTE 2 The port MAC address is named Bridge Port individual MAC address in IEEE 802.1D.

EXAMPLE A device with n ports needs one interface MAC address and n port MAC addresses.

4.2.2.3 Coding of the DLPDU field DestinationAddress**4.2.2.3.1 Overview**

This field shall be coded as data type OctetString[6]. The value of the field DestinationAddress shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

NOTE Octet 1 contains the Individual/Group Address Bit (Isb).

4.2.2.3.2 DCP-PDUs

For DCP-Multicast-PDUs, this field shall be coded according to Table 9, Table 10 and Table 11. DCP-Unicast-PDUs shall not use multicast or broadcast addresses.

Table 9 – DCP_MulticastMACAdd for Identify

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-00-00	With FrameID=0xFEFE used for DCP-Identify-ReqPDU

Table 10 – DCP_MulticastMACAdd for Hello

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-00-01	With FrameID=0xFEFC used for DCP-Hello-ReqPDU

Table 11 – DCP_MulticastMACAdd

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-00-02 – 00-00-FF	Reserved for other applications

4.2.2.3.3 PTCP-PDUs

For PTCP-PDUs, the value shall be set according to Table 12, Table 13, Table 14, Table 15, Table 16, Table 17, Table 18 and Table 19.

Table 12 – PTCP_MulticastMACAdd range 1

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-01-00 – 00-03-FF	Reserved for other applications

Table 13 – PTCP_MulticastMACAdd range 2

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-04-00	In conjunction with PTCP-AnnouncePDU and FrameID (=0xFF00) used for clock synchronization
01-0E-CF	00-04-01 – 00-04-1E	Reserved
01-0E-CF	00-04-1F	Reserved

Table 14 – PTCP_MulticastMACAdd range 3

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-04-20	In conjunction with PTCP-RTSyncPDU with follow up and FrameID(=0x0020) used for working clock synchronization
01-0E-CF	00-04-22 – 00-04-3E	Reserved
01-0E-CF	00-04-3F	Reserved

Table 15 – PTCP_MulticastMACAdd range 4

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-04-40	In conjunction with PTCP-FollowUpPDU and FrameID(=0xFF20) used for working clock synchronization
01-0E-CF	00-04-41 – 00-04-5E	Reserved
01-0E-CF	00-04-5F	Reserved

Table 16 – PTCP_MulticastMACAdd range 5

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-04-60 – 00-04-7F	Reserved for other applications

Table 17 – PTCP_MulticastMACAdd range 6

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-04-80	In conjunction with PTCP-RTSyncPDU and FrameID(=0x0080) used for working clock synchronization
01-0E-CF	00-04-81 – 00-04-9E	Reserved
01-0E-CF	00-04-9F	Reserved

Table 18 – PTCP_MulticastMACAdd range 7

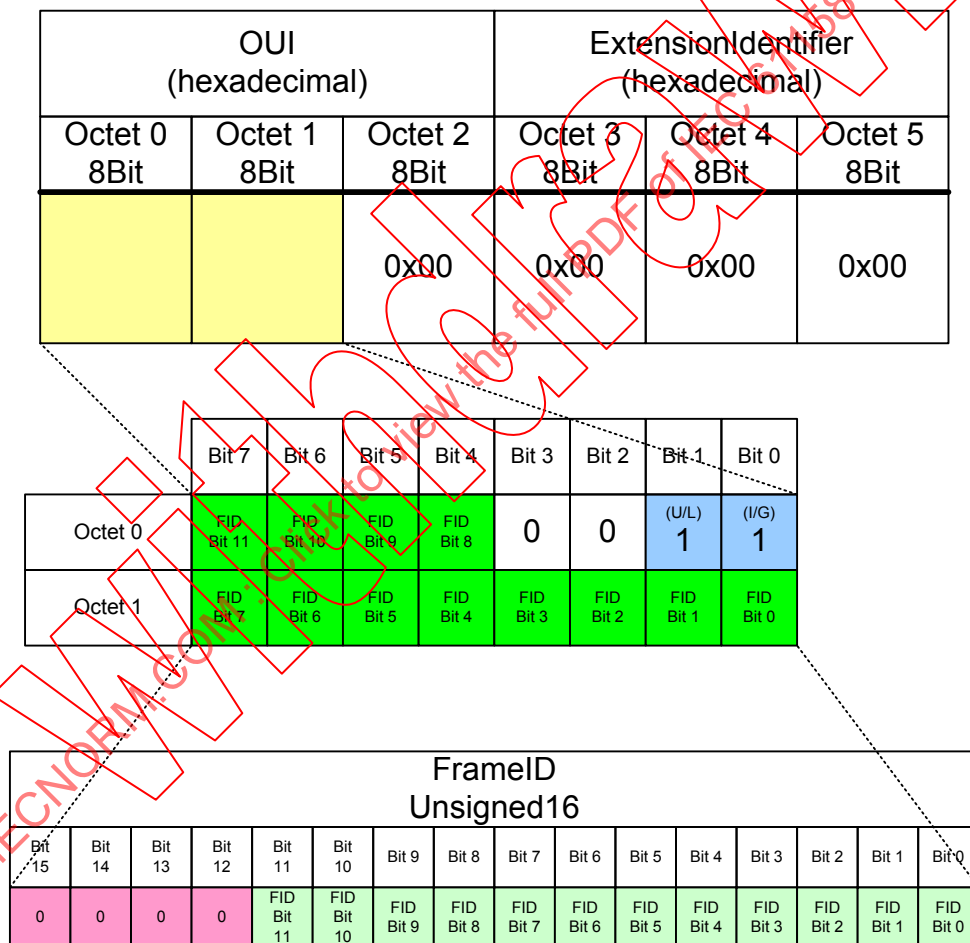
Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-A0	00-04-A0 – FF-FF-FF	Reserved for other applications

Table 19 – PTCP_MulticastMACAdd range 8

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-80-C2	00-00-0E	In conjunction with PTCP-DelayReqPDU and FrameID(=0xFF40), PTCP-DelayResPDU with follow up and FrameID(=0xFF41), PTCP-DelayFuResPDU and FrameID(=0xFF42) and PTCP-DelayResPDU without follow up and FrameID (=0xFF43) used for delay measurement

4.2.2.3.4 RTC-PDUs with FastForwarding

For RTC-PDUs with FastForwarding, this field shall be coded according to Figure 13 as a “Locally administered group address” according to the 48-bit universal LAN MAC addresses of IEEE 802.



where

(U/L) means „Universally or Locally administered address“

(I/G) means „Individual/Group address“

FID means FrameID

Figure 13 – FastForwardingMulticastMACAdd

The field DA is used for the forwarding decision and the field FrameID is used for the CPM decision.

4.2.2.3.5 RTC-PDUs with RT_CLASS_3 destination multicast address

For RTC-PDUs with RT_CLASS_3 destination multicast address, this field shall be coded according to Table 20.

Table 20 – RT_CLASS_3 destination multicast address

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-01-01	RT_CLASS_3 destination multicast address

4.2.2.3.6 RTC-PDUs with RT_CLASS_3 invalid frame multicast address

For RTC-PDUs with RT_CLASS_3 invalid frame multicast address, this field shall be coded according to Table 21.

Table 21 – RT_CLASS_3 invalid frame multicast address

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-01-02	RT_CLASS_3 invalid frame multicast address

4.2.2.4 Coding of the field LT

This field shall be coded as data type Unsigned16 with the values according to IEEE 802.3. This specification uses the values according to Table 22.

Table 22 – LT (Length/Type)

Value (hexadecimal)	Meaning
0x0800	IP (UDP, RPC, SNMP, ICMP)
0x0806	ARP
0x8100	Tag Control Information
0x8892	RTC, RTA, DCP, PTCP
0x88f7	IEEE 802.1AS
0x88CC	LLDP

4.2.2.5 Coding of the field TagControlInformation

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 11: TagControlInformation.VLAN_Id

This field shall be coded according to IEEE 802.1Q.

NOTE The VLAN_Id value zero means that no VLANs are used.

Bit 12: TagControlInformation.CanonicalFormatIdentifier

This field shall be coded according to IEEE 802.1Q.

NOTE CFI is constant 0.

Bit 13 – 15: TagControlInformation.Priority

This field shall be coded according to IEEE 802.1Q.

This field shall be coded with the values according to Table 23.

Table 23 – TagControlInformation.Priority

Value (hexadecimal)	Meaning
0x00	IP, DCP
0x01	Reserved
0x02	Reserved
0x03	Reserved
0x04	Reserved
0x05	Low prior RTA_CLASS_1 or RTA_CLASS_UDP
0x06	RT_CLASS_UDP, RT_CLASS_1, RT_CLASS_2, RT_CLASS_3 ^a high prior RTA_CLASS_1 or RTA_CLASS_UDP
0x07	PTCP-AnnouncePDU
^a RT_CLASS_3 frames uses TagControlInformation.Priority only outside of the RED period.	

NOTE If the used IP implementation does not provide Tag Control Information then the RTA_CLASS_UDP and the RT_CLASS_UDP frames may be without Tag Control Information.

4.2.2.6 Coding of the field FrameID

This field shall be coded as data type Unsigned16 with the values according to Table 24, Table 25, Table 26, Table 27, Table 28, Table 29, Table 30, Table 31, Table 32, Table 33, Table 34, Table 35 and Table 36. This field identifies the structure and the type of the APDU.

Table 24 – FrameID range 1

Value (hexadecimal)	Meaning	Use
0x0000 – 0x001F	Reserved	Reserved
0x0020	PTCP-RTSyncPDU (with follow up)	Precision transparent clock protocol synchronization with follow up Working Clock for Send Clock and Phase Synchronization
0x0021 – 0x007F	Reserved	Reserved

Table 25 – FrameID range 2

Value (hexadecimal)	Meaning	Use
0x0080	PTCP-RTSyncPDU	Precision transparent clock protocol synchronization without follow up Working Clock for Send Clock and Phase Synchronization
0x0081 – 0x009F	Reserved	—
0x00A0 – 0x00FF	Reserved	—

Table 26 – FrameID range 3

Value (hexadecimal)	Meaning	Use
0x0100 – 0x06FF	Dedicated to communication class RT_CLASS_3 (RED) unicast and multicast	RED, non redundant, normal or DFP ^a
0x0700 – 0x0FFF	Dedicated to communication class RT_CLASS_3 (RED) unicast and multicast	RED, redundant ^b , normal or DFP
0x1000 – 0x47FF	Reserved	—

^a A network monitor may sort “normal” and “DFP” frame by using the first SFCRC16 of the frame.

^b In this case two FrameIDs (e.g. 0xXXX0 and 0xXXX1) are used for one CR. One for each direction in which the frame travels the ring. The pair shall be identified ignoring the least significant bit of the FrameID.

Table 27 – FrameID range 4

Value (hexadecimal)	Meaning	Use
0x4800 – 0x4FFF	Reserved	—
0x5000 – 0x57FF	Reserved	—
0x5800 – 0x5FFF	Reserved	—
0x6000 – 0x67FF	Reserved	—

Table 28 – FrameID range 5

Value (hexadecimal)	Meaning	Use
0x6800 – 0x6FFF	Reserved	—
0x7000 – 0x77FF	Reserved	—
0x7800 – 0x7FFF	Reserved	—

Table 29 – FrameID range 6

Value (hexadecimal)	Meaning	Use
0x8000 – 0xBBFF	Dedicated to communication class RT_CLASS_1 (GREEN) unicast	GREEN, non redundant, normal ^a ^b
0xBC00 – 0xBFFF	Dedicated to RT_CLASS_1 (GREEN) multicast	GREEN, non redundant, normal ^a ^b

^a Used as substitution for the former communication class RT_CLASS_1(legacy) shown in Table 30.

^b May be used as RT_CLASS_2 (ORANGE) only in conjunction with ARProperties.StartupMode == Legacy.

Table 30 – FrameID range 7

Value (hexadecimal)	Meaning	Use
0xC000 – 0xF7FF	Dedicated to communication class RT_CLASS_UDP unicast	RT_CLASS_UDP (recommended) and RT_CLASS_1(legacy) ^a ; The concurrent use of this range for RT_CLASS_UDP and RT_CLASS_1 shall be rejected.

Value (hexadecimal)	Meaning	Use
0xF800 – 0xFBFF	Dedicated to communication class RT_CLASS_UDP multicast	RT_CLASS_UDP (recommended) and RT_CLASS_1(legacy) ^a ; The concurrent use of this range for RT_CLASS_UDP and RT_CLASS_1 shall be rejected.
^a RT_CLASS_1(legacy) shall be supported by IO controller and IO supervisor. It should not be generated by an IO device.		

Table 31 – FrameID range 8

Value (hexadecimal)	Meaning	Use
0xFC00	Reserved	—
0xFC01	Alarm High	RTA_CLASS_1 and RTA_CLASS_UDP
0xFC02 – 0xFDFD	Reserved	—
0xFE00	Reserved	—
0xFE01	Alarm Low	RTA_CLASS_1 and RTA_CLASS_UDP
0xFE02 – 0xFEFB	Reserved	—
0xFEFC	DCP-Hello-ReqPDU	DCP
0xFEFD	DCP-Get-ReqPDU, DCP-Get-ResPDU, DCP-Set-ReqPDU, DCP-Set-ResPDU	DCP
0xFEFE	DCP-Identify-ReqPDU	DCP
0xFEFF	DCP-Identify-ResPDU	DCP

Table 32 – FrameID range 9

Value (hexadecimal)	Meaning	Use
0xFF00	PTCP-AnnouncePDU (working clock)	Precision transparent clock announce protocol Isochronous application, send clock and phase synchronization
0xFF01 – 0xFF1F	Reserved	—
0xFF20	PTCP-FollowUpPDU (working clock)	Send Clock and phase synchronization (PTCP- FollowUpPDU)
0xFF21 – 0xFF3F	Reserved	—
0xFF40	PTCP-DelayReqPDU	For delay measurement
0xFF41	PTCP-DelayResPDU	For delay measurement with follow up
0xFF42	PTCP-DelayFuResPDU	For delay measurement with follow up
0xFF43	PTCP-DelayResPDU	For delay measurement without follow up
0xFF44 – 0xFF5F	Reserved	—

Table 33 – FrameID range 10

Value (hexadecimal)	Meaning	Use
0xFF60 – 0xFF6F	Reserved	—

Table 34 – FrameID range 11

Value (hexadecimal)	Meaning	Use
0xFF70 – 0xFF7F	Reserved	—

Table 35 – FrameID range 12

Value (hexadecimal)	Meaning	Use
0xFF80 – 0xFF8F	FragmentationFrameID See 4.2.2.7	Peer to peer protocol to enable SendClockFactors less than 5 independent of the frame size of the concurrent used protocols.

NOTE The principle of fragmentation is usable with all SendClockFactors.

Table 36 – FrameID range 13

Value (hexadecimal)	Meaning	Use
0xFF90 – 0xFFFF	Reserved	—

4.2.2.7 Coding of the field FragmentationFrameID

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 3: FragmentationFrameID.FragSequence

This field shall be used to identify the fragmented frame by the receiver and shall be coded according to Table 37.

Table 37 – FragmentationFrameID.FragSequence

Value (hexadecimal)	Meaning
0x00 – 0x0F	Frame number incremented by the sender for each frame which is fragmented.

Bit 4 – 15: FragmentationFrameID.Constant

This field shall be coded with the values according to Table 38.

Table 38 – FragmentationFrameID.Constant

Value (hexadecimal)	Meaning
0xFF8	Upper part of the FrameID according to Table 35

4.3 Discovery and basic configuration

4.3.1 DCP syntax description

4.3.1.1 DLDPDU abstract syntax reference

The DLDPDU abstract syntax from 4.1.1 shall be applied.

4.3.1.2 DCP APDU abstract syntax

Table 39 defines the abstract syntax of the DCP-PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs.

Table 39 – DCP APDU syntax

APDU name	APDU structure
DCP-PDU	DCP-Multicast-PDU ^ DCP-Unicast-PDU
DCP-Multicast-PDU	DCP-Identify-ReqPDU ^g ^ DCP-Hello-ReqPDU
DCP-Unicast-PDU	DCP-Get-ReqPDU ^ DCP-Set-ReqPDU ^ DCP-Get-ResPDU ^ DCP-Set-ResPDU ^ DCP-Identify-ResPDU
DCP-Identify-ReqPDU	DCP-IdentifyFilter-ReqPDU ^ DCP-IdentifyAll-ReqPDU
DCP-IdentifyFilter-ReqPDU	DCP-MC-Header, [NameOfStationBlock] ^ [AliasNameBlock], IdentifyReqBlock* ^a
DCP-IdentifyAll-ReqPDU	DCP-MC-Header, AllSelectorBlock
DCP-Hello-ReqPDU	DCP-UC-Header, NameOfStationBlockRes, { IPPParameterBlockRes, DeviceIDBlockRes, [DeviceOptionsBlockRes] ^e , DeviceRoleBlockRes, DeviceInitiativeBlockRes, [DeviceInstanceBlockRes] } ^f
DCP-Identify-ResPDU	DCP-UC-Header, { IdentifyResBlock* ^b , NameOfStationBlockRes ^c , IPPParameterBlockRes ^c , DeviceIDBlockRes ^c , DeviceVendorBlockRes ^c , DeviceOptionsBlockRes ^c , DeviceRoleBlockRes ^c , [DeviceInitiativeBlockRes] ^d , [DeviceInstanceBlockRes] } ^f
DCP-Get-ReqPDU	DCP-UC-Header, GetReqBlock*
DCP-Get-ResPDU	DCP-UC-Header, (GetResBlock ^ GetNegResBlock)*
DCP-Set-ReqPDU	DCP-UC-Header, [StartTransactionBlock, BlockQualifier], [FactoryResetBlock, BlockQualifier] ^ [ResetToFactoryBlock, BlockQualifier] ^ SetReqBlock*, [StopTransactionBlock, BlockQualifier]
DCP-Set-ResPDU	DCP-UC-Header, (SetResBlock ^ SetNegResBlock)*
^a The content of the field value of the IdentifyReqBlock of the PDU shall be interpreted as a filter at the receiving instance. All included DataBlocks shall be operated with a logical AND and it shall only responded with a DCP-Identify-ResPDU if all filter criteria match. ^b The content of the field value of the IdentifyResBlock of the PDU shall contain the option and suboption requested in the IdentifyReqBlock. The IdentifyResBlock shall only contain the AliasNameBlock used in the IdentifyReqBlock. ^c This field shall only be present if it is not already part of the IdentifyResBlock. ^d This field shall only be present if the usage of DCP-Hello-ReqPDU is activated. ^e This field should be omitted. ^f Additional blocks may be appended by the sender. ^g May be used in conjunction with interface MAC address.	

Table 40 defines structures for substitutions of elements of the APDU structures shown in Table 39.

Table 40 – DCP substitutions

Substitution name	Structure
DCP-MC-Header	ServiceID, ServiceType, Xid, ResponseDelayFactor, DCPDataLength
DCP-UC-Header	ServiceID, ServiceType, Xid, Padding* ^a , DCPDataLength ^a Number of Padding octets shall be 2.
IdentifyReqBlock	DeviceRoleBlock ^ DeviceVendorBlock ^ DeviceIDBlock ^ DeviceOptionsBlock ^ MACAddressBlock ^ IPPParameterBlock ^ DHCPParameterBlock ^ ManufacturerSpecificParameterBlock

Substitution name	Structure
IdentifyResBlock	NameOfStationBlockRes ^ DeviceRoleBlockRes ^ DeviceVendorBlockRes ^ DeviceIDBlockRes ^ DeviceOptionsBlockRes ^ MACAddressBlockRes ^ IPParameterBlockRes ^ DHCPParameterBlockRes ^ ManufacturerSpecificParameterBlockRes ^ AliasNameBlockRes
GetReqBlock	NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ MACAddressType ^ IPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ DeviceInstanceType ^ AllSelectorType
GetResBlock	NameOfStationBlockRes ^ DeviceRoleBlockRes ^ DeviceVendorBlockRes ^ DeviceIDBlockRes ^ DeviceOptionsBlockRes ^ MACAddressBlockRes ^ IPParameterBlockRes ^ FullIPSuiteBlockRes ^ DHCPParameterBlockRes ^ ManufacturerSpecificParameterBlockRes ^ DeviceInstanceBlockRes
GetNegResBlock	ControlOption, SuboptionResponse, DCPBlockLength, NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ MACAddressType ^ IPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ DeviceInstanceType, BlockError, [AddDataValue]
SetReqBlock	SignalType ^ NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ IPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType, DCPBlockLength, BlockQualifier, SignalValue ^ NameOfStationValue ^ DeviceRoleValue ^ DeviceVendorValue ^ DeviceIDValue ^ DeviceOptionsValue ^ IPParameterValue ^ FullIPSuiteValue ^ DHCPParameterValue ^ ManufacturerSpecificParameterValue Only the adjunctive blocks shall be combined to a SetReqBlock Example: SignalType, DCPBlockLength, BlockQualifier, SignalValue
SetResBlock	ControlOption, SuboptionResponse, DCPBlockLength, NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ IPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ SignalType ^ FactoryResetType ^ ResetToFactoryType ^ StartTransactionType ^ StopTransactionType, BlockError(=0), [AddDataValue]
SetNegResBlock	ControlOption, SuboptionResponse, DCPBlockLength, NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ IPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ SignalType ^ FactoryResetType ^ ResetToFactoryType ^ StartTransactionType ^ StopTransactionType, BlockError(<>0), [AddDataValue]
AllSelectorType	AllSelectorOption, SuboptionAllSelector
AllSelectorBlock	AllSelectorType, DCPBlockLength
DeviceInitiativeType	DeviceInitiativeOption, DeviceInitiativeSuboption Only used for the DCP-Hello-ReqPDU and DCP-Identify-ResPDU
DeviceInitiativeBlockRes	DeviceInitiativeType, DCPBlockLength, BlockInfo, DeviceInitiativeValue
StartTransactionType	ControlOption, SuboptionStart
StartTransactionBlock	StartTransactionType, DCPBlockLength
StopTransactionType	ControlOption, SuboptionStop
StopTransactionBlock	StopTransactionType, DCPBlockLength
SignalType	ControlOption, SuboptionSignal

Substitution name	Structure
FactoryResetType	ControlOption, SuboptionFactoryReset
FactoryResetBlock	FactoryResetType, DCPBlockLength
ResetToFactoryType	ControlOption, SuboptionResetToFactory
ResetToFactoryBlock	ResetToFactoryType, DCPBlockLength
NameOfStationType	DevicePropertiesOption, SuboptionNameOfStation
NameOfStationBlock	NameOfStationType, DCPBlockLength, NameOfStationValue
NameOfStationBlockRes	NameOfStationType, DCPBlockLength, BlockInfo, NameOfStationValue
AliasNameType	DevicePropertiesOption, SuboptionAliasName
AliasNameBlock	AliasNameType, DCPBlockLength, AliasNameValue
AliasNameBlockRes	AliasNameType, DCPBlockLength, BlockInfo, AliasNameValue
DeviceRoleType	DevicePropertiesOption, SuboptionDeviceRole
DeviceRoleBlock	DeviceRoleType, DCPBlockLength, DeviceRoleValue
DeviceRoleBlockRes	DeviceRoleType, DCPBlockLength, BlockInfo, DeviceRoleValue
DeviceRoleValue	DeviceRoleDetails, Padding
DeviceVendorType	DevicePropertiesOption, SuboptionDeviceVendor
DeviceVendorBlock	DeviceVendorType, DCPBlockLength, DeviceVendorValue
DeviceVendorBlockRes	DeviceVendorType, DCPBlockLength, BlockInfo, DeviceVendorValue
DeviceIDType	DevicePropertiesOption, SuboptionDeviceID
DeviceIDBlock	DeviceIDType, DCPBlockLength, DeviceIDValue
DeviceIDBlockRes	DeviceIDType, DCPBlockLength, BlockInfo, DeviceIDValue
DeviceIDValue	VendorIDHigh, VendorIDLow, DeviceIDHigh, DeviceIDLow
DeviceOptionsType	DevicePropertiesOption, SuboptionDeviceOptions
DeviceOptionsBlock	DeviceOptionsType, DCPBlocklength, DeviceOptionsValue
DeviceOptionsBlockRes	DeviceOptionsType, DCPBlocklength, BlockInfo, DeviceOptionsValue
DeviceOptionsValue	(Option ^a , Suboption ^b)* ^c ^a See Table 49 for the values ^b See Table 50, Table 51, Table 52, Table 53, Table 54, Table 55, Table 56, and Table 57 for the values ^c If no entry exists, then omit the DeviceOptionsBlockRes in the DCP-Identify-Res-PDU
DeviceInstanceType	DevicePropertiesOption, SuboptionDeviceInstance Not usable in conjunction with DCP-Set-ReqPDU
DeviceInstanceBlockRes	DeviceInstanceType, DCPBlockLength, BlockInfo, DeviceInstanceValue
DeviceInstanceValue	InstanceHigh, InstanceLow
MACAddressType	IPOption, SuboptionMACAddress
MACAddressBlock	MACAddressType, DCPBlockLength, MACAddressValue
MACAddressBlockRes	MACAddressType, DCPBlockLength, BlockInfo, MACAddressValue
IPParameterType	IPOption, SuboptionIPParameter
IPParameterBlock	IPParameterType, DCPBlockLength, IPParameterValue
IPParameterBlockRes	IPParameterType, DCPBlockLength, BlockInfo, IPParameterValue
IPParameterValue	IPAddress, Subnetmask, StandardGateway
FullIPSuiteType	IPOption, SuboptionFullIPSuite

Substitution name	Structure
FullIPSuiteBlockRes	FullIPSuiteType, DCPBlockLength, BlockInfo, FullIPSuiteValue
FullIPSuiteValue	IPAddress, Subnetmask, StandardGateway, IPAddress[4] ^a ^a This array contains up to four DNS server IP addresses. The value "0.0.0.0" indicates "no address".
DHCPPParameterType	DHCPOption, SuboptionDHCP
DHCPPParameterBlock	DHCPPParameterType, DCPBlockLength, DHCPPParameterValue
DHCPPParameterBlockRes	DHCPPParameterType, DCPBlockLength, BlockInfo, DHCPPParameterValue
DHCPPParameterValue	SuboptionDHCP, DHCPPParameterLength, DHCPPParameterData
ManufacturerSpecificParameterType	ManufacturerSpecificOption, SuboptionManufacturerSpecific
ManufacturerSpecificParameterBlock	ManufacturerSpecificParameterType, DCPBlockLength, ManufacturerSpecificParameterValue
ManufacturerSpecificParameterBlock Res	ManufacturerSpecificParameterType, DCPBlockLength, BlockInfo, ManufacturerSpecificParameterValue
ManufacturerSpecificParameterValue	ManufacturerOUI, ManufacturerSpecificString

4.3.1.3 Coding section related to header fields

4.3.1.3.1 Coding of the field ServiceID

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 41. They shall be set in the appropriate PDU.

Table 41 – ServiceID

Value (hexadecimal)	Meaning
0x00 – 0x02	Reserved
0x03	Get
0x04	Set
0x05	Identify
0x06	Hello
0x07 – 0xFF	Reserved

4.3.1.3.2 Coding of the field ServiceType

4.3.1.3.2.1 Coding of the field ServiceType for request

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0: ServiceType.Selection

This field shall be coded with the values according to Table 42.

Table 42 – ServiceType.Selection

Value (hexadecimal)	Meaning
0x00	Request
0x01	Reserved

Bit 1 – 7: ServiceType.Reserved

This field shall be coded with the values according to Table 43.

Table 43 – ServiceType.Reserved

Value (hexadecimal)	Meaning
0x00	Default
0x01 – 0x7F	Reserved

4.3.1.3.2.2 Coding of the field ServiceType for response

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0: ServiceType.Selection

This field shall be coded with the values according to Table 44.

Table 44 – ServiceType.Selection

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Response

Bit 1: ServiceType.Reserved_1

This field shall be coded with the values according to Table 45.

Table 45 – ServiceType.Reserved_1

Value (hexadecimal)	Meaning
0x00	Default
0x01	Reserved

Bit 2: ServiceType.Response

This field shall be coded with the values according to Table 46.

Table 46 – ServiceType.Response

Value (hexadecimal)	Meaning
0x00	Success Default
0x01	ServiceID not supported Optional, otherwise the request may be ignored.

Bit 3 – 7: ServiceType.Reserved_2

This field shall be coded with the values according to Table 47.

Table 47 – ServiceType.Reserved_2

Value (hexadecimal)	Meaning
0x00	Default
0x01 – 0x1F	Reserved

4.3.1.3.3 Coding of the field Xid

This field shall be coded as data type Unsigned32. It shall contain a transaction identification chosen by the client to associate requests and responses between a client and a server.

4.3.1.3.4 Coding of the field DCPDataLength

This field shall be coded as data type Unsigned16. It shall contain the total length of data followed the DCP-UC-Header or DCP-MC-Header in octets. The maximum length of DCP Data is 1 432 octets.

4.3.1.3.5 Coding of the field ResponseDelayFactor

This field shall be coded as data type Unsigned16 according to Table 48. It shall contain a delay factor that shall be used by the server to calculate its individual response delay time.

Table 48 – ResponseDelayFactor

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001	Allowed value without spread
0x0002 – 0x1900	Allowed value with spread
0x1901 – 0xFFFF	Reserved

The server shall calculate the response delay time (allowed values between 0 ms and 64 s) according to the following formulae.

The last two octets of the MAC address (see the 48-bit universal LAN MAC addresses of IEEE 802) shall be used as random number K. Octet 6 of the MAC address represents the low-order octet and octet 5 the high-order octet.

$$\text{Spread} = K \text{ MOD } \text{ResponseDelayFactor} \quad (1)$$

where

Spread	is the specific spreading factor
K	is the random number, last two octets of the MAC address as random number
ResponseDelayFactor	is the factor for the calculation of an individual response delay

Spread not equal zero:

$$\text{Minimal Response Delay} = 10 \text{ ms} \times \text{Spread} \quad (2)$$

where

Minimal Response Delay	is the minimal response delay

Spread is the specific spreading factor

Spread equal zero:

$$\text{Minimal Response Delay} = 0 \text{ ms} \quad (3)$$

where

Minimal Response Delay is the minimal response delay

The client shall calculate the response delay timeout according to the following formulae.

ResponseDelayFactor equal one:

$$\text{Response delay timeout} = 400 \text{ ms} \quad (4)$$

where

Response delay timeout is the response delay timeout

ResponseDelayFactor greater than one:

$$\text{Intermediate value} = 1 \text{ s} + \text{ResponseDelayFactor} \times 10 \text{ ms} \quad (5)$$

where

Intermediate value is the intermediate value of the response delay

ResponseDelayFactor is the factor for the calculation of an individual response delay

$$\text{Response delay timeout} = \text{Round up to next second (Intermediate value)} \quad (6)$$

where

Response delay timeout is the response delay timeout

Intermediate value is the intermediate value of the response delay

4.3.1.4 Coding section of block fields

4.3.1.4.1 General

The block fields are parted into option, suboption, block length, block info and value. Every block shall assure Unsigned16 alignment. The added padding octets, with the value zero, shall be counted for the DCPDataLength and shall not be counted for the DCPBlockLength. Table 49 shows the list of available options and Table 50, Table 51, Table 52, Table 53, Table 54, Table 55 and Table 56 shows the list of available suboptions.

Table 49 – List of options

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	IPOption
0x02	DevicePropertiesOption
0x03	DHCPOption

Value (hexadecimal)	Meaning
0x04	Reserved
0x05	ControlOption
0x06	DeviceInitiativeOption
0x07 – 0x7F	Reserved
0x80 – 0xFE	ManufacturerSpecificOption
0xFF	AllSelectorOption

Table 50 – List of suboptions for option IOption

Value (hexadecimal)	Meaning	Accessible
0x01	SuboptionMACAddress	Read
0x02	SuboptionIPParameter	Read / Write
0x03	SuboptionFullIPSuite	Read / Write Optional
Other	Reserved	—

Table 51 – List of suboptions for option DevicePropertiesOption

Value (hexadecimal)	Meaning	Accessible
0x01	SuboptionDeviceVendor	Read
0x02	SuboptionNameOfStation	Read / Write
0x03	SuboptionDeviceID	Read
0x04	SuboptionDeviceRole	Read
0x05	SuboptionDeviceOptions	Read
0x06	SuboptionAliasName	Used as filter only
0x07	SuboptionDeviceInstance	Read Optional
Other	Reserved	—

Table 52 – List of suboptions for option DHCPOption

Value (hexadecimal)	Meaning	Accessible
See Table 57	SuboptionDHCP	Read / Write

Table 53 – List of suboptions for option ControlOption

Value (hexadecimal)	Meaning	Accessible
0x01	SuboptionStart	Write
0x02	SuboptionStop	Write
0x03	SuboptionSignal	Write
0x04	SuboptionResponse	—

Value (hexadecimal)	Meaning	Accessible
0x05	SuboptionFactoryReset	Write
0x06	SuboptionResetToFactory	Write
Other	Reserved	—

Table 54 – List of suboptions for option DeviceInitiativeOption

Value (hexadecimal)	Meaning	Accessible
0x01	SuboptionDeviceInitiative	Read
Other	Reserved	—

Table 55 – List of suboptions for option AllSelectorOption

Value (hexadecimal)	Meaning	Accessible
0xFF	SuboptionAllSelector	Used as filter only
Other	Reserved	—

Table 56 – List of suboptions for option ManufacturerSpecificOption

Value (hexadecimal)	Meaning	Accessible
0x00 – 0xFF	SuboptionManufacturerSpecific	Manufacturer specific

4.3.1.4.2 Coding section related to IPOption**4.3.1.4.2.1 Coding of the field IPOption**

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.2.2 Coding of the field SuboptionMACAddress

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.2.3 Coding of the field SuboptionIPParameter

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.3 Coding section related to DevicePropertiesOption**4.3.1.4.3.1 Coding of the field DevicePropertiesOption**

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.3.2 Coding of the field SuboptionDeviceVendor

This field shall be coded as data type Unsigned8 according to Table 51.

4.3.1.4.3.3 Coding of the field SuboptionNameOfStation

This field shall be coded as data type Unsigned8 according to Table 51.

4.3.1.4.3.4 Coding of the field SuboptionDeviceID

This field shall be coded as data type Unsigned8 according to Table 51.

4.3.1.4.3.5 Coding of the field SuboptionDeviceRole

This field shall be coded as data type Unsigned8 according to Table 51.

4.3.1.4.3.6 Coding of the field SuboptionDeviceOptions

This field shall be coded as data type Unsigned8 according to Table 51.

This option/suboption combination should be used to convey a list of the supported options and suboptions which are defined as optional and could be used for all options and suboptions.

4.3.1.4.3.7 Coding of the field SuboptionAliasName

This field shall be coded as data type Unsigned8 according to Table 51.

4.3.1.4.3.8 Coding of the field SuboptionDeviceInstance

This field shall be coded as data type Unsigned8 according to Table 51.

4.3.1.4.4 Coding section related to DHCPOption**4.3.1.4.4.1 Coding of the field DHCPOption**

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.4.2 Coding of the field SuboptionDHCP

This field shall be coded as data type Unsigned8. The allowed values shall be according to Table 57.

Table 57 – SuboptionDHCP

Value (decimal)	Data Length	Description	References
0 – 11	—	Reserved ^a	—
12	1 +	Host Name ^d	RFC 2132
13 – 42	—	Reserved ^a	—
43	1 +	Vendor specific information	RFC 2132
44 – 53	—	Reserved ^a	—
54	4	Server identifier	RFC 2132
55	1 +	Parameter request list ^c	RFC 2132
56 – 59	—	Reserved ^a	—
60	1 +	Class-identifier	RFC 2132
61	2 +	DHCP client identifier	RFC 2132, RFC 4361, and RFC 4363
62 – 80	—	Reserved ^a	—
81	1 +	FQDN, Fully Qualified Domain Name ^d	—
82 – 96	—	Reserved ^a	—

Value (decimal)	Data Length	Description	References
97	Variable	UUID-based Client (=addressed Station) Identifier	—
98 – 254	—	Reserved ^a	—
255 ^b	1	See 4.3.1.4.24	—
^a The reserved options should be used in the same meaning as they are defined in the corresponding RFCs for DHCP. ^b By RFC 2132 defined as “END” but handled internally. Thus this key is overloaded as DHCP control in this standard. ^c In this option, at least the parameters 1 (subnet mask) and 3 (router) should be requested. ^d The use of these suboptions is still not fixed and subject to change in later revisions.			

4.3.1.4.5 Coding section related to ControlOption

4.3.1.4.5.1 Coding of the field ControlOption

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.5.2 Coding of the field SuboptionStart

This field shall be coded as data type Unsigned8 according to Table 53 and with a DCPBlockLength according to Table 58.

Table 58 – Coding of DCPBlockLength in conjunction with SuboptionStart

Value (hexadecimal)	Meaning
0x0000 – 0x0001	Reserved
0x0002	Default
0x0003 – 0xFFFF	Reserved

There are no additional data for this suboption.

4.3.1.4.5.3 Coding of the field SuboptionStop

This field shall be coded as data type Unsigned8 according to Table 53 and with a DCPBlockLength according to Table 59.

Table 59 – Coding of DCPBlockLength in conjunction with SuboptionStop

Value (hexadecimal)	Meaning
0x0000 – 0x0001	Reserved
0x0002	Default
0x0003 – 0xFFFF	Reserved

There are no additional data for this suboption.

4.3.1.4.5.4 Coding of the field SuboptionSignal

This field shall be coded as data type Unsigned8 according to Table 53 and with a DCPBlockLength according to Table 60.

Table 60 – Coding of DCPBlockLength in conjunction with SuboptionSignal

Value (hexadecimal)	Meaning
0x0000 – 0x0003	Reserved
0x0004	Default
0x0005 – 0xFFFF	Reserved

4.3.1.4.5.5 Coding of the field SuboptionResponse

This field shall be coded as data type Unsigned8 according to Table 53.

4.3.1.4.5.6 Coding of the field SuboptionFactoryReset

This field shall be coded as data type Unsigned8 according to Table 53 and with a DCPBlockLength according to Table 61.

Table 61 – Coding of DCPBlockLength in conjunction with SuboptionFactoryReset

Value (hexadecimal)	Meaning
0x0000 – 0x0001	Reserved
0x0002	Default
0x0003 – 0xFFFF	Reserved

There are no additional data for this suboption.

Factory Reset shall permanently set

- The NameOfStation to “”
- The IP address, the subnet mask and the standard gateway to 0.0.0.0
- All other parameters to the manufacturers default value.

NOTE All stored parameters of the physical device like, PDIRData, PDPortDataAdjust, ... are set to default as manufacturers default value.

Every device should support a possibility to execute “reset to factory” without an engineering device. This could be done with a switch or by other means.

The execution of Factory Reset should not influence the switching of a node.

4.3.1.4.5.7 Coding of the field SuboptionResetToFactory

This field shall be coded as data type Unsigned8 according to Table 53 and with a DCPBlockLength according to Table 62.

Table 62 – Coding of DCPBlockLength in conjunction with SuboptionResetToFactory

Value (hexadecimal)	Meaning
0x0000 – 0x0001	Reserved
0x0002	Default
0x0003 – 0xFFFF	Reserved

There are no additional data for this suboption.

This Suboption offers different modes coded by the BlockQualifier.

4.3.1.4.6 Coding section related to DeviceInitiativeOption

4.3.1.4.6.1 Coding of the field DeviceInitiativeOption

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.6.2 Coding of the field SuboptionDeviceInitiative

This field shall be coded as data type Unsigned8 according to Table 54 and with a DCPBlockLength according to Table 63.

Table 63 – Coding of DCPBlockLength in conjunction with SuboptionDeviceInitiative

Value (hexadecimal)	Meaning
0x0000 – 0x0003	Reserved
0x0004	Default
0x0005 – 0xFFFF	Reserved

4.3.1.4.7 Coding section related to AllSelectorOption

4.3.1.4.7.1 Coding of the field AllSelectorOption

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.7.2 Coding of the field SuboptionAllSelector

This field shall be coded as data type Unsigned8 according to Table 55.

4.3.1.4.8 Coding section related to ManufacturerSpecificOption

4.3.1.4.8.1 Coding of the field ManufacturerSpecificOption

This field shall be coded as data type Unsigned8 according to Table 49.

4.3.1.4.8.2 Coding of the field SuboptionManufacturerSpecific

This field shall be coded as data type Unsigned8 according to Table 56.

4.3.1.4.9 Coding of the field DCPBlockLength

This field shall be coded as data type Unsigned16 and shall contain the length of the specific data of the suboption without DCPBlockLength itself and without counting padding octets.

4.3.1.4.10 Coding of the field BlockQualifier

This field shall be coded as data type Unsigned16.

Allowed values with the option IPOption, DevicePropertiesOption, DHCPOption and ManufacturerSpecificOption shall be according to Table 64.

Table 64 – BlockQualifier with options IPOption, DevicePropertiesOption, DHCPOption and ManufacturerSpecificOption

Bit	Value (binary)	Meaning
Bit 0	0	Store the value of the option/suboption and its attribute "temporary" in the DCP ASE and use the value subsequently. The attribute temporary means that the value shall be reset to its default value after a power off / power on cycle.
	1	Store the value of the option/suboption and its attribute "permanent" in the DCP ASE and use the value subsequently. The attribute permanent means that the value shall be available after a power off / power on cycle.
Bit 1 – 15	—	Reserved

Allowed values with the option ControlOption and suboption SuboptionResetToFactory shall be according to Table 65.

Table 65 – BlockQualifier with option ControlOption and suboption SuboptionResetToFactory

Bit	Value (decimal)	Meaning	Range
Bit 0	0	Reserved	—
	1		
Bit 1 – 15	0	Reserved	Interface
	1	Recommended Reset application data Reset data which have been stored permanent in submodules and modules to factory values.	
	2	Mandatory Reset communication parameter All parameters active for the interface or the ports and the ARs shall be set to the default values and if permanently stored reset. This mode is intended to set the addressed communication interface of a device into a state which almost similar to the "out of the box" state.	
	3	Recommended Reset engineering parameter Reset engineering parameters which have been stored permanent in the IOC or IOD to factory values.	
	4	Recommended Resets all stored data Resets all stored data in the IOD or IOC to its factory default values.	

Bit	Value (decimal)	Meaning	Range
	5 – 7	Reserved	Device
	8	Recommended Resets all stored data in the IOD or IOC to its factory values. This service shall reset the communication parameters of all interfaces of the device and should reset all parameters of the device. This mode is intended to set the device into a state which is similar to the “out of the box” state.	
	9	Optional Reset and restore data Reset installed software revisions to factory images.	
	10 – 15	Reserved	
	other	Reserved	—
The applicative behavior shall be defined by profiles and/or the local application.			

In all other cases the field BlockQualifier shall be set according to Table 66.

Table 66 – BlockQualifier with all other options and suboptions

Bit	Value (binary)	Meaning
Bit 0 – 15	0	Default
Bit 0 – 15	other	Reserved

4.3.1.4.11 Coding of the field BlockError

This field shall be coded as data type Unsigned8 and shall be set according to Table 67.

Table 67 – BlockError

Value (hexadecimal)	Meaning	Usage
0x00	No error	Positive response Parameter delivered and accepted
0x01	Option not supported	If optional option is not supported
0x02	Suboption not supported or no DataSet available or BlockQualifier not supported	If optional suboption is not supported
0x03	Suboption not set	If suboption could not be set for local reasons
0x04	Resource error	If there is a temporary resource error within the server
0x05	SET not possible by local reasons	If Set service is not possible for local reasons
0x06	In operation, SET not possible	If Set service is not possible because of application operation
0x07 – 0xFF	Reserved	Shall not be used.

4.3.1.4.12 Coding of the field BlockInfo

This field shall be coded as data type Unsigned16. The allowed values shall be set according to Table 68, Table 69, Table 70 and Table 71.

Table 68 – BlockInfo for SuboptionIPParameter

Bit	Meaning
Bit 0 – 1	See Table 69
Bit 2 – 6	Reserved
Bit 7	See Table 70
Bit 8 – 15	Reserved

Table 69 – Bit 1 and Bit 0 of BlockInfo for SuboptionIPParameter

Bit 1	Bit 0	Meaning for SuboptionIPParameter
0	0	No IP parameter
0	1	IP parameter set
1	0	IP parameter set via DHCP
1	1	Reserved

Table 70 – Bit 7 of BlockInfo for SuboptionIPParameter

Bit 7	Meaning for SuboptionIPParameter
0	No IP address conflict detected ^a IP address active
1	IP address conflict detected ^a IP address not active
^a	An optional local defined test shows that the requested IP address is already in use by a different device. The device is not able to activate the delivered IP address.

Table 71 – BlockInfo for all other suboptions

Bit	Meaning
Bit 0 – 15	Reserved

4.3.1.4.13 Coding of the field DeviceInitiativeValue

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0: DeviceInitiativeValue.Hello

This field shall be set according to Table 72.

Table 72 – DeviceInitiativeValue

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Device does not issue a DCP-Hello-ReqPDU after power on.
0x01	ON	Device does issue a DCP-Hello-ReqPDU after power on.

Bit 1 – 15: DeviceInitiativeValue.reserved

This field shall be set according to 3.4.2.2.

4.3.1.4.14 Coding of the field SignalValue

This field shall be coded as data type Unsigned16 according to Table 73.

Table 73 – SignalValue

Value (hexadecimal)	Meaning	Usage
0x0100	Flash Once	Flash a LED (e.g. the Ethernet LINK LED) or an alternative signalization with duration of 3 s with a frequency of 1 Hz (500 ms on, 500 ms off).

4.3.1.4.15 Coding of the field NameOfStationValue**4.3.1.4.15.1 General**

Each NameOfStation shall be unique in case a device is equipped with more than one interface submodule.

4.3.1.4.15.2 Encoding

This field shall be coded as data type OctetString with 1 to 240 octets. The definition of RFC 5890 and the following syntax applies:

- 1 or more labels, separated by [.]
- Total length is 1 to 240
- Label length is 1 to 63
- Labels consist of [a-z0-9-]
- Labels do not start with [-]
- Labels do not end with [-]
- The first label does not have the form “port-xyz” or “port-xyz-abcde” with a, b, c, d, e, x, y, z = 0...9, to avoid wrong similarity with the field AliasNameValue
- Station-names do not have the form n.n.n.n, n = 0...999

NOTE 1 Labels do only start with ‘xn-’ if the original string contains characters other than [a-z0-9-].

EXAMPLE 1 “device-1.machine-1.plant-1.vendor”

EXAMPLE 2 “mühle1.0.mühle1.plant.com” is coded as “xn-mhle1-kva.xn-lmhle1-vxa4c.plant.com”

NOTE 2 The field NameOfStationValue is not terminated by zero.

NOTE 3 Devices with more than one Ethernet interface need more than one unique NameOfStation. In this case the name of station is a name of interface. Because most devices own only one interface, the name of interface is also the name of station.

4.3.1.4.15.3 Semantics with Identify service

The following rules shall be applied:

- Name of station with length 1 to 240 octets:
Only the device shall answer which has exactly got this name. In the comparison of the two names on equality the DNS conventions shall be considered.
- Name of station with length == 0:
Only those devices shall answer, which have not received a station name yet.

The network representation of the NameOfStation shall be according to 4.3.1.4.15.2.

4.3.1.4.15.4 Semantics with Set service

The following rules shall be applied:

- Name of station with length 1 to 240 octets:
The station shall set its name according to the NameOfStationValue.
- Name of station with length == 0:
The station shall delete its stored name.

The network representation of the NameOfStation shall be according to 4.3.1.4.15.2.

4.3.1.4.16 Coding of the field NameOfPort

This field shall be coded as OctetString[8] or OctetString[14] as “port-xyz” or “port-xyz-rstuv” where x, y, z is in the range “0”-“9” from 001 up to 255 and r, s, t, u, v is in the range “0”-“9” from 00000 up to 65535. The values x, y, z shall be used to identify the port number. The values r, s, t, u, v shall be used to identify the slot which contains the port.

The values “port-001-00000” shall be used for the first port submodule for an interface within slot 0 if any other slot may contain another port submodule.

The values “port-001” shall be used for the first port submodule for an interface if no other slot may contain another port submodule.

Furthermore, the definition of RFC 5890 shall be applied.

4.3.1.4.17 Coding of the field AliasNameValue

4.3.1.4.17.1 Encoding

This field shall be coded as OctetString. The content shall be the concatenation of the content of the fields NameOfPort and NameOfStation.

$$\text{AliasNameValue} = \text{NameOfPort} + "." + \text{NameOfStation} \quad (7)$$

where

AliasNameValue	is an additional name of a node. It addresses one particular interface of this node.
NameOfPort	is the neighbor's name of port
NameOfStation	is the neighbor's name of station

4.3.1.4.17.2 Semantics with Identify service

The following rules shall be applied:

- Alias name with length != 0:
Only the device shall answer which has exactly got this additional alias name. In the comparison of the two names on equality the DNS conventions shall be considered. The device shall not answer with the AliasName. Instead it shall use the current NameOfStation (even if the NameOfStation is not set).
- Name of the station with length == 0:
Not allowed in a request.

NOTE The AliasName is an unique alternative name for a device or an interface of a device. The alias name is derived from topology or neighborhood information. The alias name of station cannot be set or get through means of DCP.

4.3.1.4.18 Coding of the field DeviceRoleDetails

This field shall be coded as data type Unsigned8 with the values according to Table 74.

Table 74 – DeviceRoleDetails

Bit	Meaning	Usage
Bit 0	PNIO Device	The device contains an IO device interface.
Bit 1	PNIO Controller	The device contains an IO controller interface.
Bit 2	PNIO Multidevice	The device contains multiple IO device interfaces.
Bit 3	PNIO Supervisor	The device contains an IO supervisor interface.
Bit 4 – 7	Reserved	—

4.3.1.4.19 Coding of the field MACAddressValue

This field shall be coded as data type OctetString[6]. The value of the field shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

4.3.1.4.20 Coding of the field IPAddress

This field shall be coded as data type OctetString[4]. The encoding of the fields shall be according to RFC 791, RFC 5735 and to Table 75.

The value zero in all octets shall be used to delete a currently set IP address.

Table 75 – IPAddress

Value (CIDR ^a)	Meaning
0.0.0.0 / 8	No IP address assigned
127.0.0.0 / 8	Reserved Loopback
224.0.0.0 / 4	Reserved Multicast
Other	IP address assigned
Reserved	If a subnet is used, the top most IPAddress of this subnet is used as a subnet broadcast address. The assignment of this IPAddress should be rejected.
^a Notation according to classless inter domain routing defined in RFC 4632	

NOTE The object IP suite (IPAddress, Subnetmask and StandardGateway) exist only one time. The flag “store permanently” is an attribute of the object and changed in conjunction with every write access.

4.3.1.4.21 Coding of the field Subnetmask

This field shall be coded as data type OctetString[4]. The encoding of the fields shall be according to RFC 791, RFC 5735 and to Table 76.

Table 76 – Subnetmask

Value	Number of hosts	Meaning
255.255.255.255	1	Subnet mask assigned
255.255.255.254	2	
255.255.255.252	4	
255.255.255.248	8	
255.255.255.240	16	
255.255.255.224	32	

Value	Number of hosts	Meaning
255.255.255.192	64	
255.255.255.128	128	
255.255.255.000	256	
255.255.254.000	512	
255.255.252.000	1 024	
255.255.248.000	2 048	
255.255.240.000	4 096	
255.255.224.000	8 192	
255.255.192.000	16 384	
255.255.128.000	32 768	
255.255.000.000	65 536	
255.254.000.000	131 072	
255.252.000.000	262 144	
255.248.000.000	524 288	
255.240.000.000	1 048 576	
255.224.000.000	2 097 152	
255.192.000.000	4 194 304	
255.128.000.000	8 388 608	
255.000.000.000	16 777 216	
254.000.000.000	33 554 432	
252.000.000.000	67 108 864	Subnet mask valid, support optional according to the rules of IANA
248.000.000.000	134 217 728	
240.000.000.000	268 435 456	
224.000.000.000	536 870 912	
192.000.000.000	1 073 741 824	Subnet mask valid, but rejected according to the rules of IANA
128.000.000.000	2 147 483 648	
000.000.000.000	4 294 967 296	Subnet mask valid, but rejected according to the rules of IANA if a StandardGateway is assigned. Subnet mask valid and assigned if no StandardGateway is assigned.
Other	Reserved	Invalid subnet mask

4.3.1.4.22 Coding of the field StandardGateway

This field shall be coded as data type OctetString[4]. The encoding of the field shall be according to RFC 791, RFC 5735, Table 75 and Table 77.

If no IPAddress is assigned, the values of IPAddress, Subnetmask and StandardGateway shall be set to zero.

Table 77 – StandardGateway

Value (CIDR ^a)	Meaning
0.0.0.0 / 32	No standard gateway assigned The application shall set up the local IP implementation to avoid IP routing
Own IPAddress / 32	No standard gateway assigned The application shall set up the local IP implementation to avoid IP routing
Other ^b	Standard gateway assigned Should be checked against the field subnetmask
Reserved ^c	No standard gateway assigned
^a Notation according to classless inter domain routing defined in RFC 1518 and RFC 1519	
^b Valid according to Table 75 and Table 76	
^c Invalid according to Table 75 and Table 76	

The checking rules for the StandardGateway depends on the existence of more than one interface. For more interfaces, only one StandardGateway (only one object) shall exist. This means that every manipulation of the StandardGateway from any interface has an impact on all other interfaces. Thus, the checking in this case shall be done by local means and local rules.

4.3.1.4.23 Coding of the fields DHCPParameterValue using DHCP Option 61

The client identifier can be used by a DHCP client to request its IP parameters from a DHCP server. A device shall be able to use the client identifier from the data set in the following way:

4.3.1.4.23.1 Coding of the field DHCPParameterLength

This field shall be coded as data type Unsigned16 and shall contain the length of the specific subsequent data.

4.3.1.4.23.2 Use of MACAddress as client identifier

If the device shall obtain its IP parameters using the DHCP protocol and use the MAC address as a client identifier, then the option 61 shall have the following values:

SuboptionDHCP = 61

DHCPParameterLength = 1

DHCPParameterData = 1

4.3.1.4.23.3 Use of NameOfStation as client identifier

If the device shall obtain its IP parameters using the DHCP protocol and use the NameOfStation as client identifier the option 61 shall have the following values (in addition to RFC 2132):

SuboptionDHCP = 61

DHCPParameterLength = 1

DHCPParameterData = 0

If the name of station of the device has been changed the DHCP request shall automatically use the new NameOfStation in the next DHCP request.

4.3.1.4.23.4 Use of arbitrary client identifier

If the device shall obtain its IP parameters using the DHCP protocol and use an arbitrary string as client identifier the option 61 shall have the following values:

SuboptionDHCP	= 61
DHCPPParameterLength	=> 1
DHCPPParameterData[first octet]	= 0
DHCPPParameterData[second octet ...n]	= either "NameOfStation" or "AliasName"

4.3.1.4.24 Coding of the fields DHCPPParameterValue using DHCP Option 255

The DHCP option 255 is defined as END by RFC 2131. It is used automatically by the DHCP client or server. Thus this option is overloaded by DCP and shall be used in the following way:

SuboptionDHCP	= 255
DHCPPParameterLength	= 1
DHCPPParameterData	= 0: Don't use DHCP (Default) = 1: Don't use DHCP but set all DHCPOptions to their "Reset to factory" value = 2: Use DHCP with the given set of DHCPOptions

NOTE If DHCP is enabled, than all earlier set IPAddress, Subnetmask and StandardGateway values are deleted. These fields are now controlled by DHCP.

This option should be the last one in a list of DHCP options in a set request.

4.3.1.4.25 Coding of the field ManufacturerOUI

This field shall be coded as OctetString[3] with the Organizationally unique identifier (OUI) as defined by IEEE 802.

NOTE The value of this central administrative number is given by the IEEE Registration Authority Committee. Available at <<http://standards.ieee.org/regauth>>

4.3.1.4.26 Coding of the field ManufacturerSpecificString

This field shall be coded as data type OctetString and shall contain the values of the related option and suboptions. If the BlockValue has an odd length a padding octet shall be added at the end in order to be Unsigned16 aligned. The padding octet shall have the value 0.

4.3.1.4.27 Coding of the field DeviceVendorValue

This field shall be coded as data type VisibleString and shall contain the same content as the field DeviceType without the trailing blanks.

4.3.1.4.28 Coding of the field DHCPParameterData

This field shall be coded as data type OctetString and shall contain the values of the related option and suboptions. If the BlockValue has an odd length a padding octet shall be added at the end in order to be Unsigned16 aligned. The padding octet shall have the value 0.

4.3.1.4.29 Coding of the field AddDataValue

This field shall be coded as data type OctetString and shall contain the values of the related option and suboptions. If the BlockValue has an odd length a padding octet shall be added at the end in order to be Unsigned16 aligned. The padding octet shall have the value 0.

4.3.2 DCP protocol state machines

4.3.2.1 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.3.2.2 Application Relationship Protocol Machines

4.3.2.2.1 DCP Unicast Sender

4.3.2.2.1.1 Primitive definitions

4.3.2.2.1.1.1 Primitives exchanged between remote machines

The remote service primitives including their associated parameters issued or received by DCPUCS are described in the service definition and shown in Table 78.

Table 78 – Remote primitives issued or received by DCPUCS

Primitive	Source	Destination	Associated parameters	Functions
DCP_Get.req	local means	DCPUCS	CREP, DA, ListOfOptions	This service is used to fetch data from the remote DCP ASE. The parameter DA contains the Layer2 address of the server and the parameter ListOfOptions the requested information.
DCP_Set.req	CTLDINA	DCPUCS	CREP, DA, ListOfData, ListOfControlCommands	This service is used to convey data to the remote DCP ASE. The parameter DA contains the Layer2 address of the server and the parameter ListOfData the requested information. The ListOfControlCommand contains steering information.
LMPM_A_Data.cnf (-)	LMPM	DCPUCS	CREP, LMPM_status	—
LMPM_A_Data.cnf (+)	LMPM	DCPUCS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPUCS	CREP, DA, SA, Prio, A_SDU	—
DCP_Get.cnf (-)	DCPUCS	local means	ERRCLS, ERRCODE	—
DCP_Get.cnf (+)	DCPUCS	local means	ListOfData	—
DCP_Set.cnf (-)	DCPUCS	CTLDINA	ERRCLS, ERRCODE	—
DCP_Set.cnf (+)	DCPUCS	CTLDINA	CREP, ListOfResponse	This service conveys the data and the execution info.

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.req	DCPUCS	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.1.1.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by DCPUCS are described in the service definition and shown in Table 79.

Table 79 – Local primitives issued or received by DCPUCS

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.3.2.2.1.2 State transition diagram

The state transition diagram of the DCPUCS is shown in Figure 14.

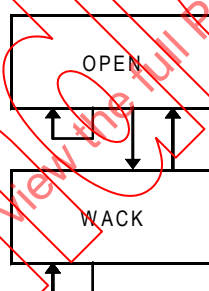


Figure 14 – State transition diagram of DCPUCS

States of the DCPUCS

- OPEN** Wait for a DCP_Get or DCP_Set request from the application and send it to the server. Then change to state WACK and wait for the response.
- WACK** Wait for a DCP_Get or DCP_Set response from the server and create a DCP_Get or DCP_Set confirmation.
If the server doesn't respond, retry the request and after the retry limit is expired create a negative confirmation.

4.3.2.2.1.3 State machine description

The state machine gets its initial values from the DCP ASE attributes. Within the OPEN state, DCP Get and Set service requests issued by the DCP user are transformed to Data Link Layer services. After issuing the transmission of data, the WACK state is used to wait for the Response-PDU causing a confirmation primitive to the DCP user.

4.3.2.2.1.4 DCPUCS state table

Table 80 contains the complete description of the DCPUCS state table.

Table 80 – DCPUCS state table

#	Current State	Event /Condition =>Action	Next State
1	OPEN	DCP_Get.req () => Pending:= GET DA := From instance identified by CREP XID := SXID A_SDU := DCP-GetReqPDU with parameters from the request Store A_SDU Retry:= Max Retry Limit StartTimer (UC Client Timeout) LMPM_A_Data.req ()	WACK
2	OPEN	DCP_Set.req () => Pending:= SET DA := From instance identified by CREP XID := SXID A_SDU := DCP-SetReqPDU with parameters from the request Store A_SDU Retry:= Max Retry Limit StartTimer (UC Client Timeout) LMPM_A_Data.req ()	WACK
3	OPEN	LMPM_A_Data.ind () => ignore	OPEN
4	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
5	OPEN	TimerExpired (UC Client Timeout) => ignore	OPEN
6	WACK	DCP_Get.req () => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Get.cnf (-)	WACK
7	WACK	DCP_Set.req () => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Set.cnf (-)	WACK
8	WACK	LMPM_A_Data.ind () /DGP-Get-ResPDU && Pending == GET && XID == SXID => SXID := SXID+1 StopTimer (UC Client Timeout) DCP_Get.cnf (+)	OPEN
9	WACK	LMPM_A_Data.ind () /DGP-Get-ResPDU && (Pending != GET XID != SXID) => ignore	WACK

#	Current State	Event /Condition =>Action	Next State
10	WACK	LMPM_A_Data.ind () /DCP-Set-ResPDU && Pending == SET && XID == SXID => SXID:= SXID+1 StopTimer (UC Client Timeout) DCP_Set.cnf (+)	OPEN
11	WACK	LMPM_A_Data.ind () /DCP-Set-ResPDU && (Pending != SET XID != SXID) => ignore	WACK
12	WACK	LMPM_A_Data.cnf (+) /LMPM_status == OK => ignore	WACK
13	WACK	LMPM_A_Data.cnf (-) /LMPM_status != OK && Pending == GET => ERRCLS := PROTOCOL ERRCODE := LMPM StopTimer (UC Client Timeout) DCP_Get.cnf (-)	OPEN
14	WACK	LMPM_A_Data.cnf (-) /LMPM_status != OK && Pending == SET => ERRCLS := PROTOCOL ERRCODE := LMPM StopTimer (UC Client Timeout) DCP_Set.cnf (-)	OPEN
15	WACK	TimerExpired (UC Client Timeout) /Retry != 0 => A_SDU:=from Stored A_SDU Retry:= Retry-1 StartTimer (UC Client Timeout) LMPM_A_Data.req ()	WACK
16	WACK	TimerExpired (UC Client Timeout) /Retry == 0 && Pending == GET => ERRCLS := PROTOCOL ERRCODE := TIMEOUT DCP_Get.cnf (-)	OPEN
17	WACK	TimerExpired (UC Client Timeout) /Retry == 0 && Pending == SET => ERRCLS := PROTOCOL ERRCODE := TIMEOUT DCP_Set.cnf (-)	OPEN

#	Current State	Event /Condition =>Action	Next State
18	WACK	LMPM_A_Data.ind () /(DCP-Set-ResPDU DCP-Get-ResPDU) => ignore	WACK

4.3.2.2.1.5 Functions, Macros, Timers and Variables

Table 81 contains the functions, macros, timers and variables used by the DCPUCS and their arguments and their descriptions.

Table 81 – Functions, Macros, Timers and Variables used by the DCPUCS

Name	Type	Function/Meaning
StartTimer	Function	This function is used to start or restart a timer.
StopTimer	Function	This function is used to stop a timer.
TimerExpired	Function	This function signals that a timer is expired.
UC Client Timer	Timer	This Timer, restarted on the requestor when sending an unicast request, will terminate waiting for an response.
ERRCLS	Variable	This local variable contains the entity signaling the error.
ERRCODE	Variable	This local variable contains the error reason in the context of the ERRCLS.
Max Retry Limit	Variable	This local variable contains the retry limit.
Pending	Variable	This local variable holds the information whether a Set or Get service is pending.
Retry	Variable	This local variable contains the retry counter. The maximum value shall be taken from the appropriate DCP ASE attribute.
SXID	Variable	This local variable contains the Transaction ID XID used to identify services uniquely. The initial value shall be derived from a random number generator.
UC Client Timeout	Variable	The value of this local variable shall be taken from the appropriate DCP ASE attribute. The default value is 1 s.

4.3.2.2.2 DCP Unicast Receiver

4.3.2.2.2.1 Primitive definitions

4.3.2.2.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DCPUCR are described in the service definition and shown in Table 82.

Table 82 – Remote primitives issued or received by DCPUCR

Primitive	Source	Destination	Associated parameters	Functions
DCP_Set.rsp	CMINA	DCPUCR	CREP, ListOfResponse	—
LMPM_A_Data.cnf	LMPM	DCPUCR	CREP, LMPM_status	—

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.ind	LMPM	DCPUCR	CREP, DA, SA, Prio, A_SDU	—
DCP_Set.ind	DCPUCR	CMINA	CREP, ListOfData, ListOfControlCommands	—
LMPM_A_Data.req	DCPUCR	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.1.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by DCPUCR are described in the service definition and shown in Table 83.

Table 83 – Local primitives issued or received by DCPUCR

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.3.2.2.2 State transition diagram

The state transition diagram of the DCPUCR is shown in Figure 15.

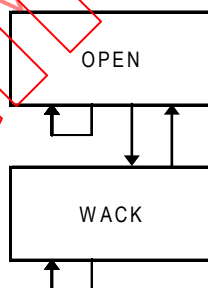


Figure 15 – State transition diagram of DCPUCR

States of the DCPUCR

OPEN	Wait for a DCP Get or Set service and prepare the response for GET
WACK	Wait for the application response for SET to create the DCP set response

4.3.2.2.3 State machine description

Within the OPEN state, the state machine is waiting for the first LMPM_A_Data service indication. After passing the indication to the user the WACK state is entered. A response service will activate an LMPM_A_Data request and bring the machine back to the OPEN state.

4.3.2.2.4 DCPUCR state table

Table 84 contains the complete description of the DCPUCR state table. A LMPM_A_Data.ind primitive will be accepted if type is DCP-Get-ResPDU or DCP-Set-ResPDU.

Table 84 – DCPUCR state table

#	Current State	Event /Condition =>Action	Next State
1	OPEN	DCP_Set.rsp () => ignore	OPEN
2	OPEN	LMPM_A_Data.ind () /DCP-Get-ReqPDU && (SAM == NIL SAM == SA) && (DA != Multicast) && ValidSelectors => SAM := SA SXID := XID SDA := SA DA := SDA XID := SXID A_SDU := Create DCP-Get-ResPDU with ListOfData taken from the DCP ASE according the ListOfData of the DCP-Get-ReqPDU StartTimer (Client Hold Time) LMPM_A_Data.req ()	OPEN
3	OPEN	LMPM_A_Data.ind () /DCP-Get-ReqPDU && (SAM == NIL SAM == SA) && (DA != Multicast) && !ValidSelectors => SAM := SA SXID := XID SDA := SA DA := SDA XID := SXID ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet A_SDU := Create DCP-Get-ResPDU with error information StartTimer (Client Hold Time) LMPM_A_Data.req ()	OPEN
4	OPEN	LMPM_A_Data.ind () /DCP-Set-ReqPDU && (SAM == NIL SAM == SA) && (DA != Multicast) => SAM := SA SXID := XID SDA := SA DCP_Set.ind (CREP, ListOfData)	WACK
5	OPEN	LMPM_A_Data.ind () /DCP-Get-ReqPDU && (SAM != NIL && SAM != SA) => ignore	OPEN
6	OPEN	LMPM_A_Data.ind () /DCP-Set-ReqPDU && (SAM != NIL && SAM != SA) => ignore	OPEN

#	Current State	Event /Condition =>Action	Next State
7	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
8	OPEN	TimerExpired (Client Hold Time) => SAM:= NIL	OPEN
9	WACK	DCP_Set.rsp () => DA := SDA XID := SXID A_SDU := DCP-Set-ResPDU containing the response from the application StartTimer (Client Hold Time) LMPM_A_Data.req ()	OPEN
10	WACK	LMPM_A_Data.ind () => ignore	WACK
11	WACK	LMPM_A_Data.cnf () => ignore	WACK

4.3.2.2.2.5 Functions, Macros, Timers and Variables

Table 85 contains the functions, macros, timers and variables used by the DCPUCR and their arguments and their descriptions.

Table 85 – Functions, Macros, Timers and Variables used by the DCPUCR

Name	Type	Function/Meaning
StartTimer	Function	This function is used to start or restart a timer
StopTimer	Function	This function is used to stop a timer
TimerExpired	Function	This function is issued if a timer expires
ValidSelectors	Macro	Check the PDU against the DCP definition and the DCP ASE
Client Hold Timer	Timer	This timer is used to protect the server from multiple client access during state OPEN. It protects the client reaction time for the next request and is independent from the server reaction time.
Client Hold Time	Variable	The value of this time is by default 3 s
SAM	Variable	This local variable contains the SA of the client which is at the moment "owner" of the server.
SDA	Variable	This local variable contains the last valid SA conveyed with a LMPM_A_Data indication with valid data and is used for the generating of a response
SXID	Variable	This local variable contains the Transaction ID XID used to identify services uniquely in the context of the client. The value shall be taken from the request PDU.

4.3.2.2.3 DCP Multicast Sender

4.3.2.2.3.1 Primitive definitions

4.3.2.2.3.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DCPMCS are described in the service definition and shown in Table 86.

Table 86 – Remote primitives issued or received by DCPMCS

Primitive	Source	Destination	Associated parameters	Functions
DCP_Identify.req	CTLDINA	DCPMCS	ListOfFilter, ResponseDelayFactor	—
LMPM_A_Data.cnf (-)	LMPM	DCPMCS	CREP, LMPM_status	—
LMPM_A_Data.cnf (+)	LMPM	DCPMCS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPMCS	CREP, DA, SA, Prio, A_SDU	—
DCP_Identify.cnf (-)	DCPMCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCP_Identify.cnf (+)	DCPMCS	CTLDINA	ListOfDevices	—
LMPM_A_Data.req	DCPMCS	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.3.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DCPMCS are described in the service definition and shown in Table 87.

Table 87 – Local primitives issued or received by DCPMCS

Primitive	Source	Destination	Associated parameters	Functions
DCP_Identify_ind	DCPMCS	CTLDINA	SA, ListOfDevices	—

4.3.2.2.3.2 State transition diagram

The state transition diagram of the DCPMCS is shown in Figure 16.

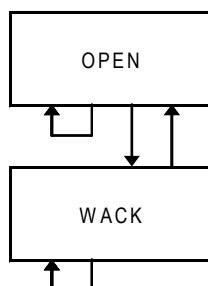


Figure 16 – State transition diagram of DCPMCS

States of the DCPMCS

OPEN	Wait for an Identify service request.
WACK	Wait for all responses until a certain time.

4.3.2.2.3.3 State machine description

The machine waits in the OPEN state for an Identify service request. After issuing the transmission of the data the WACK state is used to wait for all responses until a certain time. The timeout causes to issue the Identify confirmation service primitive with all received responses to the DCP user and sets the state machine to the OPEN state.

4.3.2.2.3.4 DCPMCS state table

Table 88 contains the complete description of the DCPMCS state machine. A LMPM_A_Data.ind primitive shall be accepted if type is DCP-IdentifyResPDU.

Table 88 – DCPMCS state table

#	Current State	Event /Condition =>Action	Next State
1	OPEN	DCP_Identify.req () => XID := SXID DA := DCP_MulticastMACAdd A_SDU := DCP-Identify-ReqPDU First := TRUE StartTimer (Response delay timeout) LMPM_A_Data.req ()	WACK
2	OPEN	LMPM_A_Data.ind () => ignore	OPEN
3	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
4	OPEN	TimerExpired (Response delay timeout) => ignore	OPEN
5	WACK	DCP_Identify.req () => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Identify.cnf (-)	WACK
6	WACK	LMPM_A_Data.ind () /(DCP-Identify-ResPDU && XID == SXID) && FIRST == FALSE => ListOfDevices.SA += SA ListOfDevices.ListOfData += A_SDU	WACK
7	WACK	LMPM_A_Data.ind () /(DCP-Identify-ResPDU && XID == SXID) && First == TRUE => ListOfDevices.SA += SA ListOfDevices.ListOfData += A_SDU First := FALSE DCP_Identify_ind (SA, ListOfDevices)	WACK

#	Current State	Event /Condition =>Action	Next State
8	WACK	LMPM_A_Data.ind () /(DCP-Identify-ResPDU && XID == SXID) => ignore	WACK
9	WACK	LMPM_A_Data.cnf (+) => ignore	WACK
10	WACK	LMPM_A_Data.cnf (-) => ERRCLS := PROTOCOL ERRCODE := LMPM StopTimer (Response delay timeout) DCP_Identify.cnf (-)	OPEN
11	WACK	TimerExpired (Response delay timeout) => ListOfDevices := Stored ListOfDevices SXID := SXID + 1 DCP_Identify.cnf (+)	OPEN

4.3.2.2.3.5 Functions, Macros, Timers and Variables

Table 89 contains the functions, macros, timers and variables used by the DCPMCS and their arguments and their descriptions.

Table 89 – Functions used by the DCPMCS

Name	Type	Function/Meaning
SXID	Variable	This local variable contains the Transaction ID XID used to Identify services uniquely. It shall be changed with every request.
First	Variable	This variable is used for the speed up of the connection establishing.
Response delay timer	Timer	This timer is used to build a timeout for the DCP Identify response
Response delay timeout	Variable	The value is derived from the ResponseDelayFactor
TimerExpired	Function	This function is issued if a timer expires
StartTimer	Function	This function is used to start or restart a timer
StopTimer	Function	This function is used to stop a timer

4.3.2.2.4 DCP Multicast Receiver

4.3.2.2.4.1 Primitive definitions

4.3.2.2.4.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DCPMCR are described in the service definition and shown in Table 90.

Table 90 – Remote primitives issued or received by DCPMCR

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	DCPMCR	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPMCR	CREP, DA, SA, Prio, A_SDU	—
LMPM_A_Data.req	DCPMCR	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.4.1.2 Primitives exchanged between local machines

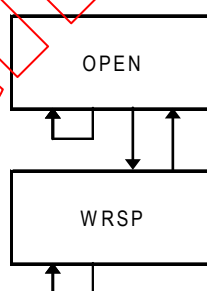
The local primitives including their associated parameters issued or received by DCPMCR are described in the service definition and shown in Table 91.

Table 91 – Local primitives issued or received by DCPMCR

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.3.2.2.4.2 State transition diagram

The state transition diagram of the DCPMCR is shown in Figure 17.

**Figure 17 – State transition diagram of DCPMCR**

States of the DCPMCR

OPEN	Waiting for DCP-IdentifyReqPDUs and compare the local data with the ListOfFilter. If a hit is detected start the response delay and change to the next state.
WRSP	Waits the response delay to send the response in case the DCP user has recognized a match with local data.

4.3.2.2.4.3 State machine description

The state machine is waiting for DCP-IdentifyReqPDUs in the OPEN state. The filter list shall be passed to the DCP user to check the filter criteria. The state machine waits in the state WRSP a certain time calculated by means of the clients Response Delay Factor to send the

response in case the DCP user has recognized a match with local data. Otherwise, nothing shall be sent.

4.3.2.2.4.4 DCPMCR state table

Table 92 contains the complete description of the DCPMCR state machine. A LMPM_A_Data.ind primitive shall be accepted if Type is DCP-IdentifyReqPDU.

Table 92 – DCPMCR state table

#	Current State	Event /Condition =>Action	Next State
1	OPEN	LMPM_A_Data.ind () /DCP-Identify-ReqPDU && Successful comparison with ListOfFilters => SAM := SA SXID := XID ResponseDelay :=CalculateResponseDelay(ResponseDelayFactor, MACAddress) StartTimer (ResponseDelay)	WRSP
2	OPEN	LMPM_A_Data.ind () /DCP-Identify-ReqPDU && ! Successful comparison with ListOfFilters => ignore	OPEN
3	OPEN	LMPM_A_Data.ind () /IDCP-Identify-ReqPDU => ignore	OPEN
4	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
5	WRSP	TimerExpired (ResponseDelay) => DA := SAM XID := SXID A_SDU := Create DCP-Identify-ResPDU with ListOfData taken from the DCP ASE according the ListOfFilters of the DCP-Identify-ReqPDU LMPM_A_Data.req ()	OPEN
6	WRSP	LMPM_A_Data.ind () => ignore	WRSP
7	WRSP	LMPM_A_Data.cnf () => ignore	WRSP

4.3.2.2.4.5 Functions, Macros, Timers and Variables

Table 93 contains the functions or macros used by the DCPMCR and their arguments and their descriptions.

Table 93 – Functions, Macros, Timers and Variables used by the DCPMCR

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or restarts a timer
StopTimer	Function	This local function stops a timer
TimerExpired	Function	This local function is issued if a timer expires

Name	Type	Function/Meaning
CalculateResponseDelay	Macro	This local macro calculates the ResponseDelay according the rules of this standard using the ResponseDelayFactor and the interface MACAddress.
Successful comparison with ListOfFilters	Macro	Checks the PDU against the parameters stored in the DCP ASE
ResponseDelayTimer	Timer	This timer is used to create the response delay
ResponseDelay	Variable	This variable contains the value according the calculation rules stated in this standard.
SAM	Variable	This local variable contains the SourceAddress of the requester used for the response.
SXID	Variable	This local variable contains the Transaction ID XID used to identify services uniquely.

4.3.2.2.5 DCP Hello Multicast Sender

4.3.2.2.5.1 Primitive definitions

4.3.2.2.5.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DCPHMCS are described in the service definition and shown in Table 94.

Table 94 – Remote primitives issued or received by DCPHMCS

Primitive	Source	Destination	Associated parameters	Functions
DCP_Hello.req	CMINA	DCPHMCS	ListOfData	—
LMPM_A_Data.cnf (-)	LMPM	DCPHMCS	CREP, LMPM_status	—
LMPM_A_Data.cnf (+)	LMPM	DCPHMCS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPHMCS	CREP, DA, SA, Prio, A_SDU	—
DCP_Hello.cnf (-)	DCPHMCS	CMINA	ERRCLS, ERRCODE	—
DCP_Hello.cnf (+)	DCPHMCS	CMINA	—	—
LMPM_A_Data.req	DCPHMCS	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.5.1.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by DCPHMCS are described in the service definition and shown in Table 95.

Table 95 – Local primitives issued or received by DCPHMCS

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.3.2.2.5.2 State transition diagram

The state transition diagram of the DCPHMCS is shown in Figure 18.

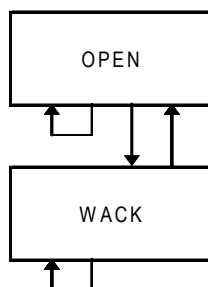


Figure 18 – State transition diagram of DCPHMCS

States of the DCPHMCS

OPEN	Wait for a Hello service request from the upper layer
WACK	Wait for the Hello confirmation from the lower layer

4.3.2.2.5.3 State machine description

The machine waits in the OPEN state for an Hello service request. After issuing the transmission of the data the WACK state is used to wait for the transmit confirmation.

4.3.2.2.5.4 DCPHMCS state table

Table 96 contains the complete description of the DCPHMCS state machine.

Table 96 – DCPHMCS state table

#	Current State	Event /Condition =>Action	Next State
1	OPEN	DCP_Hello.req () => DA := DCP_MulticastMACAdd for Hello A_SDU := DCP-Hello-ReqPDU LMPM_A_Data.req ()	WACK
2	OPEN	LMPM_A_Data.ind () => ignore	OPEN
3	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
4	WACK	LMPM_A_Data.ind () => ignore	WACK
5	WACK	LMPM_A_Data.cnf (+) => DCP_Hello.cnf (+)	OPEN
6	WACK	LMPM_A_Data.cnf (-) => ERRCLS := PROTOCOL ERRCODE := LMPM DCP_Hello.cnf (-)	OPEN

4.3.2.2.5.5 Functions, Macros, Timers and Variables

Table 97 contains the functions, macros, timers and variables used by the DCPHMCS and their arguments and their descriptions.

Table 97 – Functions, Macros, Timers and Variables used by the DCPHMCS

Name	Type	Function/Meaning
SXID	Variable	The XID shall be ignored by the Hello receiver. It may be incremented by the sender.

4.3.2.2.6 DCP Hello Multicast Receiver

4.3.2.2.6.1 Primitive definitions

4.3.2.2.6.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DCPHMCR are described in the service definition and shown in Table 98.

Table 98 – Remote primitives issued or received by DCPHMCR

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	DCPHMCR	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPHMCR	CREP, DA, SA, Prio, A_SDU	—
DCP_Hello.ind	DCPHMCR	CTLDINA	ListOfData	—

4.3.2.2.6.1.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by DCPHMCR are described in the service definition and shown in Table 99.

Table 99 – Local primitives issued or received by DCPHMCR

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.3.2.2.6.2 State transition diagram

The state transition diagram of the DCPHMCR is shown in Figure 19.

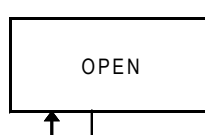


Figure 19 – State transition diagram of DCPHMCR

States of the DCPHMCR

OPEN

waiting for DCP-Hello-ReqPDUs

4.3.2.2.6.3 State machine description

The state machine is waiting for DCP-Hello-ReqPDUs in the OPEN state.

4.3.2.2.6.4 DCPHMCR state table

Table 100 contains the complete description of the DCPHMCR state machine. A LMPM_A_Data.ind primitive will be accepted if Type is DCP-Hello-ReqPDU.

Table 100 – DCPHMCR state table

#	Current State	Event /Condition =>Action	Next State
1	OPEN	LMPM_A_Data.ind () /DCP-Hello-ReqPDU && Successful comparison with ListOfData => DCP_Hello.ind ()	OPEN
2	OPEN	LMPM_A_Data.ind () /!DCP-Hello-ReqPDU => ignore	OPEN
3	OPEN	LMPM_A_Data.cnf () => ignore	OPEN

4.3.2.2.6.5 Functions, Macros, Timers and Variables

Table 101 contains the functions, macros, timers and variables used by the DCPHMCR and their arguments and their descriptions.

Table 101 – Functions, Macros, Timers and Variables used by the DCPHMCR

Name	Type	Function/Meaning
Successful comparison with ListOfData	Macro	Compare the PDU content with the local stored information and return TRUE if it fits.

4.3.3 DLL Mapping Protocol Machines

There is no DLL Mapping Protocol Machine (DMPM) defined for this Protocol.

4.4 Precision working time control

4.4.1 FAL syntax description

4.4.1.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.4.1.2 APDU abstract syntax

Table 102 defines the abstract syntax of the PTCP PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs.

Table 102 – PTCP APDU syntax

APDU name	APDU structure
PTCP-PDU	PTCP-SYNC-PDU ^ PTCP-FUP-PDU ^ PTCP-ANNOUNCE-PDU ^ PTCP-DELAY-REQ-PDU ^ PTCP-DELAY-RSP-PDU ^ PTCP-DELAY-FU-RSP-PDU
PTCP-FUP-PDU	PTCP-Header-FUP, PTCP-FollowUpPDU
PTCP-SYNC-PDU	PTCP-Header-Sync, PTCP-RTSyncPDU
PTCP-ANNOUNCE-PDU	PTCP-Header-Announce, PTCP-AnnouncePDU
PTCP-DELAY-REQ-PDU	PTCP-Header-Delay, PTCP-DelayReqPDU
PTCP-DELAY-RSP-PDU	PTCP-Header-Delay, PTCP-DelayResPDU
PTCP-DELAY-FU-RSP-PDU	PTCP-Header-Delay, PTCP-DelayFuResPDU
PTCP-RTSyncPDU	PTCPSubdomain, PTCPTime, PTCPTimeExtension, PTCPMaster, [PTCPOption], PTCPEnd, [APDU_Status] ^a
PTCP-AnnouncePDU	PTCPSubdomain, PTCPMaster, [PTCPOption], PTCPEnd
PTCP-FollowUpPDU	PTCPSubdomain, PTCPTime, [PTCPOption], PTCPEnd
PTCP-DelayReqPDU	PTCPDelayParameter, [PTCPOption], PTCPEnd
PTCP-DelayResPDU	PTCPDelayParameter, PTCPPortParameter, PTCPPortTime, [PTCPOption], PTCPEnd
PTCP-DelayFuResPDU	PTCPDelayParameter, [PTCPOption], PTCPEnd

^a Legacy support

Table 103 defines structures for substitutions of elements of the APDU structures.

Table 103 – PTCP substitutions

Substitution name	Structure
PTCP-Header-Sync	PTCP_reserved_1, PTCP_reserved_2, PTCP_Delay10ns, PTCP_SequenceID, PTCP_Delay1ns_Byte ^d , Padding ^a , PTCP_Delay1ns
PTCP-Header-FUP	[HWPadding*] ^b , PTCP_SequenceID, [Padding*] ^c , PTCP_Delay1ns_FUP
PTCP-Header-Announce	[HWPadding*] ^b , PTCP_SequenceID, [Padding*] ^e
PTCP-Header-Delay	[HWPadding*] ^b , PTCP_SequenceID, [Padding*] ^c , PTCP_Delay1ns
PTCPSubdomain	PTCP_TLVHeader, PTCP_MasterSourceAddress, PTCP_SubdomainUUID
PTCPTime	PTCP_TLVHeader, PTCP_EPOCHNumber, PTCP_Seconds, PTCP_NanoSeconds
PTCPTimeExtension	PTCP_TLVHeader, PTCP_Flags, PTCP_CurrentUTCOffset, [Padding*] ^a
PTCPMaster	PTCP_TLVHeader, PTCP_MasterPriority1, PTCP_MasterPriority2, PTCP_ClockClass, PTCP_ClockAccuracy, PTCP_ClockVariance, [Padding*] ^a
PTCPDelayParameter	PTCP_TLVHeader, PTCP_PortMACAddress
PTCPPortParameter	PTCP_TLVHeader, [Padding*] ^a , PTCP_T2PortRxDelay, PTCP_T3PortTxDelay
PTCPPortTime	PTCP_TLVHeader, [Padding*] ^a , PTCP_T2TimeStamp
PTCPOption	PTCP_TLVHeader, PTCP_OUI, PTCP_SubType, Data*, [Padding*] ^a
PTCPend	PTCP_TLVHeader(0)

^a 32 bit alignment shall be ensured
^b 12 octets padding
^c 2 octets padding
^d Legacy to cover hardware needs
^e 6 octets padding

4.4.1.3 Coding section related to common basic fields

4.4.1.3.1 Coding of the field PTCP_reserved_1

This field shall be coded as data type Unsigned32.

4.4.1.3.2 Coding of the field PTCP_reserved_2

This field shall be coded as data type Unsigned32.

4.4.1.3.3 Coding of the field HWPadding

This field shall be coded as data type Unsigned8.

4.4.1.3.4 Coding of the field PTCP_TLVHeader

The coding of this field shall be according to 3.4.2.4. and the individual bits shall have the following meaning:

Bit 0 – 8: PTCP_TLVHeader.Length

The value contains the sum of subsequent octets of the according block.

Bit 9 – 15: PTCP_TLVHeader.Type

This field shall be coded with the values according to Table 104.

Table 104 – PTCP_TLVHeader.Type

Value (hexadecimal)	Meaning	Use
0x00	PTCPEnd	Mandatory
0x01	PTCPSubdomain	Mandatory
0x02	PTCPTime	Mandatory
0x03	PTCPTimeExtension	Mandatory
0x04	PTCPMaster	Mandatory
0x05	PTCPPortParameter	Mandatory
0x06	PTCPDelayParameter	Mandatory
0x07	PTCPPortTime	Mandatory
0x08 – 0x7E	Reserved	—
0x7F	PTCPOption Organizationally specific	Optional

4.4.1.3.5 Coding section related to PTCP-Header

These fields contains the propagation delay time in nanoseconds. It is parted into a field for the 10 ns part and a field for the 1 ns part. All delay fields are only relevant for Sync-, FollowUp-, DelayRes- without DelayFuRes- and DelayFuRes-Messages.

4.4.1.3.5.1 Coding of the field PTCP_Delay10ns

This field shall be coded as data type Unsigned32 and contains the 10 ns part of the propagation delay time. This field shall be coded with the values according to Table 105.

Table 105 – PTCP_Delay10ns

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	Valid 10 nanosecond part of the propagation delay time
0xFFFFFFFF	Invalid This value indicates an overflow and marks the delay fields as invalid

4.4.1.3.5.2 Coding of the field PTCP_Delay1ns_Byte

The field contains the 1 ns part of the propagation delay time.

NOTE Legacy to cover hardware needs

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0 – 3: PTCP_Delay1ns_Byte.Value

The value contains the 1 ns part of the propagation delay time. This field shall be coded with the values according to Table 106.

Table 106 – PTCP_Delay1ns_Byte.Value

Value (hexadecimal)	Meaning
0x00 – 0x09	Number of ns
0x0A – 0x0F	Reserved

Bit 4 – 7: PTCP_Delay1ns_Byte.Reserved

This field shall be set according to 3.4.2.2.

Instead of this field only the field PTCP_Delay1ns may be used.

4.4.1.3.5.3 Coding of the field PTCP_Delay1ns

This field shall be coded as data type Unsigned32 and contains the 1 ns part of the propagation delay time. This field shall be coded with the values according to Table 107.

Table 107 – PTCP_Delay1ns

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	Valid 1 nanosecond part of the propagation delay time
0xFFFFFFFF	Invalid This value indicates an overflow and marks the delay fields as invalid

4.4.1.3.5.4 Coding of the field PTCP_Delay1ns_FUP

This field shall be coded as data type Integer32 and contains the correction of the propagation delay time in 1 ns. This field shall be coded with the values according to Table 108.

Table 108 – PTCP_Delay1ns_FUP

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	Nanosecond part of the propagation delay time

4.4.1.3.5.5 Coding of the field PTCP_SequenceID

This field shall be coded as data type Unsigned16 with the values according to Table 109.

Table 109 – PTCP_SequenceID

Meaning	Usage
Sync frame	The PTCP master increments this field for every sync frame.
Follow up frame	The PTCP master or a follow up injecting PTCP forwarder mirrors the value from the Sync frame.
Delay request frame	The PTCP delay measurement requester increments this field for every measurement.
Delay response frame	The responder mirrors the value from the Delay request frame.
Delay response follow up frame	The responder uses the value from the Delay response frame.

The receiver of the PTCP frames shall use this field to detect repetitions, loss of frames and timeliness of data. For this reason the PTCP_SequenceID value range shall be divided in the ratio 15/16 for valid and 1/16 for invalid as shown in Figure 20.

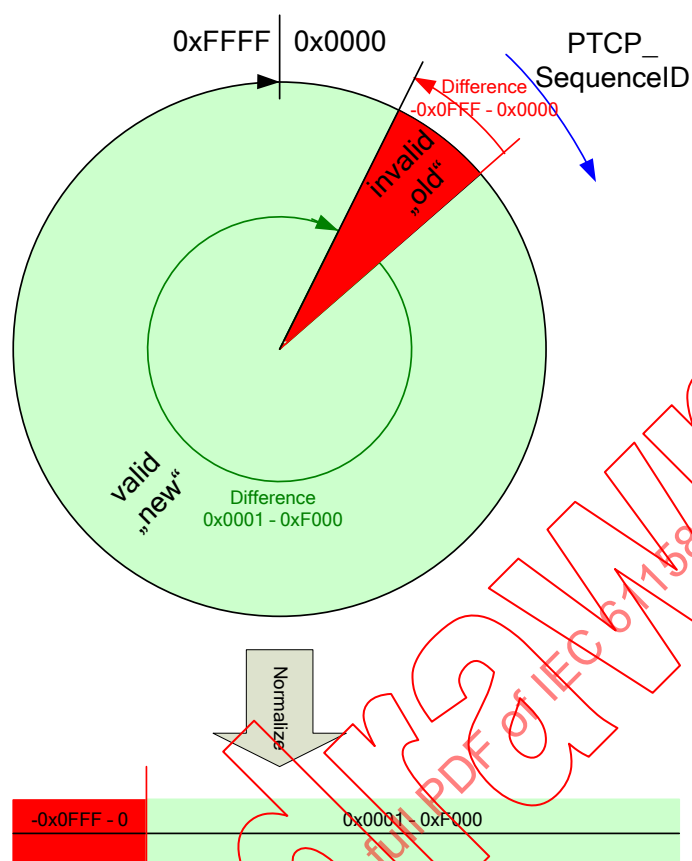


Figure 20 – PTCP_SequenceID value range

4.4.1.3.6 Coding of the field PTCP_OUI

This field shall be coded as OctetString[3] with the Organizationally unique identifier (OUI) as defined in IEEE 802.

NOTE The value of this central administrative number is given by the IEEE Registration Authority Committee. Available at <<http://standards.ieee.org/regauth>>

4.4.1.3.7 Coding of the field PTCP_SubType

This field shall be coded as Unsigned8. The value is organizationally specific. For the OUI the parameter PTCP_SubType shall be coded as in Table 110.

Table 110 – PTCP_SubType for OUI (=00-0E-CF)

Value (hexadecimal)	Meaning	Usage
0x00 – 0xFF	Reserved	—

4.4.1.3.8 Coding of the field PTCP_Seconds

This field shall be coded as data type Unsigned32 with the values according to Table 111. The resolution of this field is 1 s.

Table 111 – PTCP_Seconds

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	Time in s

4.4.1.3.9 Coding of the field PTCP_NanoSeconds

This field shall be coded as data type Unsigned32 with the values according to Table 112. The resolution of this field is 1 ns.

Table 112 – PTCP_NanoSeconds

Value (hexadecimal)	Meaning
0x00000000 – 0x3B9AC9FF	Time in ns
0x3B9ACA00 – 0xFFFFFFFF	Reserved

4.4.1.4 Coding section of PTCP-PDU specific fields

4.4.1.4.1 Coding of the field PTCP_MasterSourceAddress

This field shall be coded as data type OctetString[6]. The value of the field Destination Address shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

4.4.1.4.2 Coding of the field PTCP_SubdomainUUID

This field shall be coded as data type UUID.

NOTE The value NIL indicates no synchronization within the Read Real Sync Data service.

4.4.1.4.3 Coding of the field PTCP_Flags

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 7: PTCP_Flags.reserved_1

This field shall be set according to 3.4.2.2.

Bit 8 – 9: PTCP_Flags.LeapSecond

This field shall be set according to Table 113.

Table 113 – PTCP_Flags.LeapSecond

Value (hexadecimal)	Meaning
0x00	No leap second
0x01	Last minute has 61 s
0x02	Last minute has 59 s
0x03	Reserved

Bit 10 – 15: PTCP_Flags.reserved_2

This field shall be set to zero.

4.4.1.4.4 Coding of the field PTCP_EPOCHNumber

This field shall be coded as data type Unsigned16 and contain the 0x100000000 s part of the time. The value of the PTCP_EPOCHNumber field shall be the value of epoch number of the global time properties data set of the clock issuing this message according to Table 114 and Table 115. The starting point of time is similar to the IEC 61588 the 1970-01-01 T 00:00:00 Z.

Table 114 – Timescale correspondence between MJD, UTC and PTCP_EPOCHNumber

MJD (Modified Julian Day)	UTC (Universal Time Coordinated)	UTC Offset (Leap seconds)	Value PTCP_Seconds	Value PTCP_EPOCH- Number
0:00:00 15 020	1900-01-01 T 00:00:00 Z	—	—	—
0:00:00 40 587	1970-01-01 T 00:00:00 Z	—	0	0
0:00:00 41 317	1972-01-01 T 00:00:00 Z	10	63 672 000 + 10	0
16:57:44 51 357	1999-06-28 T 16:57:44 Z	32	930 589 064 + 32	0
—	2006-01-01 T 00:00:00 Z	33	—	0

NOTE The coding YYYY-MM-DD T HH:MM:SS Z is according ISO 8601, where 'T' signals that a time follows the date and 'Z' signals that the time is UTC.

Table 115 – Timescale correspondence between PTCP_EPOCHNumber, PTCP_Second, PTCP_Nanosecond, CycleCounter and SendClockFactor

CycleCounter value (hexadecimal)	PTCP_EPOCHNumber value (decimal)	PTCP_Second value (decimal)	PTCP_Nanosecond value (decimal)	SendClockFactor value (decimal)
0x0000000000000000	0	0	0	32
0x00000000000010000	0	65	536 000 000	32
0x0000000100000000	0	4 294 967	296 000 000	32
0x0001000000000000	65	2 302 102 470	656 000 000	32

$$\text{PTCP\_Epoch} = \text{PTCP\_EpochNumber} \times 0x100000000 \quad (8)$$

where

PTCP_EPOCH is the helper variable for the calculation of seconds
PTCP_EPOCHNumber is the epoch number

$$\text{PTCP\_Time} = \text{PTCP\_Epoch} + \text{PTCP\_Second} + \text{PTCP\_Nanosecond} \quad (9)$$

where

PTCP_Time is the PTCP time
PTCP_EPOCH is the helper variable for the calculation of seconds
PTCP_Second is the PTCP second value
PTCP_Nanosecond is the PTCP nanosecond value

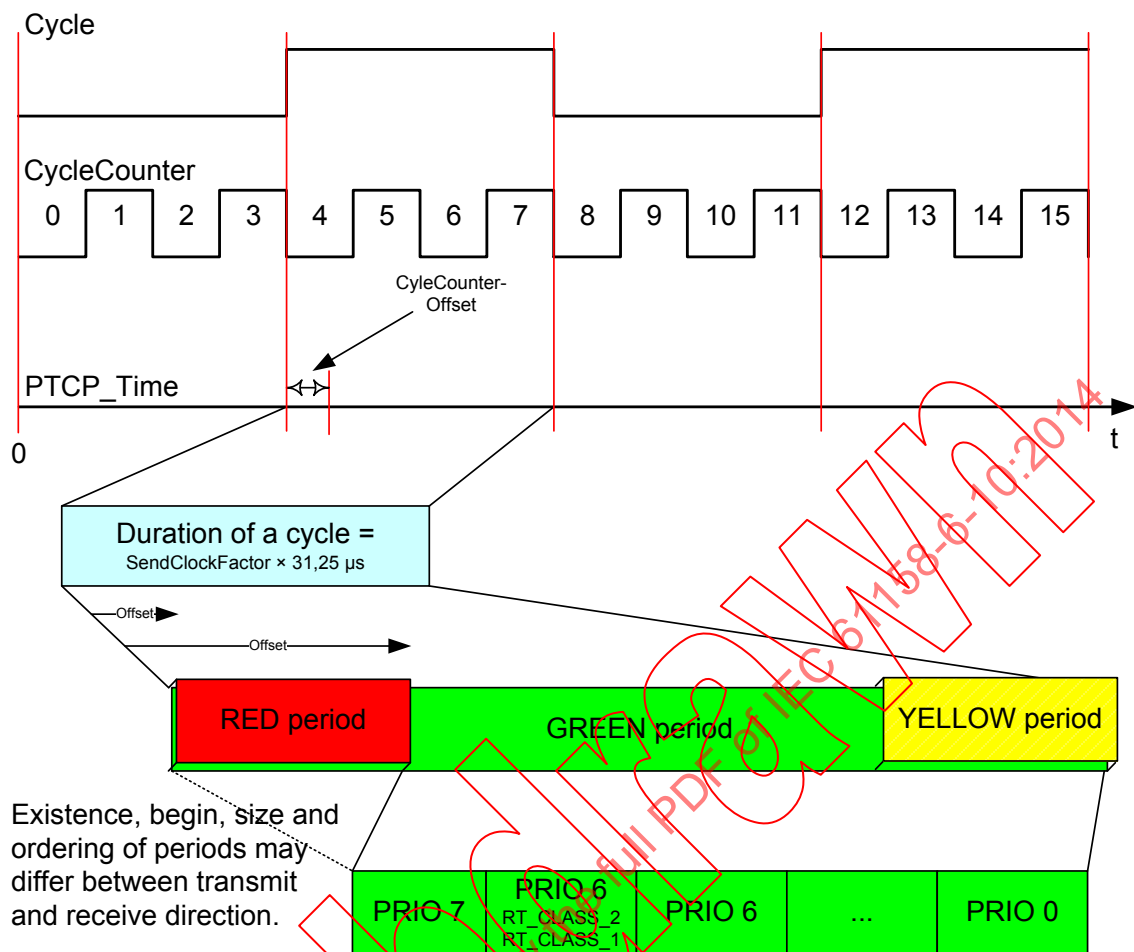


Figure 21 – Timescale correspondence between PTCP_Time and CycleCounter

For the synchronization of the CycleCounter as shown in Figure 21, the calculation shall accept PTCP_Time values with an offset to the cycle begin.

$$\text{Cycle} = \text{PTCP\_Time} \text{ DIV } (\text{SendClockFactor} \times 31,25 \mu\text{s}) \quad (10)$$

where

Cycle is the cycle
PTCP_Time is the PTCP time
SendClockFactor is the factor of the send clock, the length of one phase

$$\text{CycleCounter} = \text{Cycle} \times \text{SendClockFactor} \quad (11)$$

where

CycleCounter is the counter of cycles
Cycle is the cycle
SendClockFactor is the factor of the send clock, the length of one phase

$$\text{CycleCounterOffset} = \text{PTCP\_Time} \text{ MOD } (\text{SendClockFactor} \times 31,25 \mu\text{s}) \quad (12)$$

where

CycleCounterOffset is the offset of the cycle counter

PTCP_Time is the PTCP time

SendClockFactor is the factor of the send clock, the length of one phase

NOTE 1 DIV does an integer division and delivers the quotient.

NOTE 2 MOD does an integer division and delivers the remainder.

4.4.1.4.5 Coding of the field PTCP_CurrentUTCOffset

This field shall be coded as data type Integer16. The value of the current PTCP_CurrentUTCOffset field shall be the value of current_utc_offset of the global time properties data set of the clock issuing this message according to Table 114.

4.4.1.4.6 Coding of the field PTCP_MasterPriority1

PTCP_MasterPriority1 is used in the execution of the best master clock algorithm. Lower values take precedence. The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0 – 2: PTCP_MasterPriority1.Priority

This field shall be set according to Table 116 and Table 117.

Table 116 – PTCP_MasterPriority1.Priority for SyncID == 0 and SyncProperties.Role == 2

Value (hexadecimal)	Usage	Meaning
0x00	Reserved	—
0x01	Primary synchronization master	Clock synchronization (primary master)
0x02	Secondary synchronization master	Clock synchronization (secondary master)
0x03 – 0x07	Reserved	—

Table 117 – PTCP_MasterPriority1.Priority for SyncID == 0 and SyncProperties.Role == 1

Value (hexadecimal)	Usage	Meaning
0x00	Sync slave	Clock synchronization (slave)
0x01 – 0x07	Reserved	—

Bit 3 – 5: PTCP_MasterPriority1.Level

This field shall be set according to Table 118.

Table 118 – PTCP_MasterPriority1.Level

Value (hexadecimal)	Usage	Meaning
0x00	Level 0 (highest)	See Annex J
0x01	Level 1	
0x02 – 0x06	...	
0x07	Level 7 (lowest)	

Bit 6: PTCP_MasterPriority1.Reserved

This field shall be set zero.

Bit 7: PTCP_MasterPriority1.Active

This field shall be set zero for the PDSyncData.

This field shall be set zero for the PTCP-ANNOUNCE-PDU if the sync master doesn't convey PTCP-SYNC-PDU's and shall be set to one if the sync master conveys PTCP-SYNC-PDU's.

Thus this field shall be set to one for the PTCP-SYNC-PDU after the best master selection is done.

4.4.1.4.7 Coding of the field PTCP_MasterPriority2

This field shall be coded as Unsigned8. PTCP_MasterPriority2 is used in the execution of the best master clock algorithm. Lower values take precedence. The value of PTCP_MasterPriority2 shall be configurable according to Table 119.

Table 119 – PTCP_MasterPriority2

Value (hexadecimal)	Usage	Meaning
0x00 – 0xFE	Reserved	—
0xFF	Default	Default

4.4.1.4.8 Coding of the field PTCP_ClockClass

This field shall be coded as data type Unsigned8 with the values according to Table 120. For working clock synchronization the arbitrary timescale (ARB) shall be used.

Table 120 – PTCP_ClockClass for SyncID == 0 (working clock synchronization)

Value (hexadecimal)	Usage	Meaning
0x00 – 0x0C	Reserved	—
0x0D	Shall designate a PTCP clock as a primary reference that is synchronized to an application specific source of time. The timescale distributed shall be ARBITRARY. A clock class 0x0D clock shall not be a slave to another clock in the PTCP domain.	Default
0x0E	Shall designate a PTCP clock that has previously been designated as clock class 0x0D but that has lost the ability to synchronize to an application specific source of time and is in holdover mode and within holdover specifications. The timescale distributed shall be ARBITRARY. A clock class 0x0E clock shall not be a slave to another clock in the PTCP domain	—
0x0F – 0xBA	Reserved	—
0xBB	Degradation alternative B for a PTCP clock of clock class 0x07 that is not within holdover specification. A clock of clock class 0xBB may be a slave to another clock in the PTCP domain.	—
0xBC – 0xC0	Reserved	—
0xC1	Degradation alternative B for a PTCP clock of clock class 0x0E that is not within holdover specification. A clock of clock class 0xC1 may be a slave to another clock in the PTCP domain.	—
0xC2 – 0xFE	Reserved	—
0xFF	Shall be the clock class of a slave-only clock.	—

4.4.1.4.9 Coding of the field PTCP_ClockAccuracy

This field shall be coded as data type Unsigned8 with the values according to Table 121. The PTCP_ClockAccuracy indicates the expected accuracy of a clock when it is master. The value of PTCP_ClockAccuracy shall represent its current status.

Table 121 – PTCP_ClockAccuracy

Value (hexadecimal)	Usage	Meaning
0x00 – 0x1F	Reserved	—
0x20	The time is accurate to within 25 ns	—
0x21	The time is accurate to within 100 ns	Default for working clock synchronization
0x22	The time is accurate to within 250 ns	—
0x23	The time is accurate to within 1 µs	—
0x24	The time is accurate to within 2,5 µs	—
0x25	The time is accurate to within 10 µs	—
0x26	The time is accurate to within 25 µs	—
0x27	The time is accurate to within 100 µs	—
0x28	The time is accurate to within 250 µs	—
0x29	The time is accurate to within 1 ms	—
0x30 – 0xFD	Reserved	—
0xFE	Unknown	—
0xFF	Reserved	—

4.4.1.4.10 Coding of the field PTCP_ClockVariance

This field shall be coded as data type Unsigned16 with the values according to Table 122. The value characterizes the quality of a clock. Each master clock shall maintain an estimate of its inherent precision. This is the precision of the timestamps included in the sync-frames issued by the PTCP synchronization master when it is not synchronized to another PTCP synchronization master using the PTCP protocol.

Table 122 – PTCP_ClockVariance

Value (hexadecimal)	Usage	Meaning
0x0000	Reserved	Default
0x0001 – 0xFFFF	Reserved	—

4.4.1.4.11 Coding of the field PTCP_T2PortRxDelay

This field shall be coded as data type Unsigned32 with the values according to Table 123.

Table 123 – PTCP_T2PortRxDelay

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	Port receive delay (MDI to SHIM) in nanoseconds.

4.4.1.4.12 Coding of the field PTCP_T3PortTxDelay

This field shall be coded as data type Unsigned32 with the values according to Table 124.

Table 124 – PTCP_T3PortTxDelay

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	Port transmit delay (SHIM to MDI) in nanoseconds.

4.4.1.4.13 Coding of the field PTCP_PortMACAddress

This field shall be coded as data type OctetString[6]. The value of the field PTCP_PortMACAddress shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

4.4.1.4.14 Coding of the field PTCP_T2TimeStamp

This field shall be coded as data type Unsigned32 with the values according to Table 125.

Table 125 – PTCP_T2TimeStamp

Value (hexadecimal)	Meaning
0x00000000 – 0xFFFFFFFF	TimeStamp in nanoseconds

4.4.2 AP-Context state machine

There is no AP-Context State Machine defined for this Protocol.

4.4.3 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.4.4 Application Relationship Protocol Machines

4.4.4.1 Generic Synchronization with PTCP

4.4.4.1.1 Timestamp

The PTCP message timestamp point for a transmitted or received frame shall correspond to the leading edge of the first bit of the octet immediately following the Start Frame Delimiter octet of an IEEE 802.3 frame as shown in Figure 22.

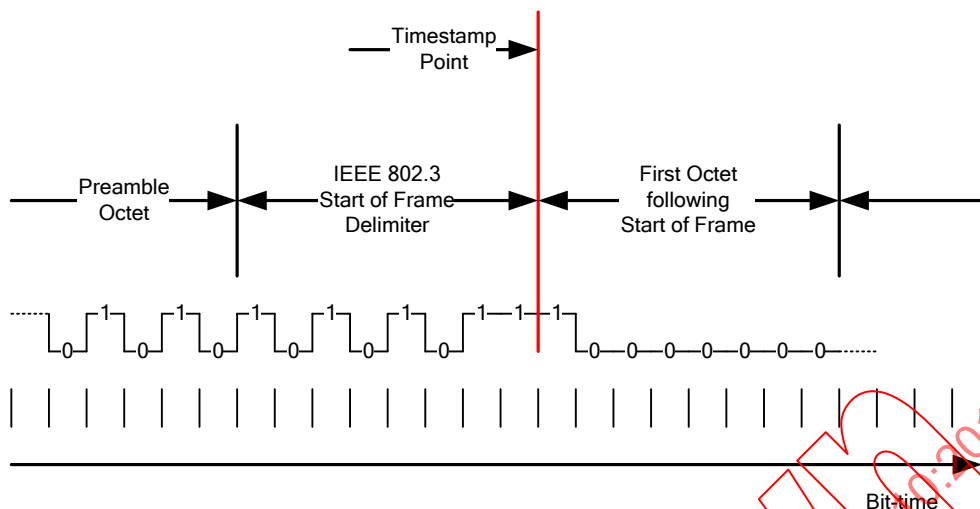


Figure 22 – Message timestamp point

4.4.4.1.2 Timer model

PTCP uses a the timer model shown in Figure 23.

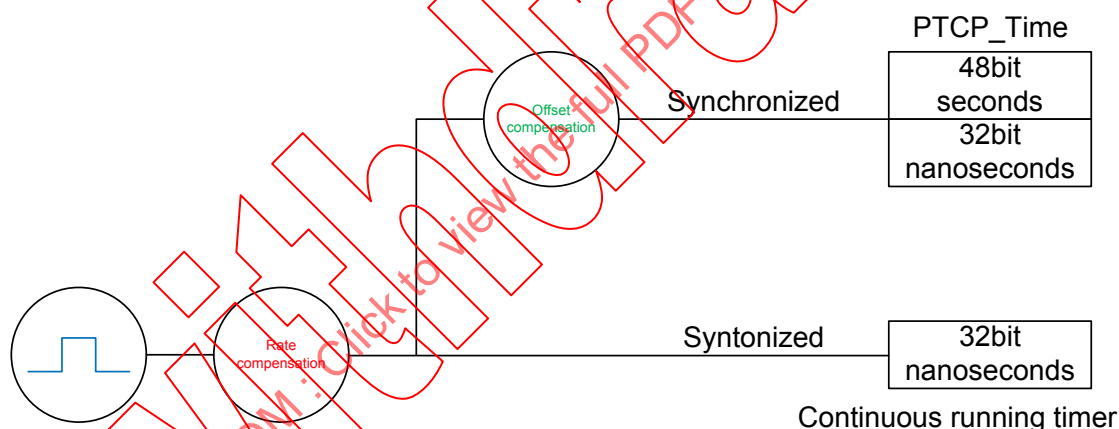


Figure 23 – Timer model

It consist of two timers. The first, at least 32 bit wide syntonized continuous running timer with a resolution of one nanosecond, is used for the line delay and bridge delay measurement, the second 80 bit wide synchronized timer with a resolution of one nanosecond ist used for the PTCP_Time of master or slave.

4.4.4.1.3 Generic model

For each received and transmitted sync, delay request or delay response message a time stamping unit shall generate a time stamp. Figure 24 shows the four timestamps used for synchronization.

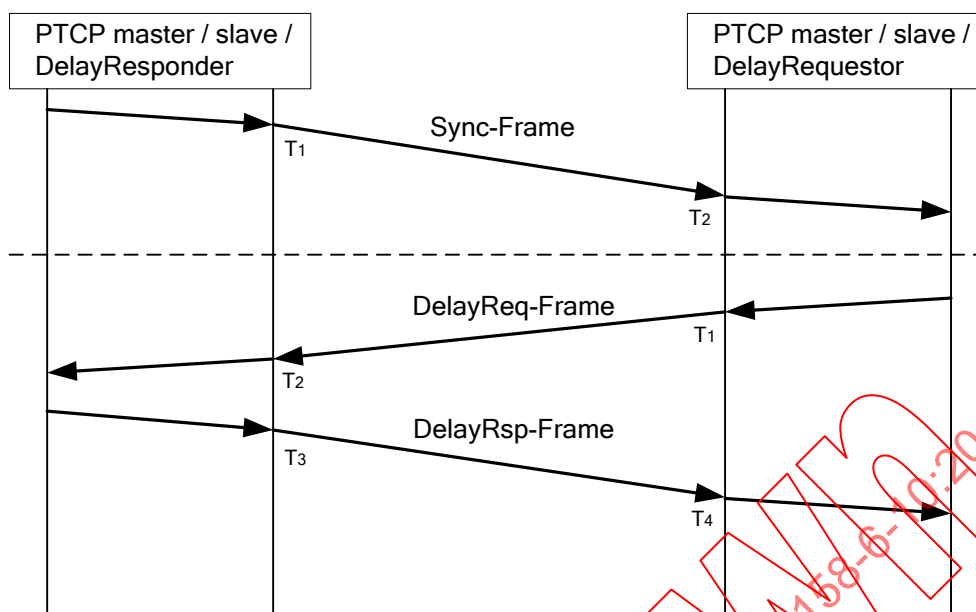


Figure 24 – Four message timestamps

The generic model uses the following formulae.

$$\text{SendTimeStamp} = T_1 \quad (13)$$

$$\text{ReceiptTimeStamp} = T_2 \quad (14)$$

$$\text{SendTimeStamp} = T_3 \quad (15)$$

$$\text{ReceiptTimeStamp} = T_4 \quad (16)$$

where

- T_1 is measured by the local clock of the port which sends a Sync- or a DelayReq-Frame
- T_2 is measured by the local clock of the port which receives a Sync- or a DelayReq-Frame
- T_3 is measured by the local clock of the port which sends a DelayRes-Frame
- T_4 is measured by the local clock of the port which receives a DelayRes-Frame

4.4.4.1.4 Line delay measurement and peer rate compensation

The line delay measurement and the peer rate compensation calculation between DelayRequestor and DelayResponder are described in Figure 25. DelayReq-, DelayRes and DelayFuRes-Messages shall be used for delay measurement and shall not be propagated. The DelayFuRes-Frame is used to transmit the value of ResDelay to the DelayRequestor.

DelayResponder which are not able to measure the line delay but want to receive the Sync-Frame shall use a DelayRes.req with a ResDelay value of zero.

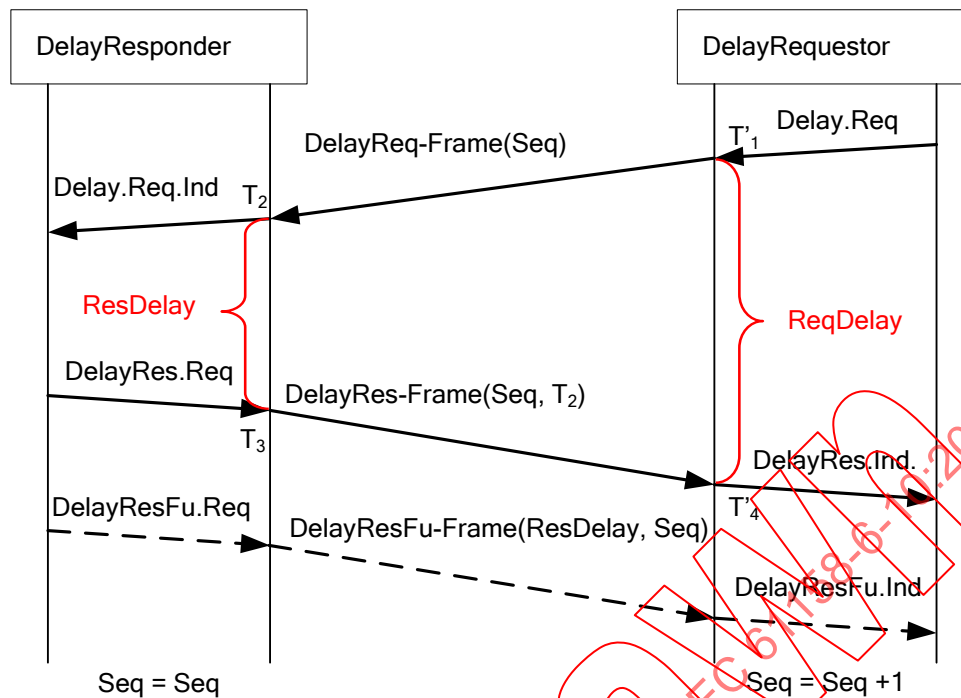


Figure 25 – Line delay protocol with follow up

If a time stamping unit is capable to calculate the response delay time on the fly and insert the response time in the delay response frame no delay follow up message is necessary.

The line delay measurement without DelayFuRes-message between DelayRequestor and DelayResponder is described in Figure 26.

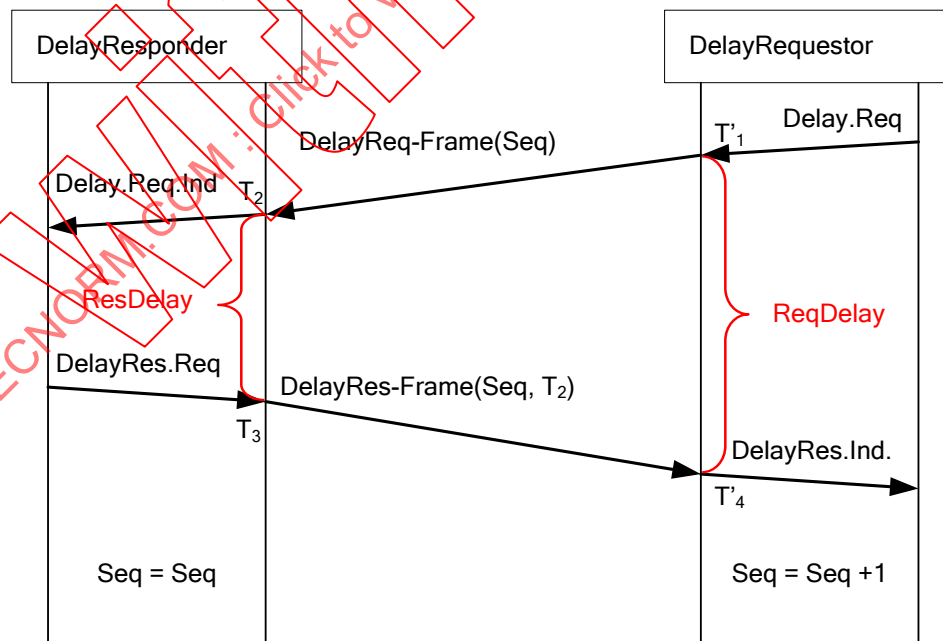


Figure 26 – Line delay protocol without follow up

The calculation of line delay and peer rate compensation factor is shown in following formulae. If the DelayRes-PDU contains the field T_2 , the Formula (19) shall be used, if the ResDelay is zero, RCF_{peer} shall be set to one.

$$\text{ResDelay} = T_3 - T_2 \quad (17)$$

where

ResDelay is the delay at responder side
 T_3 is the time stamp of sending DelayRes-Frame as DelayResponder
 T_2 is the time stamp of receiving DelayReq-Frame as DelayResponder

$$\text{ReqDelay} = T'_4 - T'_1 \quad (18)$$

where

ReqDelay delay requestor
 T'_4 is the time stamp of receiving DelayRes-Frame as DelayRequestor
 T'_1 is the time stamp of sending DelayReq-Frame as DelayRequestor

$$\text{RCF}_{\text{peer}} = \frac{T_2(\text{Seq}) - T_2(\text{Seq} - 1)}{T'_1(\text{Seq}) - T'_1(\text{Seq} - 1)} \quad (19)$$

where

RCF_{peer} is the peer rate compensation factor
 T_2 is the time stamp of receiving DelayReq-Frame as DelayResponder and transmitted using the DelayRes-Frame to the DelayRequestor
 T'_1 is the time stamp of sending DelayReq-Frame as DelayRequestor
 Seq is the sequence variable

$$\text{ResDelay}_{\text{peer}} = \frac{\text{ResDelay}}{\text{RCF}_{\text{peer}}} \quad (20)$$

where

$\text{ResDelay}_{\text{peer}}$ is the delay responder at this peer
 ResDelay is the delay responder
 RCF_{peer} is the peer rate compensation factor

The delay measurement shall be repeated in cycles.

4.4.4.1.5 Line delay calculation

The line delay between two devices supporting PTCP is determined as described in Figure 27.

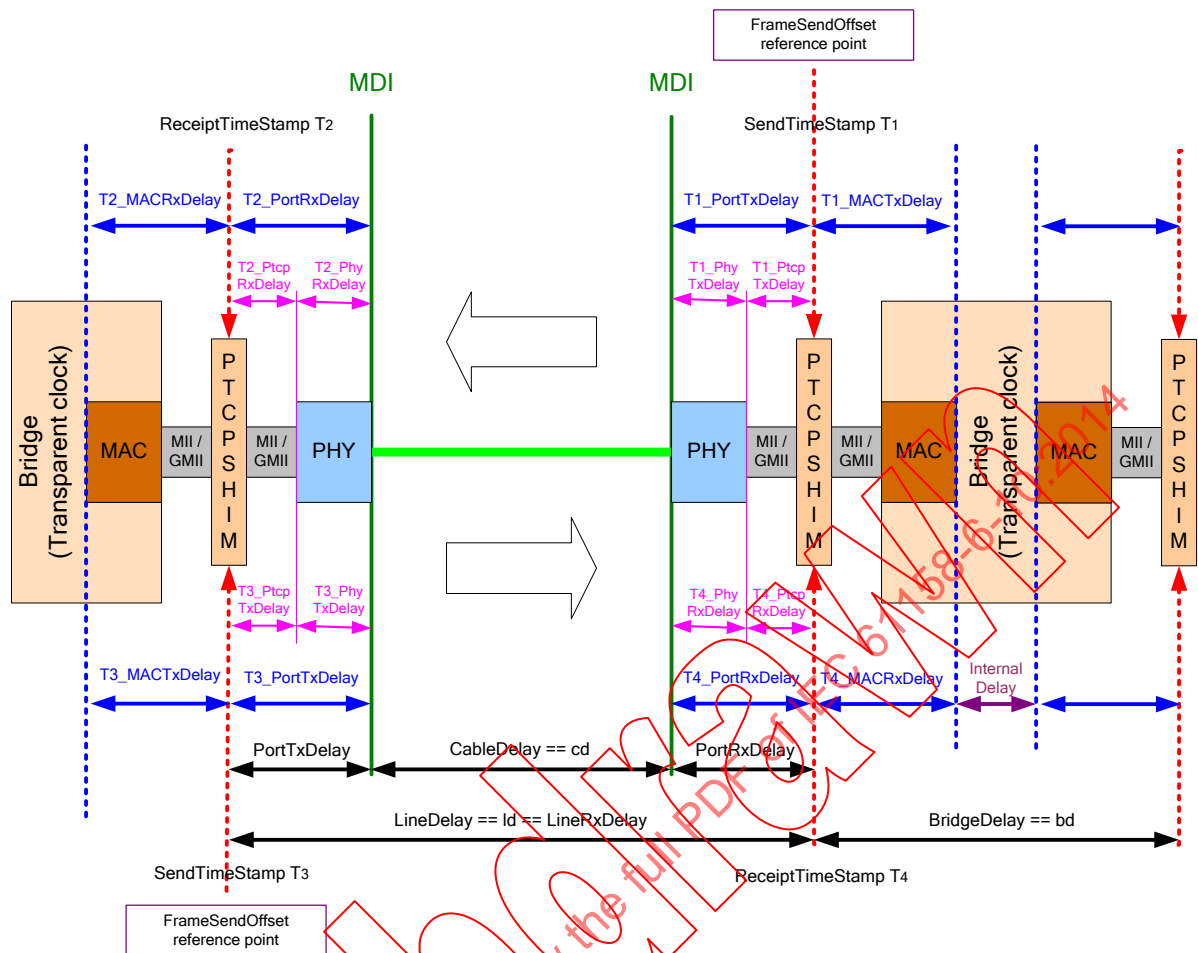


Figure 27 – Line delay measurement

NOTE 1 The Reduced Media Independent Interface (RMII) synchronises its internal 50 MHz clock with the 25 MHz clock of the PHY. It results in a quantization error of 20 ns. This jitter cannot be compensated. The Media Independent Interface (MII) is using the PHY clock without this error.

NOTE 2 The cable type Class D has a maximal delay skew of 45 ns for 100 m cable length. It results in a 45 ns offset error which cannot be compensated. A selection of better cable reduces this error.

A port receive/transmit delay (example: T1_PortTxDelay) contains the line delay of the used PHY component and the deviation from the reference time stamp.

$$T1\_PortTxDelay = T1\_PhyTxDelay + T1\_PtcpTxDelay \quad (21)$$

where

T1_PortTxDelay	is the port transmit delay
T1_PhyTxDelay	is the delay of the used PHY component
T1_PtcpTxDelay	is the deviation from the reference time stamp

$$T3\_PortTxDelay = T3\_PhyTxDelay + T3\_PtcpTxDelay \quad (22)$$

where

T3_PortTxDelay	is the port transmit delay
T3_PhyTxDelay	is the delay of the used PHY component
T3_PtcpTxDelay	is the deviation from the reference time stamp

$$T2\_PortRxDelay = T2\_PhyRxDelay + T2\_PtcpRxDelay \quad (23)$$

where

$T2\_PortRxDelay$	is the port receive delay
$T2\_PhyRxDelay$	is the delay of the used PHY component
$T2\_PtcpRxDelay$	is the deviation from the reference time stamp

$$T4\_PortRxDelay = T4\_PhyRxDelay + T4\_PtcpRxDelay \quad (24)$$

where

$T4\_PortRxDelay$	is the port receive delay
$T4\_PhyRxDelay$	is the delay of the used PHY component
$T4\_PtcpRxDelay$	is the deviation from the reference time stamp

$$CableDelay = (ReqDelay - ResDelay_{peer} - T1\_PortTxDelay - T2\_PortRxDelay - T3\_PortTxDelay - T4\_PortRxDelay) / 2 \quad (25)$$

where

$CableDelay$	is the delay of the cable
$ReqDelay$	is the delay requestor
$ResDelay_{peer}$	is the delay responder
$T1\_PortTxDelay$	is the port transmit delay
$T2\_PortRxDelay$	is the port receive delay
$T3\_PortTxDelay$	is the port transmit delay
$T4\_PortRxDelay$	is the port receive delay

$$LineDelay = CableDelay + T3\_PortTxDelay + T4\_PortRxDelay \quad (26)$$

where

$LineDelay$	is the delay between both PTCP SHIMs for received frames
$CableDelay$	is the delay of the cable
$T3\_PortTxDelay$	is the port receive/transmit delay
$T4\_PortRxDelay$	is the port receive/transmit delay

$$LineDelay_{SyncMaster} = LineDelay \times RCF_{SyncMaster} \quad (27)$$

where

$LineDelay_{SyncMaster}$	is the compensated time needed to forward a frame
$LineDelay$	is the measured value between two peers
$RCF_{SyncMaster}$	is the factor between the frequency of the SyncMaster and the frequency of the SyncSlave

NOTE 3 Formula (27) is only informative if the $LineDelay$ is measured with the synchronized timer.

A $DelayRes.req$ with a $ResDelay$ value of zero indicates that the $CableDelay$ and the $LineDelay$ are not measurable, but the responding peer wants to receive the Sync-Frame.

$LineSyncDelay$ is the line delay of the Sync-Frame. Due to the asymmetry of the involved communication path and due to the resolution of the time stamps it is basically impossible to determine the line delay exactly.

A PTCP clock can detect non PTCP neighbors by checking the line delay between nodes. As a switch delay is in the range of several (>2) μs , the delay between two nodes connected with 100 m Twisted Pair cabling (100 Base TX) is about 500 ns.

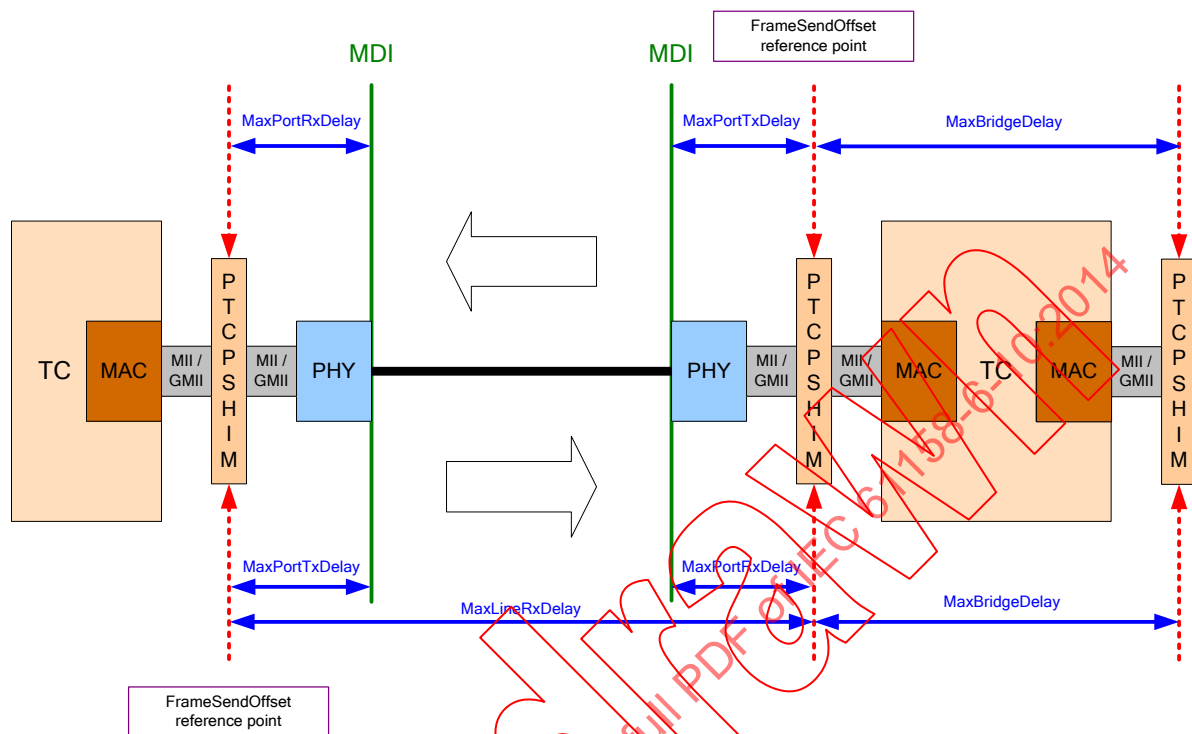


Figure 28 – Model parameter for GSDML usage

Figure 28 shows the model parameter for GSDML usage. The parameter MaxBridgeDelay, MaxPortTxDelay and MaxPortRxDelay offers engineering tools a calculation basis for RT_CLASS_3.

The to be used MaxBridgeDelay depends on the bridging mode like normal forwarding, fast forwarding and DFP concatenated forwarding.

4.4.4.1.6 Sync-Frame forwarding

A device which forwards PTCP sync messages measures the bridge delay as shown in Figure 29 and adds this time to the delay field in the sync or follow up message.

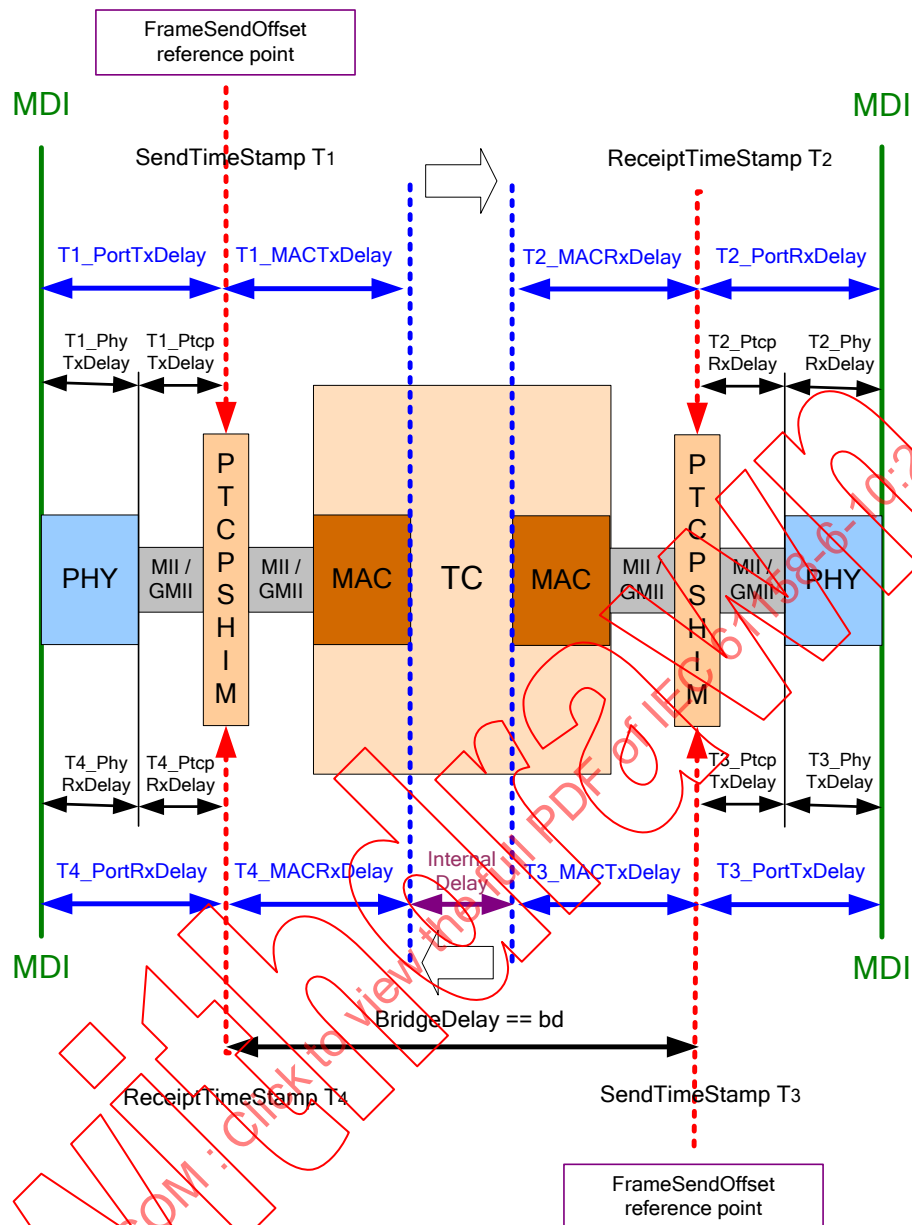


Figure 29 – Bridge delay measurement

The calculation of the bridge delay is shown in following formulae.

$$\text{BridgeDelay} = T_3 - T_4 \quad (28)$$

where

BridgeDelay	is the time needed to forward a frame
T3	is the point in time when the first bit of the SD of a frame is transmitted
T4	is the point in time when the first bit of the SD of a frame is received

$$\text{BridgeDelay}_{\text{SyncMaster}} = \text{BridgeDelay} \times \text{RCF}_{\text{SyncMaster}} \quad (29)$$

where

$\text{BridgeDelay}_{\text{SyncMaster}}$	is the compensated time needed to forward a frame
BridgeDelay	is the time needed to forward a frame

$RCF_{SyncMaster}$ is the factor between the frequency of the SyncMaster and the frequency of the SyncSlave

NOTE Formula (29) is only informative if the LineDelay is measured with the syntonized timer.

$$RCF_{SyncMaster} = \Delta t_{SyncMaster} / \Delta t_{SyncSlave} \quad (30)$$

where

$RCF_{SyncMaster}$ is the ratio between the clock of the SyncMaster and the SyncSlave calculated by the SyncSlave

$\Delta t_{SyncMaster}$ is a time interval of at least 200 ms from the SyncMaster point of view

$\Delta t_{SyncSlave}$ is a time interval measured at the SyncSlave for the given $\Delta t_{SyncMaster}$

The principle of synchronization through a sequence of devices can be seen in Figure 30. The node emitting the time frames for synchronization in the network is known as PTCP master. All other nodes that receive and/or pass on these frames are known as PTCP slave. A PTCP master uses a reference to a local time system or a global time source.

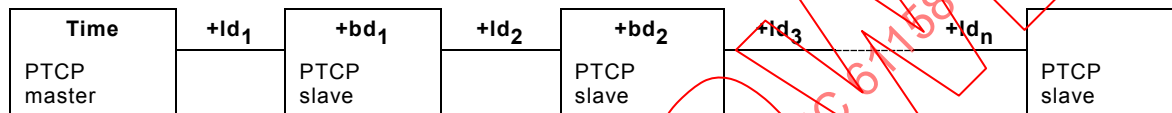


Figure 30 – Delay accumulation

The Sync-Frames are forwarded as peer to peer frames. Every device shall adjust the Sync-Frame by

- adding its bridge delay bd_x (bridge delay measurement is shown in Figure 29)
- adding the line delay ld_x (line delay measurement is shown in Figure 27),

to the delay field of the Sync-Frame as it passes through. Each PTCP slave knows the propagation delay of the sync frame. When the time of arrival T_2 is known, the drift compared to the PTCP master can also be determined.

4.4.4.1.7 Rate compensation

4.4.4.1.7.1 Bridge delay

The bridge delay should be measured with a timer syntonized / rate compensated to the SyncMaster in order to achieve high precision synchronization.

4.4.4.1.7.2 Line delay measurement

The line delay should be measured with a timer syntonized / rate compensated to the SyncMaster in order to achieve high precision synchronization.

4.4.4.1.8 Time deviation measurement

To achieve the measurement of the deviation, see Figure 31, hardware signaling is necessary. For every supported PTCP_SyncID a signal shall be generated. It may only be accessible in test lab environment.

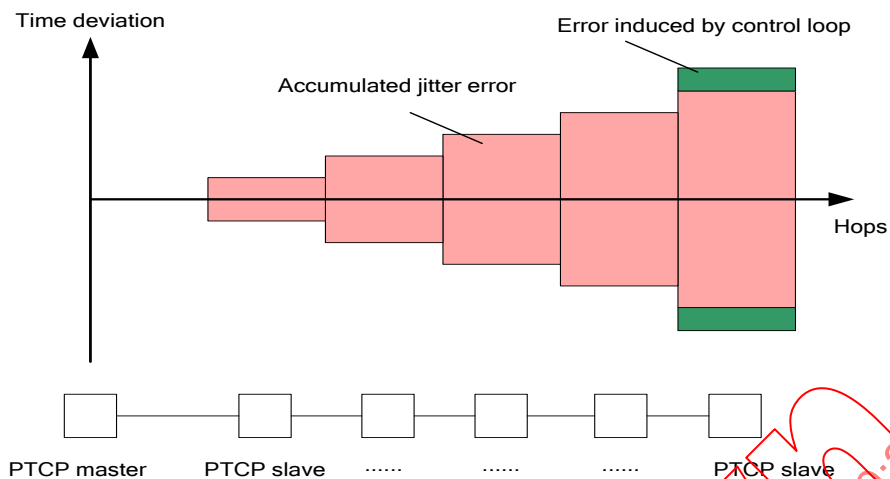


Figure 31 – Worst case accumulated time deviation of synchronization

Figure 32 and Figure 33 show in principle the measurement of the deviation.

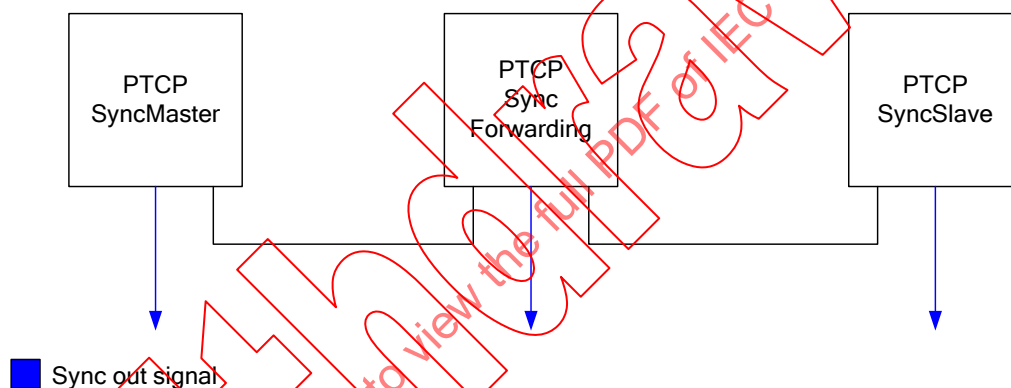


Figure 32 – Scheme for measurement of deviation

Every cycle as shown in Figure 21 shall be signaled for clock synchronization. Every second shall be signaled for time synchronization.

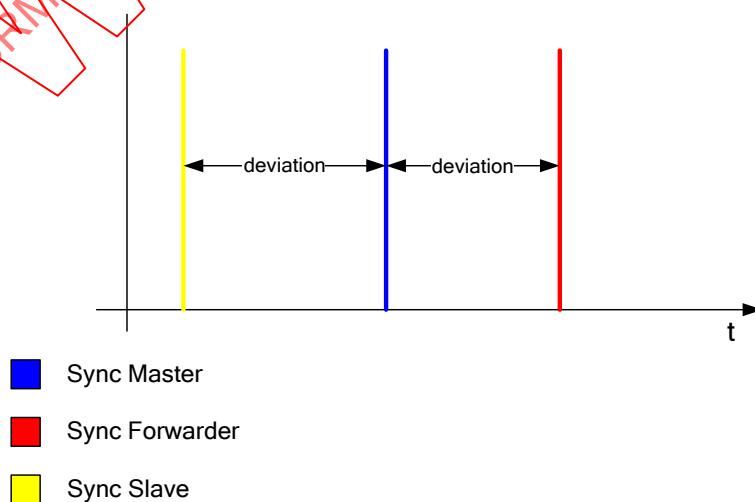


Figure 33 – Measurement of deviation

4.4.4.1.9 Synchronization Protocol

4.4.4.1.9.1 General

PTCP masters are sending Sync-Frames periodically. The sync frame has an original sending time (T_{ORG}).

A PTCP master

- sets the PTCPTime field of the sync frame to T_{ORG} , and
- sets the PTCP_Delay fields of the sync frame to its internal handling time; i.e. the time which passed between T_{ORG} and sending the sync frame.

A PTCP slave adds to the content of the corresponding PTCP_Delay field in the PTCP-Header-Sync the following terms:

- the line delay (T_{LD}) of the port where the sync frame was received, and
- the bridge delay (T_{BD}); i.e. the time to forward the sync frame.

Special case: If the PTCP master or some PTCP slave is not able to directly add the delay times into the sync frame PTCP_Delay fields, it shall generate a so called Follow-Up frame (FU) and add there its bridge delay to the PTCP_Delay field. The line delay is always added to the PTCP_Delay fields of the sync frame.

All times are always related to the measurement time stamp point; see Figure 22 for details.

Each PTCP slave compares its local clock with the sum of PTCPTime, PTCP_Delay out of the sync frame and (if applicable) PTCP_Delay out of the Follow-Up frame. The offset between T_{Slave} and T_{Master} can thus be determined. The PTCP slave follows the time of the PTCP master.

4.4.4.1.9.2 PTCP Master Synchronization with Sync-Frame

PTCP masters are sending Sync-Frames periodically. The synchronization scheme as described in Figure 34 requires transferring the exact sending time of the Sync-Frame as initial value of the delay.

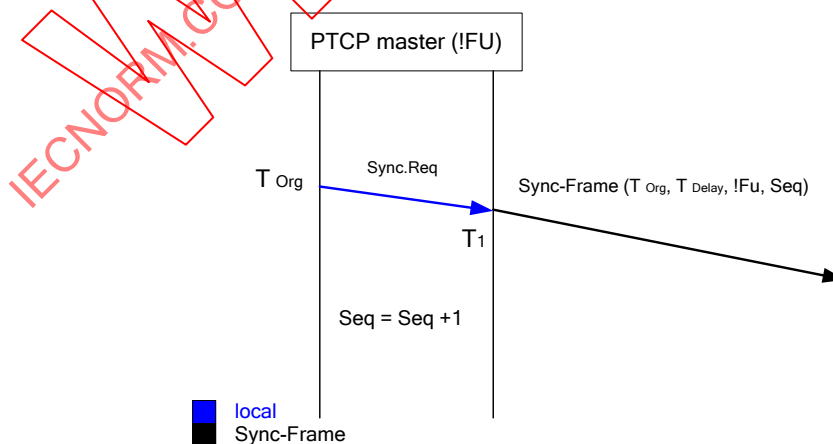


Figure 34 – PTCP master sending Sync-Frame without Follow Up-Frame

$$T_{Delay} = T_1 - T_{Org}$$

where

T_{Delay}	is the internal handling time of the sync frame within the PTCP master; it is the value of the PTCP_Delay field within the sync frame send by the PTCP Master
T_1	is the time stamp of sending the Sync-Frame by the PTCP master
T_{Org}	is the original time of the PTCP master clock

4.4.4.1.9.3 PTCP Master Synchronization with Sync- and FollowUp-Frame

Figure 35 shows a PTCP master which cannot modify a frame before transmission. A PTCP master who is not able to enter the internal handling time into the Sync-Frame directly will use a FollowUp-Frame.

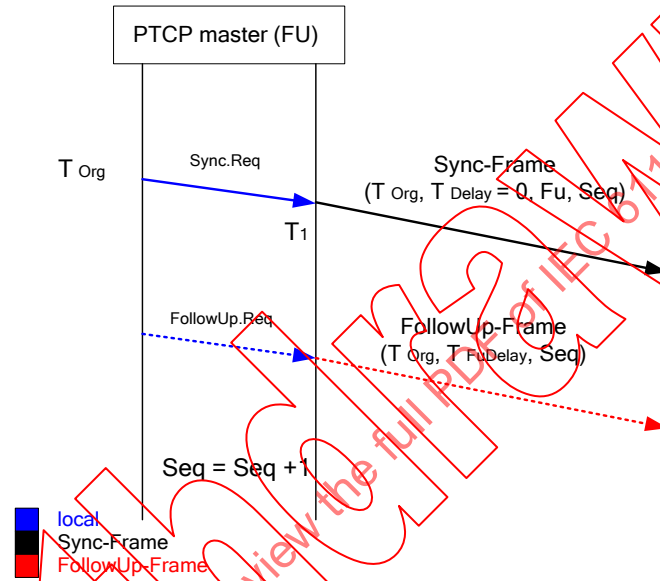


Figure 35 – PTCP master sending Sync-Frame with FollowUp-Frame

The delay field in the Sync-Frame shall be set to 0 by the PTCP master. The delay (T_{FuDelay}) in the associated FollowUp-Frame contains the initial value of the sync frame's delay.

$$T_{\text{FuDelay}} = T_1 - T_{\text{Org}} \quad (32)$$

where

T_{FuDelay}	is the internal handling time of the sync frame within the PTCP master; it is the value of the PTCP_Delay field within the FollowUp frame send by the PTCP Master.
T_{Org}	is the original time of the PTCP master clock
T_1	is the time stamp of sending the Sync-Frame by the PTCP master

4.4.4.1.9.4 PTCP Slave Synchronization (!FU)

A PTCP slave as shown in Figure 36 that can modify a frame before transmission (!FU node) adds its internal bridge delay and the line delay to the delay value in the Sync-Frame as it passes through.

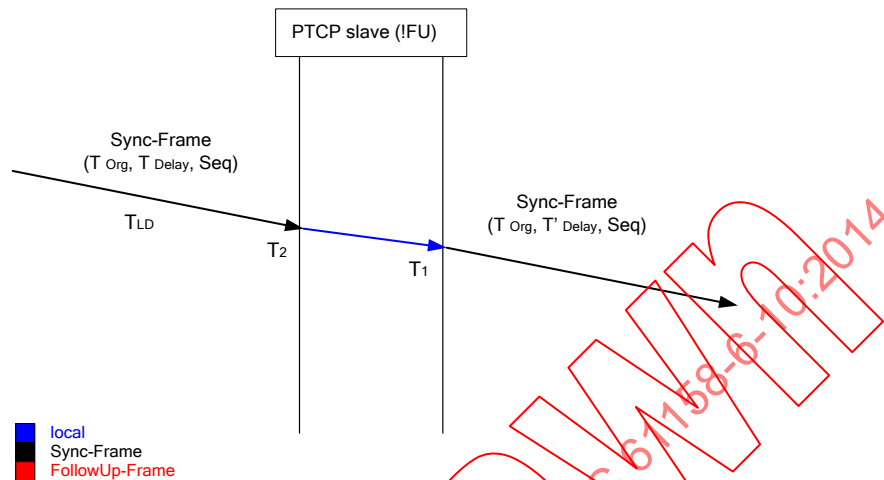


Figure 36 – !FU Sync Slave Forwarding Sync-Frame

$$T'_{\text{Delay}} = T_{\text{Delay}} + T_{\text{LD}} + (T_1 - T_2) \quad (33)$$

where

T'_{Delay}	is the value of the PTCP_Delay field within the sync frame send by the PTCP slave
T_{Delay}	is the delay time as being received by the PTCP slave
T_{LD}	is the line delay time of the receiving port
T_1	is the send time stamp of the Sync-Frame by the PTCP slave
T_2	is the receive time stamp of the Sync-Frame by the PTCP slave

If the sync frame has a FollowUp frame, it is forwarded without any modification by a !FU node.

4.4.4.1.9.5 PTCP Slave Synchronization (FU)

Some implementations of IEEE 802.3 do not allow modifying a frame during transmission.

Such as PTCP slave (FU-Node) still adds the line delay to the delay value in the sync frame, but adds its internal bridge delay to the delay value in the FollowUp-Frame.

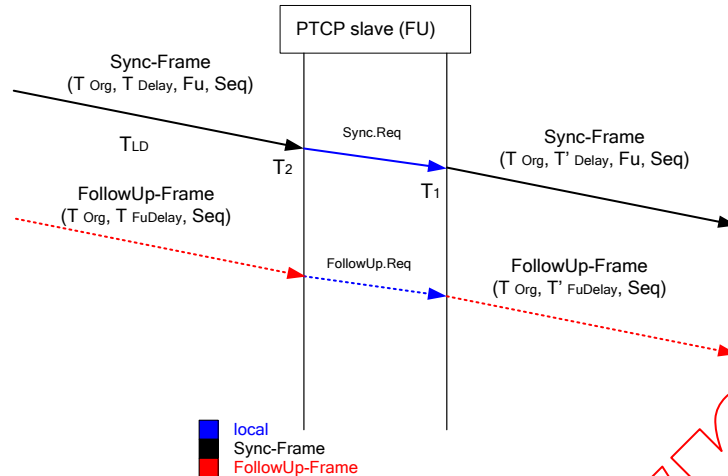


Figure 37 – FU Sync Slave Forwarding Sync- and FollowUp-Frame

$$T'_{\text{Delay}} = T_{\text{Delay}} + T_{\text{LD}} \quad (34)$$

where

T'_{Delay} is the value of the PTCP_Delay field within the sync frame send by the PTCP slave
 T_{Delay} is the delay time within the sync frame as being received by the PTCP slave
 T_{LD} is the line delay of the receiving port

$$T'_{\text{FuDelay}} = T_{\text{FuDelay}} + (T_1 - T_2) \quad (35)$$

where

T'_{FuDelay} is the value of the PTCP_Delay field within the FollowUp frame send by the PTCP slave
 T_{FuDelay} is the delay time within the FollowUp frame as being received by the PTCP slave
 T_{LD} is the line delay of the receiving port
 T_1 is the send time stamp of the Sync-Frame by the PTCP slave
 T_2 is the receive time stamp of the Sync-Frame by the PTCP slave

If no FollowUp frame exists it is generated by an FU PTCP slave. This principle is shown in Figure 38.

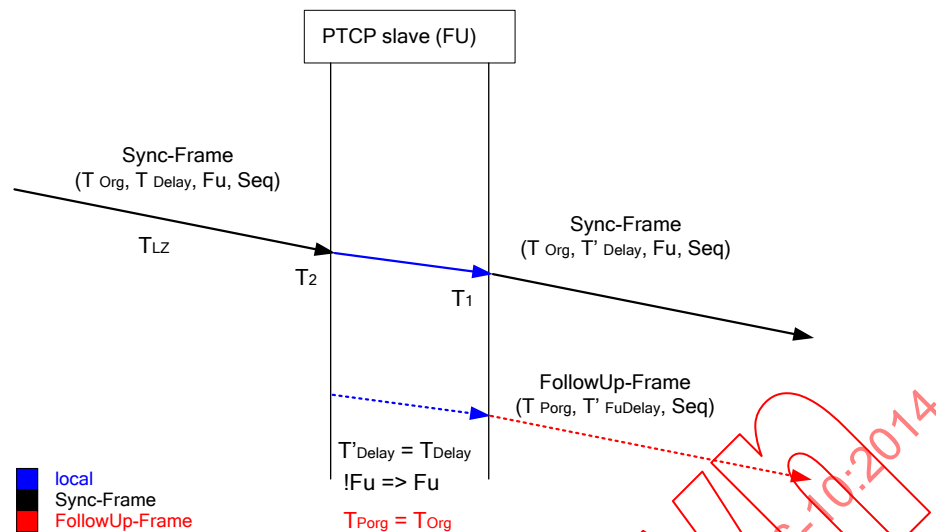


Figure 38 – FU Sync Slave Forwarding Sync- and Generating FollowUp-Frame

$$T'_{\text{Delay}} = T_{\text{Delay}} + T_{\text{LD}} \quad (36)$$

where

T'_{Delay} is the value of the PTCP_Delay field within the sync frame send by the PTCP slave
 T_{Delay} is the delay time within the sync frame as being received by the PTCP slave
 T_{LD} is the line delay of the receiving port

$$T'_{\text{FuDelay}} = (T_1 - T_2) \quad (37)$$

where

T'_{FuDelay} is the value of the PTCP_Delay field within the FollowUp frame send by the PTCP slave
 T_{LD} is the line delay of the receiving port
 T_1 is the send time stamp of the Sync-Frame by the PTCP slave
 T_2 is the receive time stamp of the Sync-Frame by the PTCP slave

4.4.4.1.10 Error recovery

A link down will delete the line delay. A line delay measurement error will stop the forwarding of all sync messages.

To deal with redundancy switchover in case of redundant paths, a Sync-Frame shall be forward only if the sequence number difference to the previous forwarded message is positive. This prevents duplicates which may corrupt the follow up sequence.

The use of an alternate path is possible as long as there is a valid line delay measurement available.

There are 2 timeouts:

- Delay measurement:
Allowed time between delay request and delay response
- Monitoring of sync:
Allowed time between two subsequent sync frames

The timeout of the line delay measurement shall be higher as the timeout of the time master and is calculated independently at every update of the line delay.

For resource limitations the size of the master data base can be limited. Two masters shall be possible at any time. This is possible as the BMA algorithm will cancel the activities of a Sync master.

The line delay values can be restricted for some technologies (e.g. 100 Base TX have delay values below 1 μ s). Error shall be reported but the reaction is beyond the scope of this specification.

4.4.4.2 Line delay measurement

4.4.4.2.1 Line Delay Request Protocol Machine

4.4.4.2.1.1 General

4.4.4.2.1.1.1 General

Figure 39 shows that the line delay is monitored by the LineT and the DelayT timers. Each time the DelayT expires a line delay measurement starts. If the LineT expires before the line delay measurement is done, an error counter is increased.

If the current measurement cycle (5 times) is finished a break for DelayT (long) is made.

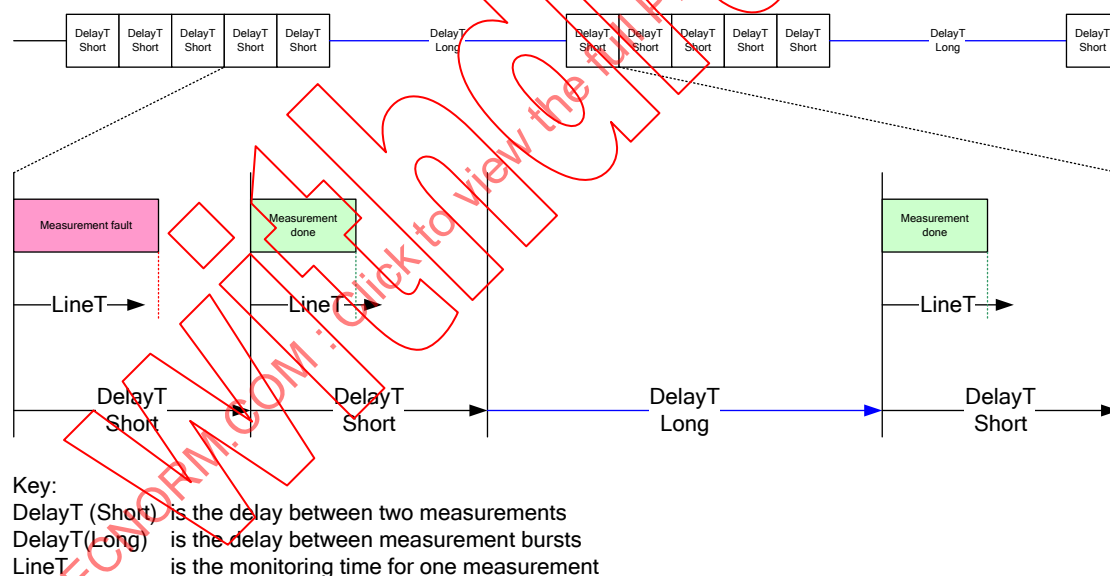


Figure 39 – Principle of the monitoring of the line delay measurement

LineT may be substituted by the checking of the ResDelay value in combination with the DelayT(Short).

4.4.4.2.1.2 Primitive definitions

4.4.4.2.1.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Line delay request Protocol Machine (DELAY_REQ) are described in the service definition and shown in Table 126.

Table 126 – Remote primitives issued or received by DELAY_REQ

Primitive	Source	Destination	Associated parameters	Functions
MAUType_Change_ind	IEEE802.3	DELAY_REQ	PortID, MAU_Type, LINK_Status	—
LMPM_P_Data.ind	LMPM	DELAY_REQ	CREP, S_Port, Tstamp, DA, SA, A_SDU	The LMPM indicates a received delay response or delay followup response.
LMPM_P_Data.cnf ()	LMPM	DELAY_REQ	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.req	DELAY_REQ	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	The DELAY_REQ issues a delay request.

4.4.4.2.1.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DELAY_REQ are described in the service definition and shown in Table 127.

Table 127 – Local primitives issued or received by DELAY_REQ

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.4.4.2.1.3 State transition diagram

The state transition diagram of the DELAY_REQ is shown in Figure 40.

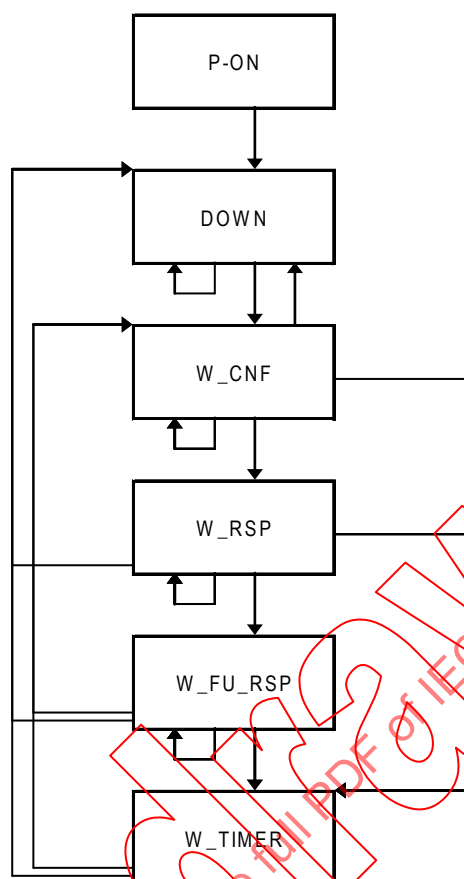


Figure 40 – State transition diagram of DELAY_REQ

States of the DELAY_REQ

P-ON	Initialization of local data.
DOWN	Link of port is down. The line delay measurement shall be initiated after link up by sending a delay request frame.
W_CNF	Wait for confirmation of delay request and store the transmit timestamp of the delay request message.
W_RSP	Wait for delay response frame from the DelayResponder.
W_FU_RSP	Wait for delay response follow up frame and calculate line delay.
W_TIMER	Wait for timeout to repeat the line delay measurement.

4.4.4.2.1.4 State machine description

The line delay measurement shall be initiated by each port and should be repeated in intervals of 8 s. The state machine which sends a delay request frame is called DelayRequestor.

4.4.4.2.1.5 DELAY_REQ state table

Table 128 contains the state table used by DELAY_REQ.

Table 128 – DELAY_REQ state table

#	Current State	Event /Condition =>Action	Next State
1	P-ON	=> Nrepeat := 0 SequenceID := 0 ErrorCounter := 0 RESET_RCF_PEER RESET_LINE_DELAY	DOWN
2	DOWN	MAUType_Change_ind () /!LINK_OK => ignore	DOWN
3	DOWN	MAUType_Change_ind () /!LINK_OK => A_PDU := PTCP_DELAY_REQ_PDU StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
4	W_CNF	DelayReq_Cnf () /Status == !OK => SequenceID++	W_TIMER
5	W_CNF	DelayReq_Cnf () /Status == OK => Tx_Tstamp_T1 := Tstamp	W_RSP
6	W_CNF	MAUType_Change_ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
7	W_CNF	MAUType_Change_ind () /!LINK_OK => ignore	W_CNF
8	W_CNF	TimerExpired (DelayT) => A_PDU := PTCP_DELAY_REQ_PDU SequenceID++ StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
9	W_CNF	TimerExpired (LineT) /MAX_ERROR_COUNT > ErrorCounter => ErrorCounter++	W_TIMER
10	W_CNF	TimerExpired (LineT) /MAX_ERROR_COUNT <= ErrorCounter => ErrorCounter := 0 RESET_LINE_DELAY	W_TIMER

#	Current State	Event /Condition =>Action	Next State
11	W_RSP	DelayResp_Ind () /!CHECK_DELAY_RSP_VALID => ignore	W_RSP
12	W_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID && CHECK_DELAY_RSP_WITH_FU_RSP => Rx_Tstamp2_T2 := Rx_Tstamp1_T2 Rx_Tstamp1_T2 := PTCP_DELAY_RES_PDU.T2TimeStamp Delay := PTCP_DELAY_RES_PDU.Delay Rx_Tstamp_T4 := Tstamp if (Nrepeat > 0) CALC_RCF_PEER	W_FU_RSP
13	W_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID && !CHECK_DELAY_RSP_WITH_FU_RSP && CHECK_DELAY_VALID => Rx_Tstamp2_T2 := Rx_Tstamp1_T2 Rx_Tstamp1_T2 := PTCP_DELAY_RES_PDU.T2TimeStamp Rx_Tstamp_T4 := Tstamp Delay := PTCP_DELAY_RES_PDU.Delay if (Nrepeat > 0) CALC_RCF_PEER CALC_LINE_DELAY Nrepeat++ ErrorCounter := 0	W_TIMER
14	W_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID && !CHECK_DELAY_RSP_WITH_FU_RSP && !CHECK_DELAY_VALID => SET_DUMMY_LINE_DELAY StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL)	W_TIMER
15	W_RSP	DelayFuResp_Ind () => ignore	W_RSP
16	W_RSP	MAUType_Change_ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
17	W_RSP	MAUType_Change_ind () /LINK_OK => ignore	W_RSP
18	W_RSP	TimerExpired (DelayT) => SequenceID++ Nrepeat := 0 RESET_RCF_PEER StartTimer (DelayT, DELAY_REQ_LONG_INTERVAL)	W_TIMER
19	W_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT > ErrorCounter => ErrorCounter++	W_TIMER

#	Current State	Event /Condition =>Action	Next State
20	W_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT <= ErrorCounter => ErrorCounter := 0 RESET_LINE_DELAY	W_TIMER
21	W_FU_RSP	DelayResp_Ind () /!CHECK_DELAY_RSP_VALID => ignore	W_FU_RSP
22	W_FU_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID => Nrepeat := 0 StopTimer (DelayT) StartTimer (DelayT, DELAY_REQ_LONG_INTERVAL) RESET_LINE_DELAY	W_TIMER
23	W_FU_RSP	DelayFuResp_Ind () /!CHECK_DELAY_FU_RSP_VALID => ignore	W_FU_RSP
24	W_FU_RSP	DelayFuResp_Ind () /CHECK_DELAY_FU_RSP_VALID => Delay := Delay + PTCP_DELAY_FURES_PDU.Delay if (Nrepeat > 0) CALC_LINE_DELAY Nrepeat++ ErrorCounter := 0	W_TIMER
25	W_FU_RSP	MAUType_Change_Ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
26	W_FU_RSP	MAUType_Change_ind () /!LINK_OK => ignore	W_FU_RSP
27	W_FU_RSP	TimerExpired (DelayT) => A_PDU := PTCP_DELAY_REQ_PDU SequenceID++ StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
28	W_FU_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT > ErrorCounter => ErrorCounter++	W_TIMER
29	W_FU_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT <= ErrorCounter => ErrorCounter := 0 RESET_LINE_DELAY	W_TIMER

#	Current State	Event /Condition =>Action	Next State
30	W_TIMER	DelayResp_Ind () => ignore	W_TIMER
31	W_TIMER	DelayFuResp_Ind () => ignore	W_TIMER
32	W_TIMER	MAUType_Change_ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
33	W_TIMER	MAUType_Change_ind () /LINK_OK => ignore	W_TIMER
34	W_TIMER	TimerExpired (DelayT) /Nrepeat < MAX_DELAY_REQ_REPEAT => A_PDU := PTCP_DELAY_REQ_PDU SequenceID++ StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
35	W_TIMER	TimerExpired (DelayT) /Nrepeat >= MAX_DELAY_REQ_REPEAT => Nrepeat := 0 RESET_RCF_PEER StartTimer (DelayT, DELAY_REQ_LONG_INTERVAL)	W_TIMER
36	W_TIMER	TimerExpired (LineT) => ignore	W_TIMER

4.4.4.2.1.6 Parameters

4.4.4.2.1.7 Functions, Macros, Timers and Variables

Table 129 contains the functions, macros, timers and variables used by the DELAY_REQ and their arguments and their descriptions.

Table 129 – Functions, macros, timers and variables used by the DELAY_REQ

Name	Type	Function/Meaning
CALC_LINE_DELAY	Function	Line delay for the arithmetical average value of at least the last 7 delay measurements
CALC_RCF_PEER	Function	Calculation of the frequency deviation to the neighbor
CHECK_DELAY_RSP_VALID	Function	Check whether the delay response is valid according to the coding rules
CHECK_DELAY_RSP_WITH_FU_RSP	Function	Check if delay response type is with or without delay response followup frame. TRUE := Wait for the delay response followup FALSE := Without delay response followup

Name	Type	Function/Meaning
CHECK_DELAY_VALID	Function	Check parameter of PTCP_DELAY_RSP_PDU TRUE := The line delay measurement is supported. FALSE := „Special case“; the connected node doesn't support line delay measurement but wants to receive the sync messages. This case is signaled by PTCP_DELAY_RSP_PDU.Delay1ns := 0x00
LINK_OK	Function	Check link MAU_Type == FULL_DUPLEX LINK_Status == Up LINK_Status == UP_CLOSED
RESET_LINE_DELAY	Function	Set line delay to zero
RESET_RCF_PEER	Function	Set RCF_PEER to one
SET_DUMMY_LINE_DELAY	Function	Set line delay to one
StartTimer	Function	This local function is used to start or restart a timer
StopTimer	Function	This local function is used to stop a timer
TimerExpired	Function	This local function is issued if a timer expires
DelayFuResp_Ind (S_Port, Tstamp, PTCP_DELAY_FU_RSP_PDU)	Macro	Receive PTCP-PDU according PTCP_DELAY_FU_RSP_PDU LMPM_P_Data.ind (CREP, D_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_DELAY_FU_RSP_PDU := A_SDU without LT and FrameID
DelayReq_Cnf (D_Port, Tstamp, Status)	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status
DelayReq_Req (Port, PTCP_DELAY_REQ_PDU)	Macro	Create PTCP-PDU according PTCP_DELAY_REQ_PDU Assignments: D_Port := Port DA := PTCP_MulticastMACadd for PTCP_DELAY_REQ_PDU SA := local port address PTCP_DELAY_REQ_PDU.SequenceID := SequenceID PTCP_DELAY_REQ_PDU.PortMACAddress := local port address A_SDU := LT, FrameID, PTCP_DELAY_REQ_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
DelayResp_Ind (S_Port, Tstamp, PTCP_DELAY_RSP_PDU)	Macro	Receive PTCP-PDU according PTCP_DELAY_RSP_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_DELAY_RSP_PDU := A_SDU without LT and FrameID
DelayT	Timer	This local timer is used control the start of a line delay measurement
LineT	Timer	This local timer is used to monitor the timely correct reception of a DelayResponse indication
DELAY_REQ_LONG_INTERVAL	Variable	This local variable contains the time of the pause of line delay measurement. Default value: 8 s
DELAY_REQ_SHORT_INTERVAL	Variable	This local variable contains the time between two line delay measurement. Default value: 200 ms
ErrorCounter	Variable	This local variable contains the actual error count.

Name	Type	Function/Meaning
MAX_DELAY_REQ_REPEAT	Variable	This local variable contains the number of line delay measurements between a pause. Default value: 5
MAX_ERROR_COUNT	Variable	This local variable contains the number of allowed erroneous line delay measurements before the line delay is reset. Default value: 3
Rx_Tstamp_T4	Variable	This local variable contains the receive timestamp of the delay response frame.
Rx_Tstamp1_T2	Variable	This local variable contains a receive timestamp of the delay request frame by the delay responder.
Rx_Tstamp2_T2	Variable	This local variable contains a receive timestamp of the delay request frame by the delay responder.
SequenceID	Variable	This local variable contains the sequence number of the delay request message. It shall be incremented with every delay measurement.
TIMEOUT_LINE_DELAY	Variable	This local variable contains the timeout for the line delay measurement response. Default value: 100 ms
Tx_Tstamp_T1	Variable	This local variable contains the transmit timestamp of the delay request frame.

4.4.4.2.2 Line Delay Response Protocol Machine

4.4.4.2.2.1 Primitive definitions

4.4.4.2.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Line delay Response Protocol Machine (DELAY_RSP) are described in the service definition and shown in Table 130.

Table 130 – Remote primitives issued or received by DELAY_RSP

Primitive	Source	Destination	Associated parameters	Functions
MAUType_Change_ind	IEEE802.3	DELAY_RSP	PortID, MAU_Type, LINK_Status	—
LMPM_P_Data.ind	LMPM	DELAY_RSP	CREP, S_Port, Tstamp, DA, SA, A_SDU	—
LMPM_P_Data.cnf ()	LMPM	DELAY_RSP	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.req	DELAY_RSP	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.2.2.1.2 Primitives exchanged between local machines

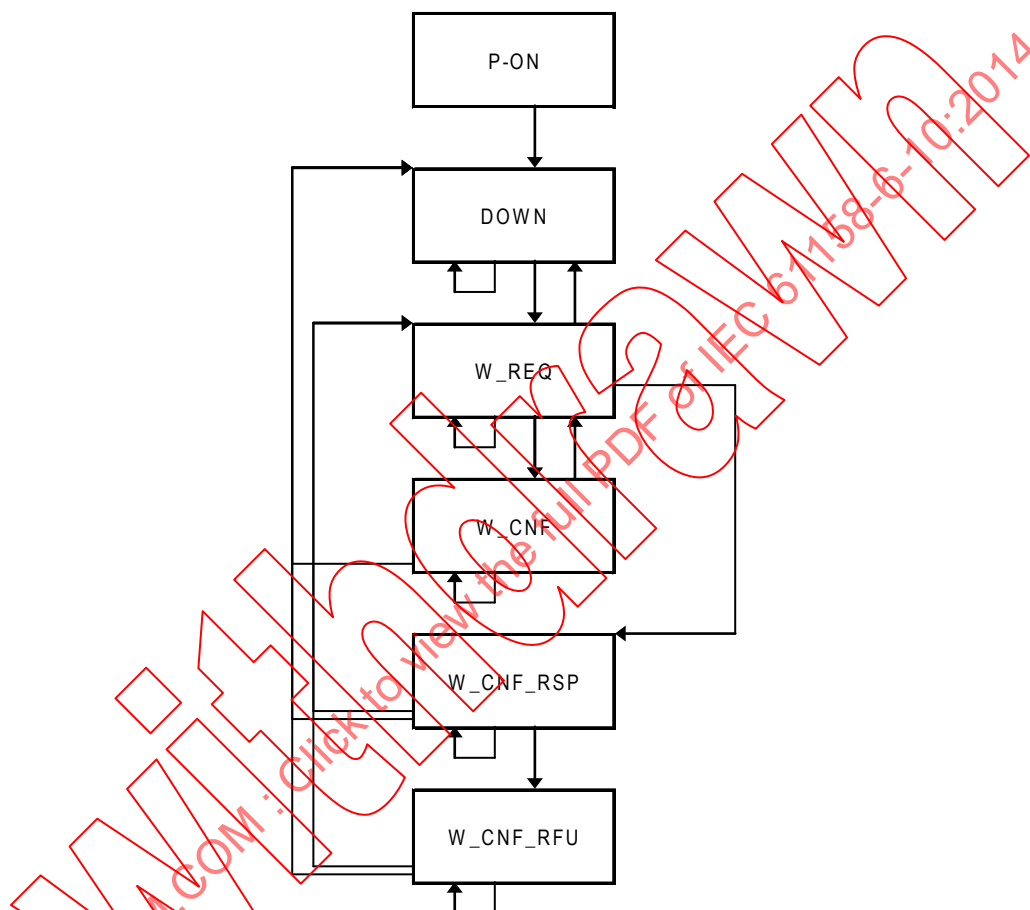
The local service primitives including their associated parameters issued or received by DELAY_RSP are described in the service definition and shown in Table 131.

Table 131 – Local primitives issued or received by DELAY_RSP

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.4.4.2.2 State transition diagram

The state transition diagram of the DELAY_RSP is shown in Figure 41.

**Figure 41 – State transition diagram of DELAY_RSP**

States of the DELAY_RSP

P-ON	Data initialization.
DOWN	Link of port is down. Wait for link up.
W_REQ	If a delay request message is received the receive time stamp will be stored and the DelayResonder sends a delay response message.
W_CNF	Wait for the confirmation of the delay response request.
W_CNF_RSP	Wait for the confirmation of the delay response message, calculate the delay response time and send a delay response follow up message.
W_CNF_RFU	Wait for the confirmation of the delay response follow up message.

4.4.4.2.2.3 State machine description

The DelayResponder state machine handles the responder functionality of the line delay measurement. It receives the line delay measurement request and responses with a line delay measurement response.

The DelayResponder shall respond with the DelayResponse without FollowUp if supported. Otherwise it shall use the DelayResponse with FollowUp.

Each delay request message shall be immediately responded with a delay response message.

4.4.4.2.2.4 DELAY_RSP state table

Table 132 contains the state table used by DELAY_RSP.

Table 132 – DELAY_RSP state table

#	Current State	Event /Condition =>Action	Next State
1	P-ON	=>	DOWN
2	DOWN	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
3	DOWN	MAUType_Change_ind () /LINK_OK => ignore	W_REQ
4	W_REQ	DelayReq_Ind () /FU_RES => store PTCP_DELAY_REQ_PDU A_PDU := PTCP_DELAY_RES_PDU Rx_Tstamp_T2 := Tstamp PTCP_DELAY_RES_PDU.T2TimeStamp := Rx_Tstamp_T2 DelayResp_Req ()	W_CNF_RSP
5	W_REQ	DelayReq_Ind () /!FU_RES => store PTCP_DELAY_REQ_PDU A_PDU := PTCP_DELAY_RES_PDU Rx_Tstamp_T2 := Tstamp PTCP_DELAY_RES_PDU.T2TimeStamp := Rx_Tstamp_T2 DelayResp_Req ()	W_CNF
6	W_REQ	DelayReq_Ind () /!FU_TIMESTAMP => store PTCP_DELAY_REQ_PDU A_PDU := PTCP_DELAY_RES_PDU PTCP_DELAY_RES_PDU.Delay := 0 DelayResp_Req ()	W_CNF
7	W_REQ	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
8	W_REQ	MAUType_Change_ind () /LINK_OK => ignore	W_REQ

#	Current State	Event /Condition =>Action	Next State
9	W_CNF	DelayResp_Cnf () => ignore	W_REQ
10	W_CNF	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
11	W_CNF	MAUType_Change_ind () /LINK_OK => ignore	W_CNF
12	W_CNF_RSP	DelayResp_Cnf () /Status != OK => ignore	W_REQ
13	W_CNF_RSP	DelayResp_Cnf () /Status == OK => Tx_Tstamp_T3 := Tstamp A_PDU := PTCP_DELAY_FU_RES_PDU PTCP_DELAY_FU_RES_PDU.Delay := CALC_RESIDENTIAL_TIME DelayFuResp_Req ()	W_CNF_RFU
14	W_CNF_RSP	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
15	W_CNF_RSP	MAUType_Change_ind () /LINK_OK => ignore	W_CNF_RSP
16	W_CNF_RFU	DelayFuResp_Cnf () => ignore	W_REQ
17	W_CNF_RFU	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
18	W_CNF_RFU	MAUType_Change_ind () /LINK_OK => ignore	W_CNF_RFU

4.4.4.2.2.5 Functions, Macros, Timers and Variables

Table 133 contains the functions, macros, timers and variables used by the DELAY_RSP and their arguments and their descriptions.

Table 133 – Functions, Macros, Timers and Variables used by the DELAY_RSP

Name	Type	Function/Meaning
DelayResp_Req (Port, PTCP_DELAY_RSP_PDU)	Macro	Create PTCP-PDU according PTCP_DELAY_RSP_PDU Assignments: D_Port := Port DA := Multicast-MAC-Address for PTCP_DELAY_RSP_PDU SA := local port address PTCP_DELAY_RSP_PDU.SequenceID := PTCP_DELAY_REQ_PDU.SequenceID PTCP_DELAY_RSP_PDU.PortMACAddress := PTCP_DELAY_REQ_PDU.PortMACAddress PTCP_DELAY_REQ_PDU.T2PortRxDelay := T2_PortRxDelay PTCP_DELAY_REQ_PDU.T3PortTxDelay := T3_PortTxDelay PTCP_DELAY_REQ_PDU.T2TimeStamp := T2_TimeStamp A_SDU := LT, FrameID, PTCP_DELAY_RSP_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
DelayFuResp_Req (Port, ResTime, PTCP_DELAY_FU_RSP_PDU)	Macro	Create PTCP-PDU according PTCP_DELAY_FU_RSP_PDU Assignments: D_Port := Port DA := Multicast-MAC-Address for PTCP_DELAY_FU_RSP_PDU SA := local source address PTCP_Delay := ResTime A_SDU := LT, FrameID, PTCP_DELAY_FU_RSP_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
DelayResp_Cnf (D_Port, Tstamp, Status)	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status
DelayFuResp_Cnf (D_Port, Tstamp, Status)	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_Status) Assignments: Status := LMPM_status
DelayReq_Ind (S_Port, Tstamp, PTCP_DELAY_REQ_PDU)	Macro	Receive PTCP-PDU according PTCP_DELAY_REQ_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_DELAY_REQ_PDU := A_SDU without LT and FrameID
Rx_Tstamp_T2	Variable	This local variable contains the receive time stamp of the delay request message.
Tx_Tstamp_T3	Variable	This local variable contains the transmit time stamp of the delay response message.
LINK_OK	Macro	Check whether the MAU_Type is FULL_DUPLEX and the LINK_Status is UP or BLOCKED.
CALC_RESIDENTIAL_TIME	Macro	Calculate the residential time ResTime := (Tx_Stamp_T3 – Rx_Stamp_T2)

4.4.4.3 Overview for master slave state tables

Figure 42 shows an overview of the interaction between the PTCP state tables. For devices without PTCP support or for PTCP slaves without an active PTCP master, the reference time sink is generated by local means.

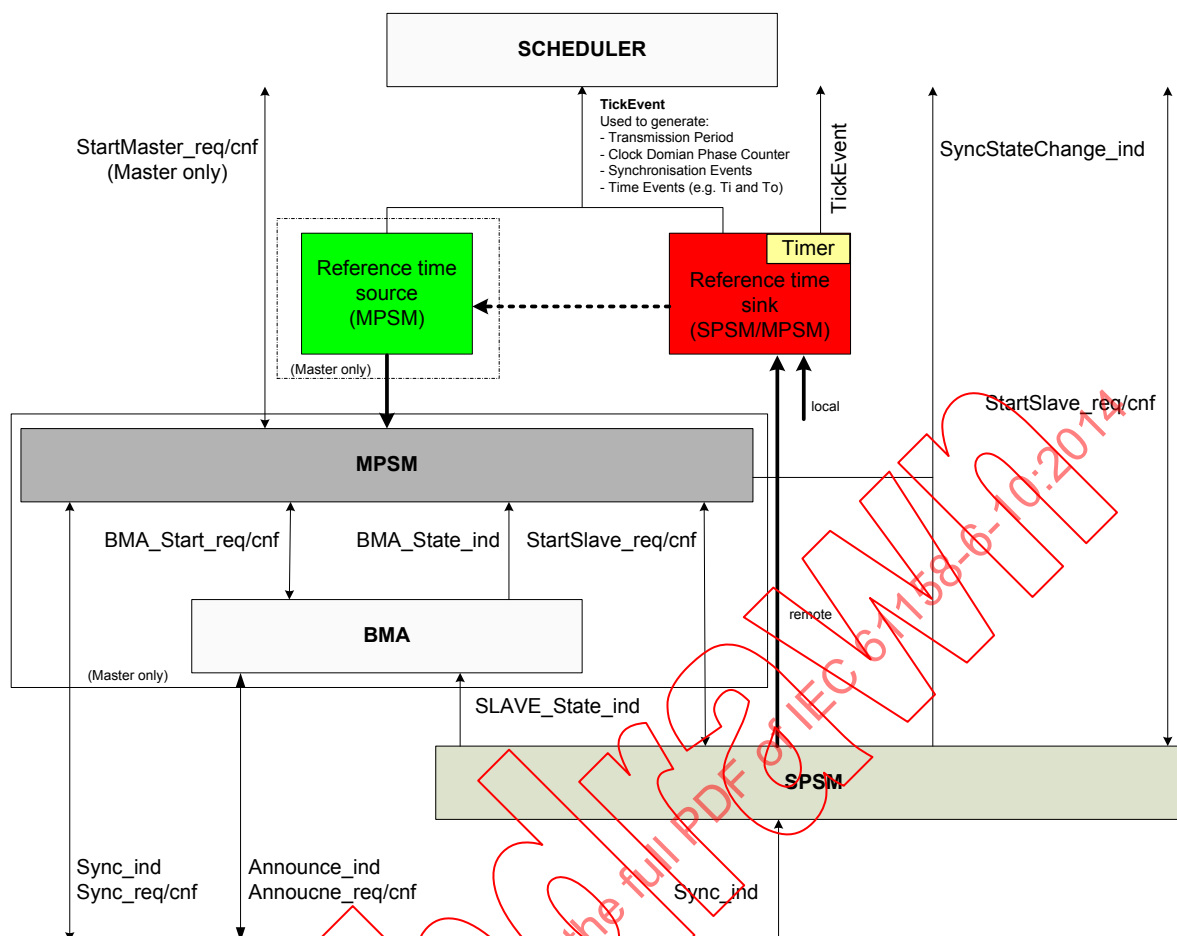


Figure 42 – Overview of PTCP

For the use a PTCP in conjunction with wireless the principles of IEEE 802.1AS should be used.

4.4.4.4 Best-Master-Algorithm

4.4.4.4.1 Best-Master-Algorithm Protocol Machine

4.4.4.4.1.1 Primitive definitions

4.4.4.4.1.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Best-Master-Algorithm Protocol Machine (SYN_BMA) are described in the service definition and shown in Table 134.

Table 134 – Remote primitives issued or received by SYN_BMA

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.req	BMA	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	BMA	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_A_Data.ind	LMPM	BMA	CREP, S_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.4.1.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by SYN_BMA are described in the service definition and shown in Table 135.

Table 135 – Local primitives issued or received by SYN_BMA

Primitive	Source	Destination	Associated parameters	Functions
BMA_Start_req	MPSM	BMA	—	Activate best master function
BMA_Start_cnf (+/-)	BMA	MPSM	—	Confirmation for best master function start
BMA_Stop_req	MPSM	BMA	—	Deactivates best master function
BMA_Stop_cnf (+/-)	BMA	MPSM	—	Confirmation for stop best master function
BMA_State_ind	BMA	MPSM	Role	During the startup period the BMA elects the best sync master. Allowed value for Role: - MASTER - SLAVE
SLAVE_State_ind	SPSM	BMA	State	Indication for master lost. The event is generated by the SLAVE. - MASTER_LOST
Announce_req	BMA	LMPM	PTCP_ANNOUNCE_PDU	Create PTCP-PDU according PTCP_ANNOUNCE_PDU Assignments: D_Port := AUTO DA :=PTCP_MulticastMACadd for PTCP_ANNOUNCE_PDU SA := local source address PTCP_ANNOUNCE_PDU.SequenceID := SequenceID PTCP_ANNOUNCE_PDU.DomainUUID := DomainUUID PTCP_ANNOUNCE_PDU.MasterSourceAddress := local master address PTCP_ANNOUNCE_PDU.MasterPriority1 := Priority PTCP_ANNOUNCE_PDU.ClockClass := Class PTCP_ANNOUNCE_PDU.ClockAccuracy := Accuracy PTCP_ANNOUNCE_PDU.ClockVariance := Variance PTCP_ANNOUNCE_PDU.MasterPriority2 := 0xFF A_SDU := LT, FrameID, PTCP_ANNOUNCE_PDU LMPM_A_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)

Primitive	Source	Destination	Associated parameters	Functions
Announce_cnf (+/-)	LMPM	BMA	State	LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status
Annouce_ind	LMPM	BMA	PTCP_ANNOUNCE_PDU	Receive PTCP-PDU according PTCP_SYNC_PDU LMPM_A_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_SYNC_PDU := A_SDU without LT and FrameID

4.4.4.4.1.2 State transition diagram

The state transition diagram of the SYN_BMA is shown in Figure 43:

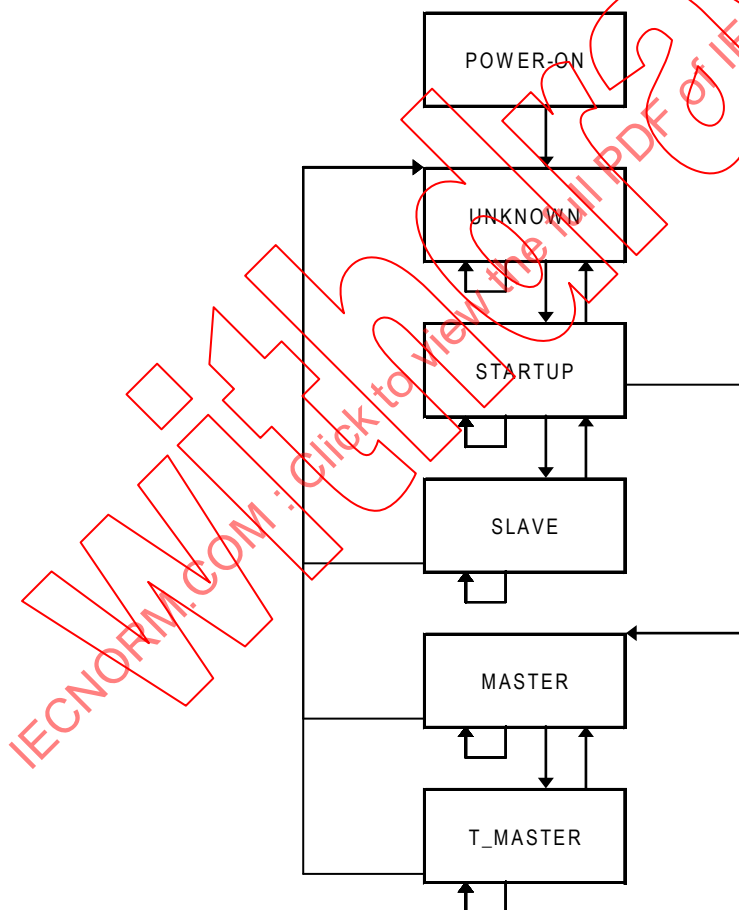


Figure 43 – State transition diagram of SYN_BMA

States of the SYN_BMA

POWER-ON

Data initialization

UNKNOWN

The synchronization role is unknown. A role shall be set via PTCP services.

STARTUP

The role set via PTCP services. The potential master starts

	sending announce messages.
SLAVE	The potential master in state SLAVE shall be synchronized by using the synchronization messages from the active master in the sync domain.
MASTER	At this state the device is, due to its synchronization attributes, elected as the best master.
T_MASTER	This is a transition state which indicates the receiving of announce messages from another potential master in the sync domain.

4.4.4.1.3 State machine description

The BMA-state-machine shall elect the best master from all announced master-capable devices during the start up period. These decisions are made local by every master. An instance of the state machine exists for every PTCP SyncID.

Due to the fact, that not every device is able to be master or should be master, the behavior of the state machine depends on his role. The role is set by PTCP services. Roles are master, secondary master and slave. A master also needs slave functionality.

Whether a device with the role master becomes master or secondary master will be decided during the start up period by the best master algorithm. Every potential master shall send announce messages to take part in master and secondary master election.

The master election will be made in two steps. With the first received announce the potentially master will be stored in a master list. With the next announce the SYN_BMA elects this master, if it's the best known master.

During master competition all potentially masters will be collected in master list. After election they drop out via altering.

If the decision to be master is dropped, the best master starts sending sync messages. The remaining potential masters stop to send announce messages and have to be slaves. If the active master fails, the backup masters restart sending sync and announce messages.

4.4.4.1.4 SYN_BMA state table

Table 136 contains the state table used by SYN_BMA.

Table 136 – SYN_BMA state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> init MasterList SequenceID := 0 BestMaster := FALSE ACTIVE_MASTER_FLAG := FALSE AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	UNKNOWN

#	Current State	Event /Condition =>Action	Next State
2	UNKNOWN	BMA_Start_req () => Status := OK A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) StartTimer (StartupTimer, MasterStartupTime) BMA_Start_cnf (+) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	STARTUP
3	UNKNOWN	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
4	UNKNOWN	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	UNKNOWN
5	STARTUP	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	STARTUP
6	STARTUP	BMA_Stop_req () => StopTimer (AnnTimer) StopTimer (StartupTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
7	STARTUP	Announce_cnf () => ignore	STARTUP
8	STARTUP	SLAVE_State_ind (State) => ignore	STARTUP
9	STARTUP	Announce_ind () /!VALID_SYNC_MASTER => ignore	STARTUP
10	STARTUP	Announce_ind () /!VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST	STARTUP
11	STARTUP	Announce_ind () /!VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY	STARTUP
12	STARTUP	TimerExpired (AnnTimer) /!LOCAL_IS_BEST_MASTER => BestMaster := FALSE StopTimer (StartupTimer) StartTimer (AnnTimer, MasterAnnounceTime) BMA_State_ind (SLAVE)	SLAVE

#	Current State	Event /Condition =>Action	Next State
13	STARTUP	TimerExpired (AnnTimer) /LOCAL_IS_BEST_MASTER => BestMaster := TRUE SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) BMA_State_ind (MASTER) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	STARTUP
14	STARTUP	TimerExpired (StartupTimer) => ACTIVE_MASTER_FLAG := TRUE	MASTER
15	STARTUP	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	STARTUP
16	SLAVE	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	SLAVE
17	SLAVE	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
18	SLAVE	Announce_cnf () => ignore	SLAVE
19	SLAVE	SLAVE_State_ind (State) /State != MASTER_LOST => ignore	SLAVE
20	SLAVE	SLAVE_State_ind (State) /State == MASTER_LOST && !BestMaster => ignore	SLAVE
21	SLAVE	SLAVE_State_ind (State) /State == MASTER_LOST && BestMaster => SequenceID++ A_PDU:= PTCP_ANNOUNCE_PDU StartTimer (StartupTimer, MasterStartupTime) BMA_State_ind (MASTER) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	STARTUP
22	SLAVE	Announce_ind () /!VALID_SYNC_MASTER => ignore	SLAVE
23	SLAVE	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST	SLAVE

#	Current State	Event /Condition =>Action	Next State
24	SLAVE	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY	SLAVE
25	SLAVE	TimerExpired (AnnTimer) /LOCAL_IS_BEST_MASTER => BestMaster := FALSE StartTimer (AnnTimer, MasterAnnounceTime)	SLAVE
26	SLAVE	TimerExpired (AnnTimer) /LOCAL_IS_BEST_MASTER => BestMaster := TRUE StartTimer (AnnTimer, MasterAnnounceTime)	SLAVE
27	SLAVE	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	SLAVE
28	MASTER	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	MASTER
29	MASTER	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
30	MASTER	Announce_cnf () => ignore	MASTER
31	MASTER	SLAVE_State_ind (State) => ignore	MASTER
32	MASTER	Announce_ind () /!VALID_SYNC_MASTER => ignore	MASTER
33	MASTER	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST	MASTER
34	MASTER	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY	MASTER
35	MASTER	TimerExpired (AnnTimer) /!MASTER_LIST_EMPTY => SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) StartTimer (StartupTimer, MasterStartupTime) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	T_MASTER
36	MASTER	TimerExpired (AnnTimer) /MASTER_LIST_EMPTY => StartTimer (AnnTimer, MasterAnnounceTime)	MASTER

#	Current State	Event /Condition =>Action	Next State
37	MASTER	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	MASTER
38	T_MASTER	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	T_MASTER
39	T_MASTER	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
40	T_MASTER	Announce_cnf () => ignore	T_MASTER
41	T_MASTER	SLAVE_State_ind (State) => ignore	T_MASTER
42	T_MASTER	Announce_ind () /!VALID_SYNC_MASTER => ignore	T_MASTER
43	T_MASTER	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST StartTimer (StartupTimer, MasterStartupTime)	T_MASTER
44	T_MASTER	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY StartTimer (StartupTimer, MasterStartupTime)	T_MASTER
45	T_MASTER	TimerExpired (AnnTimer) /!MASTER_LIST_EMPTY => SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) StartTimer (StartupTimer, MasterStartupTime) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	T_MASTER
46	T_MASTER	TimerExpired (AnnTimer) /MASTER_LIST_EMPTY => SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	T_MASTER
47	T_MASTER	TimerExpired (StartupTimer) => ignore	MASTER
48	T_MASTER	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	T_MASTER

4.4.4.1.5 Functions, Macros, Timers and Variables

Table 137 contains the functions, macros, timers and variables used by the SYN_BMA and their arguments and their descriptions.

Table 137 – Functions, Macros, Timers and Variables used by the SYN_BMA

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or restarts a timer
StopTimer	Function	This local function stops a timer
TimerExpired	Function	This local function is issued if a timer expires
ADD_TO_MASTER_LIST	Macro	<pre> Insert sync master in remote sync master list sm_sa := PTCP_ANNOUNCE_PDU.MasterSourceAddress if (MasterList.Num_entry < MAX_CNT_REMOTE_MASTER) MasterList.Insert(sm_sa) MasterList.Entry(sm_sa).Priority1 := PTCP_ANNOUNCE_PDU.Priority1 MasterList.Entry(sm_sa).Accuracy := PTCP_ANNOUNCE_PDU.Accuracy MasterList.Entry(sm_sa).Variance := PTCP_ANNOUNCE_PDU.Variance MasterList.Entry(sm_sa).Priority2 := PTCP_ANNOUNCE_PDU.Priority2 MasterList.Entry(sm_sa).SA := sm_sa MasterList.Entry(sm_sa).Receipt := 1 MasterList.Num_entry++ </pre>
AGING_MASTER_LIST	Macro	<pre> Aging of entries in remote sync master list for m := 0 to MasterList.Num_entry if (MasterList.Entry[m].Receipt == 0) MasterList.Num_entry-- MasterList.Remove(sm_sa) </pre>
Announce_cnf (+/-) (State)	Macro	<pre> LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status </pre>
Announce_ind (PTCP_ANNOUNCE_PDU)	Macro	<pre> Receive PTCP-PDU according PTCP_ANNOUNCE_PDU LMPM_A_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_ANNOUNCE_PDU := A_SDU without LT and FrameID </pre>
Announce_req (Port, PTCP_ANNOUNCE_PDU)	Macro	<pre> Create PTCP-PDU according PTCP_ANNOUNCE_PDU Assignments: D_Port := Port DA :=PTCP_MulticastMACadd for PTCP_ANNOUNCE_PDU SA := local source address PTCP_ANNOUNCE_PDU.SequenceID := SequenceID PTCP_ANNOUNCE_PDU.DomainUUID := DomainUUID PTCP_ANNOUNCE_PDU.MasterSourceAddress := local master address PTCP_ANNOUNCE_PDU.MasterPriority1 := Priority PTCP_ANNOUNCE_PDU.ClockClass := Class PTCP_ANNOUNCE_PDU.ClockAccuracy := Accuracy PTCP_ANNOUNCE_PDU.ClockVariance := Variance PTCP_ANNOUNCE_PDU.MasterPriority2 := 0xFF A_SDU := LT, FrameID, PTCP_ANNOUNCE_PDU LMPM_A_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU) </pre>
CHECK_IN_MASTER_LIST	Macro	<pre> Master in PTCP_SubDomainUUID master list sm_sa := PTCP_ANNOUNCE_PDU.MasterSourceAddress sm_sa in MasterList </pre>
init MasterList	Macro	<pre> initialize data set for remote master MasterList.Num_entry := 0 </pre>
LOCAL_IS_BEST_MASTER	Macro	Executes comparison according 4.4.4.1.6 and return TRUE if the local master is found, else FALSE.

Name	Type	Function/Meaning
TRANSMIT_PORT_VALID	Macro	Table 152 shall be applied.
UPDATE_MASTER_ENTRY	Macro	Update entry in remote sync master list MasterList.Entry{sm_sa}.Priority1 := PTCP_ANNOUNCE_PDU.Priority1 MasterList.Entry{sm_sa}.Accuracy := PTCP_ANNOUNCE_PDU.Accuracy MasterList.Entry{sm_sa}.Variance := PTCP_ANNOUNCE_PDU.Variance MasterList.Entry{sm_sa}.Priority2 := PTCP_ANNOUNCE_PDU.Priority2 MasterList.Entry{sm_sa}.Receipt += 1
VALID_SYNC_MASTER	Macro	Domain is supported PTCP_ANNOUNCE_PDU.DomainUUID == DomainUUID
AnnTimer	Timer	This local timer is used to create the interval for the sending of announce frames
MasterListAgingTimer	Timer	This local timer is used to create the interval for the aging of the entries of the master list.
StartupTimer	Timer	This local timer is used to create the interval for the checking of announce frames.
MasterAgingTime	Variable	Time for the aging of the master list. Default value: „2 x PTCP Takeover Timeout“
MasterAnnounceTime	Variable	An announce frame shall be sent every MasterAnnounceTime until MasterStartupTime. Default value: 100 ms
MasterStartupTime	Variable	Time for the receiving check of announce frames. Default value: 10 s

4.4.4.4.1.6 Finding the Best Sync Master

PTCP contains the support of multiple sync masters. The filtering of the stored sync masters to found the “to be used” sync master is done by the BMA using the keys attributes SyncID and PTCP_SubdomainUUID and the following checking rules:

NOTE 1 The PTCP_ClockClass is not used.

NOTE 2 Lower values take precedence.

1) Checking whether a PTCP_MasterPriority1.Active == TRUE sync master exists

The BMA checks whether an ACTIVE sync master exists. If another sync master is active the checking master changes to the sync slave mode.

If no ACTIVE sync master exists it continuous with the next check.

NOTE 3 If the local sync master is active, it stays active until the application decides otherwise.

2) Find the sync master with the best PTCP_MasterPriority1.Level

The BMA searches for the sync masters with the best value, taking its own value into account.

If more than one sync master exists in residual list then continue with the next check else use the found one.

3) Find the sync master with the best PTCP_MasterPriority1.Priority

The BMA searches for the sync masters with the best value of the residual list.

If more than one sync master exists in residual list then continue with the next check else use the found one.

4) Find the sync master with the best PTCP_MasterPriority2

The BMA searches for the sync masters with the best value of the residual list.

If more than one sync master exists in residual list then continue with the next check else use the found one.

5) Find the sync master with the best PTCP_ClockVariance

The BMA searches for the sync masters with the best value of the residual list.

If more than one sync master exists in residual list then continue with the next check else use the found one.

6) Find the sync master with the best PTCP_ClockAccuracy

The BMA searches for the sync masters with the best value of the residual list.

If more than one sync master exists in residual list then continue with the next check else use the found one.

7) Find the sync master with the best PTCP_MasterSourceAddress

The BMA searches for the sync masters with the lowest value (comparing octet zero to five) of the PTCP_MasterSourceAddress of the residual list.

Use the found one.

NOTE 4 The PTCP_MasterSourceAddress is used as "tie breaker".

4.4.4.5 PTCP Master Protocol Machine

4.4.4.5.1 Primitive definitions

4.4.4.5.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by PTCP Master Protocol Machine (SYN_MPSM) are described in the service definition and shown in Table 138.

Table 138 – Remote primitives issued or received by SYN_MPSM

Primitive	Source	Destination	Associated parameters	Functions
LMPM_P_Data.req	MPSM	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—
LMPM_P_Data.cnf	LMPM	MPSM	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.ind	LMPM	MPSM	CREP, S_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.5.1.1 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by SYN_MPSM are described in the service definition and shown in Table 139.

Table 139 – Local primitives issued or received by SYN_MPSM

Primitive	Source	Destination	Associated parameters	Functions
StartMaster_req	PTCP ASE	MPSM	—	Activate master function
StartMaster_cnf	MPSM	PTCP ASE	Status	Confirmation for master start
StopMaster_req	PTCP ASE	MPSM	—	Deactivates master function
StopMaster_cnf	MPSM	PTCP ASE	Status	Confirmation for master stop
StartSlave_req	MPSM	SPSM	—	Activate slave function
StartSlave_cnf	SPSM	MPSM	—	Confirmation for start slave
StopSlave_req	MPSM	SPSM	—	Deactivates slave function
StopSlave_cnf	SPSM	MPSM	—	Confirmation for stop slave
BMA_Start_req	MPSM	BMA	—	Activate best master function
BMA_Start_cnf	BMA	MPSM	—	Confirmation for best master function start
BMA_Stop_req	MPSM	BMA	—	Deactivates best master function
BMA_Stop_cnf	BMA	MPSM	—	Confirmation for stop best master function
BMA_State_ind	BMA	MPSM	Role	During the startup period the BMA elects the best sync master. Allowed value for Role: - MASTER - SLAVE
SyncStateChange_ind	MSPM	FSPM	Status	Indication for events - SINGLE_MASTER - MULTIPLE_MASTER

4.4.4.5.2 State transition diagram

The state transition diagram of the SYN_MPSM is shown in Figure 44:

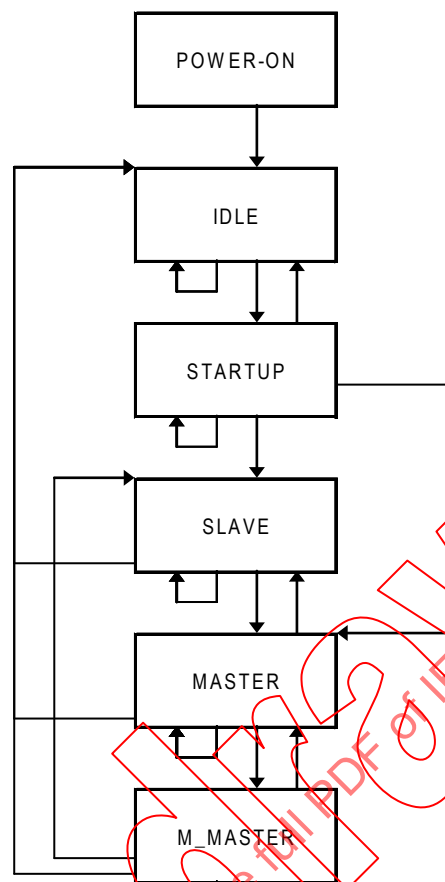


Figure 44 – State transition diagram of SYN_MPSM

States of the SYN_MPSM

POWER-ON	After start up the data management shall be initialized (POWER-ON) and the synchronization role shall be set (IDLE).
IDLE	The master role is set by PTCP services. After this the MPSM goes from state IDLE to state STARTUP and starts the BMA state machine.
STARTUP	During the start up period the MPSM is in state STARTUP.
SLAVE	The master is synchronized by sync messages from another master in the sync domain.
MASTER	This is the active primary master. Therefore it sends synchronization messages.
M_MASTER	This state handles the multiple master issues.

4.4.4.5.3 State machine description

The master protocol state machine controls the transition between the roles master and slave. The transitions are made local by every master for every supported PTCP SyncID and the engineering assigned PTCP SubDomainUUID. Therefore an instance of the state machine exists for every SyncID with the key attribute SubDomainUUID. To achieve these transitions the MPSM shall initiate synchronization and announce messages. Announce messages shall be sent by the SYN_BMA to advertise master capabilities and grant that only one master is active.

Multiple active primary masters shall be prevented by the state STARTUP and the best master algorithm. Therefore the decision to go from state STARTUP to state MASTER

(primary sync master) will be made by the SYN_BMA with extensive knowledge of concurrent masters.

If an active master (with fewer capabilities) already exists and after the start up period a new master comes along, the new master needs to synchronize to the active master. For synchronization the MPSM goes to state SLAVE.

4.4.4.5.4 SYN_MPSM state table

Table 140 contains the state table used by SYN_MPSM.

Table 140 – SYN_MPSM state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> SequenceID := 0 MultipleMaster := FALSE	IDLE
2	IDLE	StartMaster_req () => Status := OK BMA_Start_req () StartMaster_cnf (Status)	STARTUP
3	IDLE	StopMaster_req () => Status := OK StopMaster_cnf (Status)	IDLE
4	IDLE	Sync_cnf () => ignore	IDLE
5	IDLE	BMA_Stop_cnf () => ignore	IDLE
6	STARTUP	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status)	STARTUP
7	STARTUP	StopMaster_req () => Status := OK StopMaster_cnf (Status) BMA_Stop_req ()	IDLE
8	STARTUP	Sync_cnf () => ignore	STARTUP
9	STARTUP	BMA_State_ind (State) /State == SLAVE => Status := OK StartSlave_req ()	SLAVE
10	STARTUP	BMA_State_ind (State) /State == MASTER => State := SINGLE_MASTER SequenceID++ StartTimer (SyncTimer, SyncInterval) StartTimer (TimeoutT, SyncTimeOut) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	MASTER

#	Current State	Event /Condition =>Action	Next State
11	SLAVE	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status) BMA_Start_req ()	SLAVE
12	SLAVE	StopMaster_req () => Status := OK BMA_Stop_req () StopMaster_cnf (Status)	IDLE
13	SLAVE	Sync_cnf () => ignore	SLAVE
14	SLAVE	BMA_State_ind (State) /State == SLAVE => ignore	SLAVE
15	SLAVE	BMA_State_ind (State) /State == MASTER => State := SINGLE_MASTER SequenceID++ StartTimer (SyncTimer, SyncInterval) StartTimer (TimeoutT, SyncTimeOut) StopSlave_req () for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, RTCP_SYNC_PDU)	MASTER
16	MASTER	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status)	MASTER
17	MASTER	StopMaster_req () => Status := OK StopTimer (SyncTimer) StopTimer (TimeoutT) BMA_Stop_req () StopMaster_cnf (Status)	IDLE
18	MASTER	BMA_State_ind (State) /State == SLAVE => StopTimer (SyncTimer) StopTimer (TimeoutT) StartSlave_req ()	SLAVE
19	MASTER	BMA_State_ind (State) /State == MASTER => ignore	MASTER
20	MASTER	Sync_ind () /RECEIVE_PORT_VALID => ignore	MASTER
21	MASTER	Sync_ind () /RECEIVE_PORT_VALID && !NEW_SYNC_FRAME => ignore	MASTER

#	Current State	Event /Condition =>Action	Next State
22	MASTER	Sync_ind () /RECEIVE_PORT_VALID && NEW_SYNC_FRAME => MultipleMaster := TRUE	MASTER
23	MASTER	Sync_cnf () => ignore	MASTER
24	MASTER	TimerExpired (SyncTimer) => SequenceID++ StartTimer (SyncTimer, SyncInterval) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	MASTER
25	MASTER	TimerExpired (TimeoutT) /!MultipleMaster => StartTimer (TimeoutT, SyncTimeOut)	MASTER
26	MASTER	TimerExpired (TimeoutT) /MultipleMaster => MultipleMaster := FALSE StartTimer (TimeoutT, SyncTimeOut) SyncStateChange_ind (MULTIPLE_MASTER)	M_MASTER
27	M_MASTER	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status)	M_MASTER
28	M_MASTER	StopMaster_req () => Status := OK StopTimer (SyncTimer) StopTimer (TimeoutT) BMA_Stop_req () StopMaster_cnf (Status)	IDLE
29	M_MASTER	BMA_State_ind (State) /State == SLAVE => StopTimer (SyncTimer) StopTimer (TimeoutT) StartSlave_req ()	SLAVE
30	M_MASTER	BMA_State_ind (State) /State == MASTER => ignore	M_MASTER
31	M_MASTER	Sync_ind () /!RECEIVE_PORT_VALID => ignore	M_MASTER
32	M_MASTER	Sync_ind () /RECEIVE_PORT_VALID && !NEW_SYNC_FRAME => ignore	M_MASTER
33	M_MASTER	Sync_ind () /RECEIVE_PORT_VALID && NEW_SYNC_FRAME => MultipleMaster := TRUE	M_MASTER

#	Current State	Event /Condition =>Action	Next State
34	M_MASTER	Sync_cnf () => ignore	M_MASTER
35	M_MASTER	TimerExpired (SyncTimer) => SequenceID++ StartTimer (SyncTimer, SyncInterval) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	M_MASTER
36	M_MASTER	TimerExpired (TimeoutT) /!MultipleMaster => StartTimer (TimeoutT, SyncTimeOut) SyncStateChange_ind (SINGLE_MASTER)	MASTER
37	M_MASTER	TimerExpired (TimeoutT) /MultipleMaster => MultipleMaster := FALSE StartTimer (TimeoutT, SyncTimeOut)	M_MASTER

4.4.4.5.5 Functions, Macros, Timers and Variables

Table 141 contains the functions, macros, timers and variables used by the SYN_MPSM and their arguments and their descriptions.

Table 141 – Functions, Macros, Timers and Variables used by the SYN_MPSM

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or restarts a timer
StopTimer	Function	This local function stops a timer
TimerExpired	Function	This local function is issued if a timer expires
NEW_SYNC_FRAME	Macro	Check if the PTCP_SYNC frame is valid, the sequence number is newer and from the stored MasterMACAddress.
RECEIVE_PORT_VALID	Macro	Table 151 shall be applied.
Sync_cnf (+/-) (D_Port)	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status
Sync_ind (S_Port, PTCP_SYNC_PDU)	Macro	Receive PTCP-PDU according PTCP_SYNC_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_SYNC_PDU := A_SDU without LT and FrameID

Name	Type	Function/Meaning
Sync_req (D_Port, PTCP_SYNC_PDU)	Macro	Create PTCP-PDU according PTCP_SYNC_PDU Assignments: D_Port := D_Port DA := Multicast Address for PTCP_SYNC_PDU and corresponding SyncID SA := port source address PTCP_SYNC_PDU.SequenceID := SequenceID PTCP_SYNC_PDU.DomainUUID := DomainUUID PTCP_SYNC_PDU.MasterSourceAddress := local master address PTCP_SYNC_PDU.Time := local time from source PTCP_SYNC_PDU.TimeExtension := CurrentUTCOffset := Current UTC Offset PTCP_SYNC_PDU.MasterPriority1 := Priority PTCP_SYNC_PDU.ClockClass := Class PTCP_SYNC_PDU.ClockAccuracy := Accuracy PTCP_SYNC_PDU.ClockVariance := Variance PTCP_SYNC_PDU.MasterPriority2 := 0xFF A_SDU := LT, FrameID, PTCP_SYNC_PDU
TRANSMIT_PORT_VALID	Macro	LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
	Macro	Table 152 shall be applied.

4.4.4.6 PTCP Slave Protocol Machine

4.4.4.6.1 Primitive definitions

4.4.4.6.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by PTCP Slave Protocol Machine (SYN_SPSM) are described in the service definition and shown in Table 142.

Table 142 – Remote primitives issued or received by SYN_SPSM

Primitive	Source	Destination	Associated parameters	Functions
LMPM_P_Data_ind	LMPM	SPSM	CREP, S_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.6.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by SYN_SPSM are described in the service definition and shown in Table 143.

Table 143 – Local primitives issued or received by SYN_SPSM

Primitive	Source	Destination	Associated parameters	Functions
StartSlave_req	MPSM	SPSM	—	Activate slave function
StartSlave_cnf	SPSM	MPSM	Status	Confirmation for start slave
StopSlave_req	MPSM	SPSM	—	Deactivates slave function
StopSlave_cnf	SPSM	MPSM	Status	Confirmation for stop slave
SyncStateChange_ind	SPSM	FSPM	Status	Indication for events - JITTER_OUT_OF_BOUNDARY - NO_SYNC_MESSAGE_RECEIVED
SLAVE_State_ind	SPSM	BMA	Status	Indication for events used by the best master algorithms - MASTER_LOST

4.4.4.6.2 State transition diagram

The state transition diagram of the SYN_SPSM is shown in Figure 45.

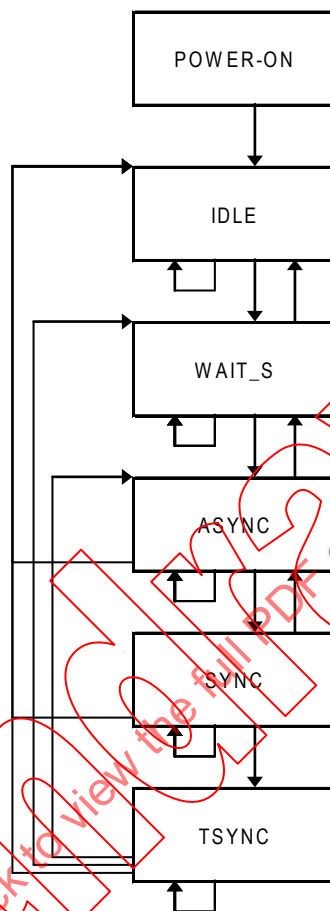


Figure 45 – State transition diagram of SYN_SPSM

States of the SYN_SPSM

POWER-ON	Data initialization
IDLE	The ASE service "StartSlave_req" shall start the slave function of the device.
WAIT_S	The slave is waiting for receiving sync messages in state WAIT_S.
ASYNC	The slave is not synchronized to the active master of the sync domain.
SYNC	The slave has achieved synchronicity with the active master of the sync domain.
TSYNC	The slave is still waiting for new sync messages.

4.4.4.6.3 State machine description

The slave protocol machine controls the transition of a sync slave device. Therefore an instance of the state machine exists for every SyncID with the key attribute PTCP_SubDomainUUID.

After start up the data management shall be initialized (POWER-ON) and the synchronization role shall be set (IDLE). The role is set by PTCP services. After this the SPSM goes from state IDLE to state WAIT_S. In state WAIT_S the slave waits for sync messages to synchronize to an active master. When receiving sync messages the SPSM goes from state WAIT_S to state ASYNC. If the slave is synchronized the SPSM shall switch to state SYNC. If in state SYNC no sync message will be received the SPSM shall switch to state TSYNC.

4.4.4.6.4 SYN_SPSM state table

Table 144 contains the state table used by SYN_SPSM.

Table 144 – SYN_SPSM state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> Stored PTCP_MasterSourceAddress := NIL	IDLE
2	IDLE	StartSlave_req () => store Parameter Status := OK StartTimer (TakeoverTimeoutT, MasterTimeOut) StartSlave_cnf (Status)	WAIT_S
3	IDLE	StopSlave_req () => Status := OK StopSlave_cnf (Status)	IDLE
4	WAIT_S	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	WAIT_S
5	WAIT_S	StopSlave_req () => Status := OK StopTimer (TakeoverTimeoutT) StopSlave_cnf (Status)	IDLE
6	WAIT_S	Sync_ind () /INVALID_SYNC_FRAME => ignore	WAIT_S
7	WAIT_S	Sync_ind () /INVALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time OFFSET_CTRL_START (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval) StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (JITTER_OUT_OF_BOUNDARY)	ASYNC
8	WAIT_S	TimerExpired (TakeoverTimeoutT) => Stored PTCP_MasterSourceAddress := NIL SLAVE_State_ind (MASTER_LOST)	WAIT_S
9	ASYNC	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	ASYNC

#	Current State	Event /Condition =>Action	Next State
10	ASYNC	StopSlave_req () => OFFSET_CTRL_STOP Status := OK StopTimer (OffsetContLoopT) StopTimer (TakeoverTimeoutT) StopTimer (SyncTimeoutT) StopSlave_cnf (Status)	IDLE
11	ASYNC	Sync_ind () /!VALID_SYNC_FRAME => ignore	ASYNC
12	ASYNC	Sync_ind () /VALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (INFO, M_Time, L_Time)	ASYNC
13	ASYNC	TimerExpired (OffsetContLoopT) /!OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval)	ASYNC
14	ASYNC	TimerExpired (OffsetContLoopT) /OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval) SyncStateChange_ind (SYNC)	SYNC
15	ASYNC	TimerExpired (TakeoverTimeoutT) => Stored PTCP_MasterSourceAddress := NIL StopTimer (SyncTimeoutT) SLAVE_State_ind (MASTER_LOST)	WAIT_S
16	SYNC	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	SYNC
17	SYNC	StopSlave_req () => OFFSET_CTRL_STOP Status := OK StopTimer (OffsetContLoopT) StopTimer (TakeoverTimeoutT) StopTimer (SyncTimeoutT) StopSlave_cnf (Status)	IDLE
18	SYNC	Sync_ind () /!VALID_SYNC_FRAME => ignore	SYNC

#	Current State	Event /Condition =>Action	Next State
19	SYNC	Sync_ind () /VALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (INFO, M_Time, L_Time)	SYNC
20	SYNC	TimerExpired (OffsetContLoopT) /OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval)	SYNC
21	SYNC	TimerExpired (OffsetContLoopT) /!OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval) SyncStateChange_ind (JITTER_OUT_OF_BOUNDARY)	ASync
22	SYNC	TimerExpired (TakeoverTimeoutT) => Stored PTCP_MasterSourceAddress := NIL SLAVE_State_ind (MASTER_LOST)	TSync
23	TSync	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	TSync
24	TSync	StopSlave_req () => OFFSET_CTRL_STOP Status := OK StopTimer (OffsetContLoopT) StopTimer (TakeoverTimeoutT) StopTimer (SyncTimeoutT) StopSlave_cnf (Status)	IDLE
25	TSync	Sync_ind () /!VALID_SYNC_FRAME => ignore	TSync
26	TSync	Sync_ind () /VALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (INFO, M_Time, L_Time)	ASync
27	TSync	TimerExpired (OffsetContLoopT) => StartTimer (OffsetContLoopT, SyncInterval) //NOTE The control loop should try to keep the clock stable	TSync
28	TSync	TimerExpired (SyncTimeoutT) => OFFSET_CTRL_STOP StopTimer (OffsetContLoopT) SyncStateChange_ind (NO_SYNC_MESSAGE_RECEIVED)	WAIT_S

4.4.4.6.5 Functions, Macros, Timers and Variables

Table 145 contains the functions, macros, timers and variables used by the SYN_SPSM and their arguments and their descriptions.

Table 145 – Functions, Macros, Timers and Variables used by the SYN_SPSM

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or restarts a timer
StopTimer	Function	This local function stops a timer
TimerExpired	Function	This local function is issued if a timer expires
OFFSET_CTRL_CALC	Macro	This local macro makes the offset compensation Parameters: - LocalTime - MasterTime
OFFSET_CTRL_IS_SYNC	Macro	This local macro measured the sync error in PLL Window //NOTE The detection of a fault depends on the filter algorithm of the selected control loop because the decision is made with local informations only.
OFFSET_CTRL_START	Macro	This local macro initialize the control loop for offset compensation Parameters: - LocalTime - MasterTime
OFFSET_CTRL_STOP	Macro	This local macro stops the offset compensation
Sync_ind	Macro	Receive PTCP-PDU according PTCP_SYNC_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) if A_SDU.FrameID == WITHOUT_FUP_PDU PTCP_SYNC_DATA := PTCP_SYNC_PDU else wait for valid PTCP_FUP_PDU from active Master LMPM_P_Data.ind (CREP, S_Port, DA, SA, A_SDU) PTCP_SYNC_DATA.Delay += PTCP_FUP_PDU.Delay Assignments: PTCP_SYNC_PDU := A_SDU without LT and FrameID PTCP_FUP_PDU := A_SDU without LT and FrameID Parameters: - SourcePort - LocalTime - PTCP_SYNC_DATA
VALID_SYNC_FRAME	Macro	See Table 146
OffsetContLoopT	Timer	This local timer is used for the offset checking between LocalTime and MasterTime. It shall be aligned to the SyncInterval for best control loop results.
SyncTimeoutT	Timer	This local timer is used to monitor the existence of the sync master. If this time expires the sync master is lost.
TakeoverTimeoutT	Timer	This local timer is used to monitor the existence of the sync master. If this time expires the preparation for a master switchover is done. This time shall always be lesser than SyncTimeoutT.
MasterTimeout	Variable	This local variable contains the value for the warning / preparation level master timeout. Default value: 3 x SyncInterval //NOTE This value is derived from the PTCPTakeoverTimeoutFactor.
PTCP_MasterSourceAddress	Variable	This shared variable contains the MasterSourceAddress of the actual SyncMaster. It is additional changed by the SYNC_RELAY.
Status	Variable	This local variable is used to store the status for further use.

Name	Type	Function/Meaning
SyncInterval	Variable	This local variable contains the value of the sync indication receiving interval. Default value: SyncInterval //NOTE This value is derived from the SyncSendFactor
SyncTimeout	Variable	This local variable contains the value fault level master timeout. Default value: 6 x SyncInterval //NOTE This value is derived from the PTCPTimeoutFactor

4.4.4.6.6 Sync receiving rules truth table for one SyncID

Table 146 contains the truth table used by the SPSM.

Table 146 – Truth table for one SyncID for receiving sync and follow up frames

Input								Output
Condition of the receive port ^b								Local
Port-State ^d	MAU-Type ^f	Line-Delay	Ingress Filter	Sub-domain-UUID	Master-MAC-Address	PTCP - Sequence ID	PTCP_Delay1ns / PTCP_Delay10ns	Frame from port valid
Disabled	—	—	—	—	—	—	—	No
!Disabled	Wrong	—	—	—	—	—	—	No
!Disabled	Ok	No	—	—	—	—	—	No
!Disabled	Ok	Yes	Yes	—	—	—	—	No
!Disabled	Ok	Yes	No	Wrong	—	—	—	No
!Disabled	Ok	Yes	No	Equal ^a	Wrong	—	—	No
!Disabled	Ok	Yes	No	Equal ^a	Equal ^a	Older	—	No
!Disabled	Ok	Yes	No	Equal ^a	Equal ^a	Newer ^c	Invalid ^e	No
!Disabled	Ok	Yes	No	Equal ^a	Equal ^a	Newer ^c	Valid ^e	Yes

^a If the own PTCP_SubdomainUUID and/or the MasterMACAddress is unknown (coded as zero), the compare with the values of the PTCP sync frame may return “equal” to optimize the startup of the synchronization.

^b At the receive port, the local PTCP_SubdomainUUID and/or the MasterMACAddress is checked against the PTCP frame.

^c If the own MasterMACAddress is unknown (coded as zero), a PTCP sync frame with the local PTCP_SubdomainUUID is defined as ‘newer’. The PTCP_SequenceID is bound to the MasterMACAddress.

^d According to IEEE 802.3 the PortStates are Disabled, Discarding, Learning and Forwarding. Only the PortState Disabled allows no forwarding or receiving of frames.

^e See the definition of the fields PTCP_Delay10ns and PTCP_Delay1ns for details.

^f MAUType is full duplex and a link exists.

4.4.4.7 Sync Relay Protocol Machine

4.4.4.7.1 Primitive definitions

4.4.4.7.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Sync Relay Protocol Machine (SYNC_RELAY) are described in the service definition and shown in Table 147.

Table 147 – Remote primitives issued or received by SYNC_RELAY

Primitive	Source	Destination	Associated parameters	Functions
LMPM_P_Data.req ()	SYNC_RELAY	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—
LMPM_P_Data.cnf ()	LMPM	SYNC_RELAY	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.ind ()	LMPM	SYNC_RELAY	CREP, D_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.7.1.1.2 Primitives exchanged between local machines

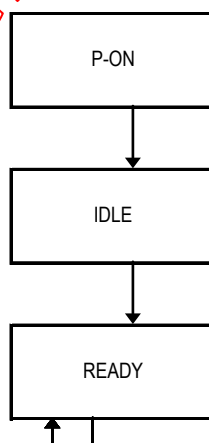
The local service primitives including their associated parameters issued or received by SYNC_RELAY are described in the service definition and shown in Table 148.

Table 148 – Local primitives issued or received by SYNC_RELAY

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.4.4.7.2 State transition diagram

The state transition diagram of the SYNC_RELAY is shown in Figure 46.

**Figure 46 – State transition diagram of SYNC_RELAY**

States of the SYNC_RELAY

P-ON	Initialization of the filtering data base for sync frames.
IDLE	Start of the receipt timer for aging the entries in the filtering data base.
READY	The state READY describes the rules for forwarding sync and followup messages.

4.4.4.7.3 State machine description

The SYNC_RELAY protocol machine describes the sync and followup forwarding rules for bridges.

Each time stamp unit of the bridge shall correct the delay field of sync frames on the fly. The time correction (bridge and line delay) is done by adding the delay while transmitting a sync frame.

The SYNC_RELAY protocol machine is necessary to avoid circulating sync messages in the network.

For a high precision the bridge delay of the sync and the follow up frame should be small and stable.

Announce frames shall be handled by the MAC_RELAY. Additional to the MAC_RELAY rules an implicit egress and ingress boundary should be activated for every port which does not support line delay measurement.

4.4.4.7.4 SYNC_RELAY state table

Table 149 contains the state table used by SYNC_RELAY. Only the PTCP frames from one master and one PTCP_SubdomainUUID for one SyncID shall be forwarded. After the selection of a master, the PTCP frames of the others shall be dropped. Table 151 and Table 152 show at a glance the conditions.

Table 149 – SYNC_RELAY state table

#	Current State	Event /Condition => Action	Next State
1	P-ON	=> Init RelayTable	IDLE
2	IDLE	=> StartTimer (PTCP_Timeout) StartTimer (PTCP_TakeoverTimeout)	READY
3	READY	TimerExpired (PTCP_Timeout) => AGING_OF_FWSM_LIST(PT) StartTimer (PTCP_Timeout)	READY
4	READY	TimerExpired (PTCP_TakeoverTimeout) => AGING_OF_FWSM_LIST(PTT) StartTimer (PTCP_TakeoverTimeout)	READY
5	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID => ignore	READY
6	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID && !NEW_SYNC_FRAME => ignore	READY
7	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID && NEW_SYNC_FRAME && !RATE_VALID => UPDATE_MASTER_ENTRY CalculateMasterRCF	READY

#	Current State	Event /Condition =>Action	Next State
8	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID && NEW_SYNC_FRAME && RATE_VALID => UPDATE_MASTER_ENTRY CalculateMasterRCF for i := all op. D_Ports \ S_Port do if (TRANSMIT_PORT_VALID) Sync_Req (i, PTCP_SYNC_PDU) //NOTE If the hardware does not support "sync request only forwarding" then a sync frame and a followup frame are conveyed.	READY
9	READY	Sync_Cnf () => ignore	READY
10	READY	Followup_Ind (S_Port, PTCP_FUP_PDU) /!RECEIVE_PORT_VALID => ignore	READY
11	READY	Followup_Ind (S_Port, PTCP_FUP_PDU) /RECEIVE_PORT_VALID && !INVALID_FU_FRAME => ignore	READY
12	READY	Followup_Ind (S_Port, PTCP_FUP_PDU) /RECEIVE_PORT_VALID && VALID_FU_FRAME && RATE_VALID => SET_FU_RECEIVED for i := all op. D_Ports \ S_Port do if (TRANSMIT_PORT_VALID) Followup_Req (i, PTCP_FUP_PDU)	READY
13	READY	Followup_Cnf () => ignore	READY

4.4.4.7.5 Functions, Macros, Timers and Variables

Table 150 contains the functions, macros, timers and variables used by the SYNC_RELAY and their arguments and their descriptions.

Table 150 – Functions, Macros, Timers and Variables used by the SYNC_RELAY

Name	Type	Function/Meaning
CalculateMasterRCF	Function	This local function calculates and updates the rate (RCF _{SyncMaster})
AGING_OF_FWSM_LIST	Macro	This local macro shall handle the entry in the RelayTable. In normal condition only none or one entry exists. PTCP_Timeout expired (PT): Delete the stored MasterMACAddress to enable the receiving of the sync frame from a primary/secondary master PTCP_TakeoverTimeout expired (PTT): Delete the stored MasterMACAddress, the stored PTCP_SubdomainUUID (if not defined locally) and set the rate invalid RelayTableEntry[SyncID, DomainUUID]: - MasterAddress - SequenceID - RxPort - FuFrame
Followup_Cnf	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status Parameters: - DestinationPort - Status
Followup_Ind	Macro	Receive PTCP-PDU according PTCP_FUP_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_FUP_PDU := A_SDU without LT and FrameID Parameters: - SourcePort - PTCP_FUP_PDU
Followup_Req	Macro	Create PTCP-PDU according PTCP_FUP_PDU Assignments: D_Port := Port DA := multicast address for PTCP_FUP_PDU SA := local port address A_SDU := LT, FrameID, PTCP_FUP_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU) Parameters: - DestinationPort - PTCP_FUP_PDU
NEW_SYNC_FRAME	Macro	This local macro check as a second check after RECEIVE_PORT_VALID if the PTCP_SYNC frame is valid. It checks the semantics and the values of the delay fields.
RATE_VALID	Macro	This local macro checks if the rate (RCF _{SyncMaster}) is calculated and valid in the range of a $\pm 100 \frac{\mu\text{Hz}}{\text{Hz}}$ ($\pm 250 \frac{\mu\text{Hz}}{\text{Hz}}$) clock.
RECEIVE_PORT_VALID	Macro	Table 151 shall be applied.
SET_FU_RECEIVED	Macro	Update entry in RelayTable[SyncID, DomainUUID] FuFrame := TRUE

Name	Type	Function/Meaning
Sync_Cnf	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Assignments: Status := LMPM_status Parameters: - DestinationPort - Status
Sync_Ind	Macro	Receive PTCP-PDU according PTCP_SYNC_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Assignments: PTCP_SYNC_PDU := A_SDU without LT and FrameID Parameters: - SourcePort - PTCP_SYNC_PDU
Sync_Req	Macro	Create PTCP-PDU according PTCP_SYNC_PDU Assignments: D_Port := Port DA := multicast address for PTCP_SYNC_PDU SA := local port address A_SDU := LT, FrameID, PTCP_SYNC_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU) Parameters: - DestinationPort - PTCP_SYNC_PDU
TRANSMIT_PORT_VALID	Macro	Table 152 shall be applied.
UPDATE_MASTER_ENTRY	Macro	Update entry in Relay Table[SyncID, DomainUUID] for sync master MasterAddress := PTCP_SYNC_PDU.MasterSourceAddress SequenceID := PTCP_SYNC_PDU.SequenceID RxPort := S_Port FuFrame := FALSE
VALID_FU_FRAME	Macro	Check RelayTable[SyncID, DomainUUID] for receive port and sequence number is valid MasterAddress == PTCP_FUP_PDU.MasterSourceAddress SequenceID == PTCP_FUP_PDU.SequenceID RxPort == S_Port FuFrame == FALSE
PTCP_TakeoverTimeout	Timer	This local variable contains the value for the warning / preparation level master timeout. Default value: 3 x SyncInterval //NOTE This value is derived from the PTCPTakeoverTimeoutFactor
PTCP_Timeout	Timer	This local variable contains the value fault level master timeout. Default value: 6 x SyncInterval //NOTE This value is derived from the PTCPTimeoutFactor

4.4.4.7.6 Sync forwarding rules for bridges truth table for one SyncID

Table 151 and Table 152 contain the truth table used by the SYNC_RELAY.

Table 151 – Truth table for one SyncID for receiving

Input						Output
Condition of the receive port ^b						Local
PortState ^c	MAUType	LineDelay	Ingress-Filter	Subdomain-UUID	MasterMACAddress	Frame from port valid
Disabled	—	—	—	—	—	No
!Disabled	Wrong	—	—	—	—	No
!Disabled	Ok	No	—	—	—	No
!Disabled	Ok	Yes	Yes	—	—	No
!Disabled	Ok	Yes	No	Wrong	—	No
!Disabled	Ok	Yes	No	Equal ^a	Wrong	No
!Disabled	Ok	Yes	No	Equal ^a	Equal ^a	Yes ^d

^a If the own PTCP_SubdomainUUID and/or the MasterMACAddress is unknown (coded as zero), the compare with the values of the PTCP sync frame may return “equal” to optimize the startup of the synchronization.

^b At the receive port, the local PTCP_SubdomainUUID and/or the MasterMACAddress is checked against the PTCP frame.

^c According to IEEE 802.3 the PortStates are Disabled, Discarding, Learning and Forwarding. Only the PortState Disabled allows no forwarding or receiving of frames.

^d An optional filter should be used to discard frames with invalid content of the PTCP_Delay1ns / PTCP_Delay10ns fields.

Table 152 – Truth table for one SyncID for transmitting

Input							Output
Local	Conditions of a transmit port ^b						Local
Rate known	Subdomain-UUID	MasterMAC-Address	PortState	MAU-Type	Line-Delay	Egress-Filter	Port for frame valid
No	—	—	—	—	—	—	No
Yes	Wrong	—	—	—	—	—	No
Yes	Equal ^a	Wrong	—	—	—	—	No
Yes	Equal ^a	Equal ^a	Discarding	—	—	—	No
Yes	Equal ^a	Equal ^a	!Discarding	Wrong	—	—	No
Yes	Equal ^a	Equal ^a	!Discarding	Ok	No	—	No
Yes	Equal ^a	Equal ^a	!Discarding	Ok	Yes	Yes	No
Yes	Equal ^a	Equal ^a	!Discarding	Ok	Yes	No	Yes

^a If the own PTCP_SubdomainUUID and/or the MasterMACAddress is unknown (coded as zero), the compare with the values of the PTCP frame may return “equal” to optimize the startup of the synchronization.

^b At the transmit port, the remote PTCP_SubdomainUUID and/or the MasterMACAddress is checked against the PTCP frame.

4.4.4.8 Scheduler Protocol Machine

4.4.4.8.1 Primitive definitions

4.4.4.8.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by SCHEDULER are described in the service definition and shown in Table 153.

Table 153 – Remote primitives issued or received by SCHEDULER

Primitive	Source	Destination	Associated parameters	Functions
LMPM_C_Data.req	SCHEDULER	LMPM	CREP, PortList, DA, SA, Prio, C_SDU, APDU_Status	—
UDP_C_Data.req	SCHEDULER	IP	IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status	—
LMPM_C_Data.cnf	LMPM	SCHEDULER	CREP, PortList, DA, SA, Prio, C_SDU, APDU_Status	—
UDP_C_Data.cnf	IP	SCHEDULER	IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status	—

4.4.4.8.1.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by SCHEDULER are described in the service definition and shown in Table 154.

Table 154 – Local primitives issued or received by SCHEDULER

Primitive	Source	Destination	Associated parameters	Functions
DMUX_Schedule_cnf	DEMUX	SCHEDULER	—	—
MUX_Schedule_cnf	MUX	SCHEDULER	—	—
DMUX_Schedule_req	SCHEDULER	DEMUX	CycleCounter, Rx_Period	—
DMUX_Schedule_req	SCHEDULER	DEMUX	Rx_Period	—
MUX_Schedule_req	SCHEDULER	MUX	CycleCounter, Tx_Period	—
MUX_Schedule_req	SCHEDULER	MUX	Tx_Period	—
SCHEDULER_add_req	PPM	SCHEDULER	CREP, ReductionRatio, Phase, Sequence, TxOptions	Add a PPM to the scheduler
SCHEDULER_remove_req	PPM	SCHEDULER	CREP	Remove a PPM from the scheduler
SCHEDULER_error_ind	SCHEDULER	CMSU CTLSU	CREP, ErrorCode	Signal an overload. In this case the NEXT phase shall be skipped
SCHEDULER_add_cnf	SCHEDULER	PPM	CREP	Add a PPM to the scheduler
SCHEDULER_remove_cnf	SCHEDULER	PPM	CREP	Remove a PPM from the scheduler

4.4.4.8.2 State transition diagram

The state transition diagram of the SCHEDULER is shown in Figure 47.

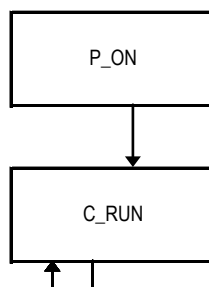


Figure 47 – State transition diagram of SCHEDULER

States of the SCHEDULER

P_ON	Data initialization
C_RUN	Scheduler is active and running

4.4.4.8.3 State machine description

The SCHEDULER generates the signals for the local handling of time events. The TickEvent is additionally used to trigger the timely correct forwarding of RT_CLASS_3 frames.

4.4.4.8.4 SCHEDULER state table

Table 155 contains the state table used by SCHEDULER.

Table 155 – SCHEDULER state table

#	Current State	Event /Condition =>Action	Next State
1	P_ON	Scheduler_Init_req () => TxPeriod := GREEN RxPeriod := GREEN MUX_Schedule_req(TxPeriod) DMUX_Schedule_req(RxPeriod)	C_RUN
2	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == GREEN && TxPeriodChecker () == GREEN => ignore	C_RUN
3	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == RED && TxPeriodChecker () == GREEN => TxPeriod := GREEN MUX_Schedule_req(CycleCounter, TxPeriod)	C_RUN
4	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == RED && TxPeriodChecker () == RED => ignore	C_RUN

#	Current State	Event /Condition =>Action	Next State
5	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == GREEN && TxPeriodChecker () == RED => TxPeriod := RED MUX_Schedule_req(CycleCounter, TxPeriod)	C_RUN
6	C_RUN	MUX_Schedule_cnf() => ignore	C_RUN
7	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == GREEN && RxPeriodChecker () == GREEN => ignore	C_RUN
8	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == RED && RxPeriodChecker () == GREEN => RxPeriod := GREEN DMUX_Schedule_req(CycleCounter, RxPeriod)	C_RUN
9	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == RED && RxPeriodChecker () == RED => ignore	C_RUN
10	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == GREEN && RxPeriodChecker () == RED => RxPeriod := RED DMUX_Schedule_req(CycleCounter, RxPeriod)	C_RUN
11	C_RUN	DMUX_Schedule_cnf() => ignore	C_RUN
12	C_RUN	TickEvent (CycleCounter, CycleOffset) /for each active PPM check whether CycleCounter, CycleOffset fits to FrameSendOffset and ReductionRatio check if PPM ready to schedule and frame not already at the LMPM => C_Data_req()	C_RUN
13	C_RUN	TickEvent (CycleCounter, CycleOffset) /for each active PPM check whether CycleCounter, CycleOffset fits to FrameSendOffset and ReductionRatio check if PPM ready to schedule and frame already at the LMPM => SCHEDULER_error.ind (ErrorCode = Overload)	C_RUN
14	C_RUN	C_Data_cnf() => ignore	C_RUN
15	C_RUN	SCHEDULER_add.req (CREP, ReductionRatio, Phase, Sequence) => Add PPM to list SCHEDULER_add.cnf (CREP)	C_RUN
16	C_RUN	SCHEDULER_remove.req (CREP) => Remove PPM from list SCHEDULER_remove.cnf (CREP)	C_RUN

4.4.4.8.5 Functions, Macros, Timers and Variables

Table 156 contains the functions, macros, timers and variables used by the SCHEDULER and their arguments and their descriptions.

Table 156 – Functions, Macros, Timers and Variables used by the SCHEDULER

Name	Type	Function/Meaning
RxPeriodChecker	Function	This local function checks the active RxPeriod and returns it.
TickEvent	Function	Local generated event from a timer. The timer shall be controlled by the PTCP master or slave in case of synchronization or free running without synchronization. Parameter: - CycleCounter - CycleOffset
TxPeriodChecker	Function	This local function checks the active TxPeriod and returns it.
C_Data_cnf	Macro	if (txOption == RTC) LMPM_C_Data.cnf (CREP, LMPM_status) if (txOption == UDP) UDP_C_Data.cnf (IPPort, CREP, LMPM_status)
C_Data_req	Macro	if (txOption == RTC) LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) if (txOption == UDP) UDP_C_Data.req (IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status)
RxPeriod	Variable	This variable contains the values: RED and GREEN
TxPeriod	Variable	This variable contains the values: RED and GREEN

4.4.4.8.6 Truth tables

4.4.4.8.6.1 PeriodChecker rules truth table

Table 157 and Table 158 contains the truth table used by the SCHEDULER to detect the required period per port.

Table 157 – Truth table for RxPeriodChecker of one port

Input		Output
RTC3PSM port state	CycleOffset inside RED period	RxPeriod
OFF	—	GREEN
—	Outside	GREEN
UP	Inside	GREEN
RUN	Inside	RED

Table 158 – Truth table for TxPeriodChecker of one port

Input		Output
RTC3PSM port state	CycleOffset inside RED period	TxPeriod
OFF	—	GREEN
—	Outside	GREEN
UP	Inside	RED
RUN	Inside	RED

4.4.5 DLL Mapping Protocol Machines

There is no DLL Mapping Protocol Machine (DMPM) defined for this protocol.

4.5 Time synchronization

For time synchronization the IEEE 802.1AS applies. The support of the IEEE 802.1AS Management Information Base (MIB) is optional.

4.6 Media redundancy

4.6.1 Media redundancy and loop prevention

For media redundancy the IEC 62439-2 applies. The support of the MRP Management Information Base (MIB) is optional.

4.6.2 Seamless media redundancy

For seamless media redundancy MRPD applies. This shall be done by using the PPM, CPM and the RED_RELAY.

MRPD shall only be operated in conjunction with MRP.

4.7 Real time cyclic

4.7.1 FAL syntax description

4.7.1.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.7.1.2 RTC APDU abstract syntax

Table 159 defines the abstract syntax of the Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 159 shall represent the content of the DLSDU in Table 5.

Table 159 – RTC APDU syntax

APDU name	APDU structure
RTC-PDU	C_SDU ^ CSF_SDU, [RTCPadding*] ^a ^b , APDU_Status
UDP-RTC-PDU	IPHeader, UDPHeader, FrameID, RTC-PDU
^a In case of RTC-PDU the number of padding octets shall be in the range of 0...40 depending on the Dataltem size. In case of UDP-RTC-PDU the number of padding octets shall be in the range of 0...12 depending on the Dataltem size.	
^b In case of RT_CLASS_3 transportation the number of padding octets is given by the engineering system.	

Table 160 defines structures for substitutions of elements of the APDU structures shown in Table 159.

Table 160 – RTC substitutions

Substitution name	Structure
C_SDU	{[Dataltem], [GAP*]}* ^a
Dataltem	{[IOCS*], [DataObjectElement*]}*
SubstituteDataltem	{[IOCS*] ^b , [SubstituteDataObjectElement*]}*

Substitution name	Structure
DataObjectElement	[Data*], IOPS
SubstituteDataObjectElement	[Data*], SubstituteDataValid
CSF_SDU	SFCRC16, SUBFRAME*, SFEndDelimiter ^a
SUBFRAME	SFPosition, SFDataLength, SFCycleCounter, DataStatus, C_SDU, SFCRC16
SFEndDelimiter	SFPosition(=0), SFDataLength(=0)
APDU_Status	CycleCounter, DataStatus, TransferStatus
^a The maximum size of a C_SDU or CSF_SDU shall be not larger than 1 440 octets.	
^b The value shall contain the same value as the field DataItem.IOCS.	

4.7.2 FAL transfer syntax

4.7.2.1 Coding section of Data-RTC-PDU specific fields

4.7.2.1.1 Overview

The field Data and IOxS shall also be used to encode user data for all other PDU types.

4.7.2.1.2 Coding of the field CycleCounter

This field shall be coded as data type Unsigned16. One increment represents a time value of 31,25 µs for RTC-PDU and 1 ms for UDP-RTC-PDU. Each RTC-PDU or UDP-RTC-PDU contains its CycleCounter created according to Figure 49 by the sender.

The receiver of the RTC-PDU or UDP-RTC-PDU shall use this field to detect repetitions, loss of frames and timeliness of data. For this reason the CycleCounter value range shall be divided in the ratio 15/16 for valid and 1/16 for invalid as shown in Figure 48.

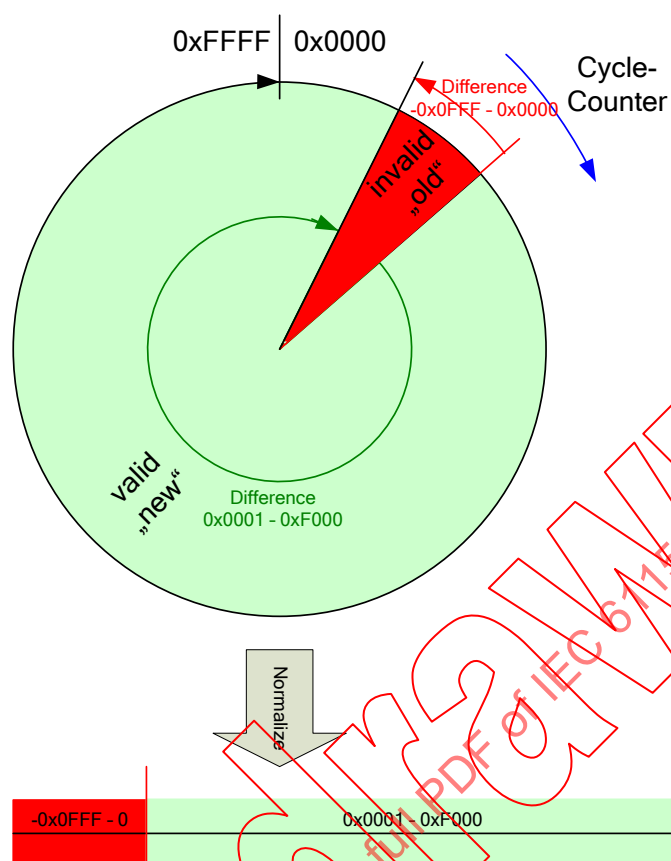


Figure 48 – CycleCounter value range

NOTE The normalization can be done using the following formula:

$\text{CycleCounter(Normalized)} := (((\text{CycleCounter(NEW)} + (0x10000 - \text{CycleCounter(STORED)}) - 1) \& 0xFFFF) - 0xF000) \times (-1)$.

Table 161 defines a maximum time window between the old and the new frame and the action of the consumer according to the difference between the CycleCounter of the last received RTC-PDU or UDP-RTC-PDU and the value of the current received PDU.

Table 161 – CycleCounter Difference

Value (hexadecimal)	Time range RTC-PDU [s]	Time range UDP-RTC-PDU [s]	Meaning	Use
-0x0FFF – 0x0000	0,128	4,096	Red sector according to Figure 48 1/16 of the CycleCounter range The consumer votes the received frame as older than the stored frame	Frame is dropped
0x0001 – 0xF000	1,92	61,44	Green sector according to Figure 48 15/16 of the CycleCounter range The consumer votes the received frame as newer than the stored frame	Frame is processed

If MRPD is used, the CycleCounter of the adjunctive frames shall be set identical by the sender.

A device shall maintain a local 64 bit counter with an increment of $31,25 \mu\text{s}$ as shown in Figure 49.

The bits 0 to 14 shall be used identically for the CycleCounter of the RTC-PDUs. The bit 15 of the 16 bit CycleCounter of the RTC-PDUs may also be identically to bit 15 of the 64 bit local counter.

The bits 5 to 19 shall be used as bit 0 to 14 identically for the CycleCounter of the UDP-RTC-PDUs. The bit 15 of the 16 bit CycleCounter of the UDP-RTC-PDUs may also be identically to bit 20 of the 64 bit local counter.

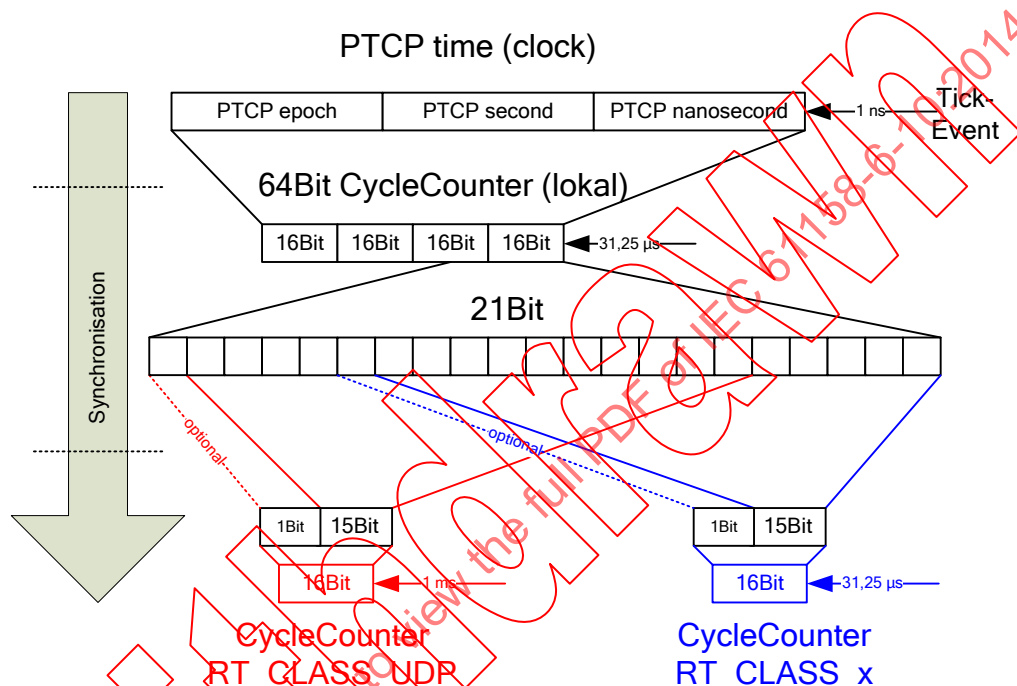


Figure 49 – Structure of the CycleCounter

In case of a global synchronization of the nodes, the bit 15 of the CycleCounter base for RTC-PDUs and the bit 20 of the CycleCounter base for UDP-RTC-PDUs shall be set according to Figure 49 and Figure 50 to avoid a false “older” detection by the receiver.

The CycleCounter shall be set due to synchronization according to the model shown in Figure 50.

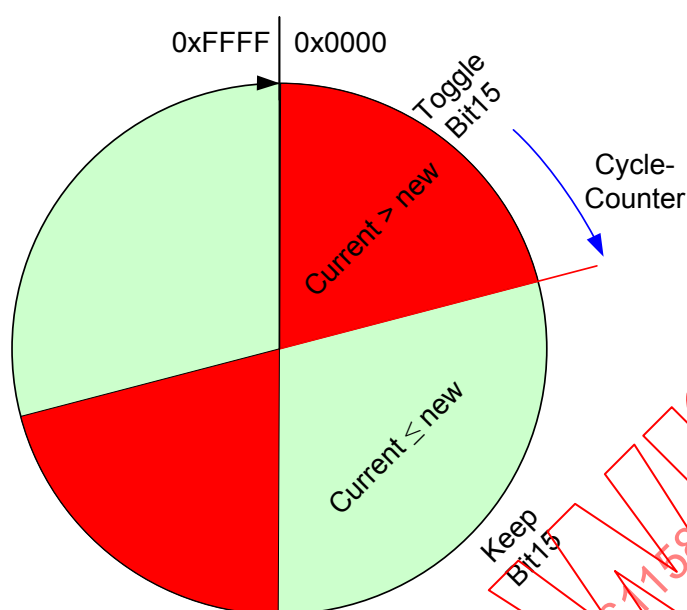


Figure 50 – Optimized CycleCounter setting

CycleCounter(current)

This means the value of the CycleCounter used by the sender before executing the synchronization.

CycleCounter(next)

This means the value of the CycleCounter needed by the sender according to the synchronization information

```

if CycleCounter(current)[0..14] <= CycleCounter(next)[0..14]
then /* current ≤ next */
SET CycleCounter[0..14] with CycleCounter(next)[0..14] and keep CycleCounter[15]
else /* current > next */
SET CycleCounter[0..14] with CycleCounter(next)[0..14] and toggle CycleCounter[15]

```

4.7.2.1.3 Coding of the field DataStatus

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0: DataStatus.State

This field shall be coded with the values according to Table 162 and evaluated after the AR is in state InDATA.

Table 162 – DataStatus.State

Value (hexadecimal)	Meaning
0x00	IOCR state is backup
0x01	IOCR state is primary

Bit 1: DataStatus.Redundancy

This field shall be coded with the values according to Table 163 and evaluated after the AR is in state InDATA.

Table 163 – DataStatus.Redundancy

Value (hexadecimal)	Meaning
0x00	Default One primary AR of a given AR-set is present
0x01	None primary AR of a given AR-set is present

NOTE An IO controller set this flag independent from the ARType to the Default value.

Bit 2: DataStatus.DataValid

This field shall be coded with the values according to Table 164.

Table 164 – DataStatus.DataValid

Value (hexadecimal)	Meaning
0x00	Dataltem invalid
0x01	Dataltem valid

It is recommended to use DataStatus.DataValid == 'Dataltem invalid' for DFP only in conjunction with IOxS.DataState == 'Bad' for all Dataltems.

Bit 3: DataStatus.Reserved_1

This field shall be set to zero but not checked by the receiver

Bit 4: DataStatus.ProviderState

This field shall be coded with the values according to Table 165.

Table 165 – DataStatus.ProviderState

Value (hexadecimal)	Meaning
0x00	Stop
0x01	Run

Bit 5: DataStatus.StationProblemIndicator

This field shall be coded with the values according to Table 166.

Table 166 – DataStatus.StationProblemIndicator

Value (hexadecimal)	Meaning
0x00	Problem detected
0x01	Normal Operation

Bit 6: DataStatus.Reserved_2

This field shall be set according to 3.4.2.2.

Bit 7: DataStatus.Ignore

This field shall be coded with the values according to Table 167 and Table 168.

Table 167 – DataStatus.Ignore of a frame

Value (hexadecimal)	Meaning
0x00	Evaluate the DataStatus
0x01	Ignore the DataStatus Shall only be set for a frame containing sub frames with SFIOCRProperties.SFCRC16 := 1

Table 168 – DataStatus.Ignore of a sub frame

Value (hexadecimal)	Meaning
0x00	Evaluate the sub frame
0x01	Ignore the sub frame; it is a surrogate sub frame. The CPM threads this subframe as invalid.

4.7.2.1.4 Coding of the field TransferStatus

This field shall be coded as data type Unsigned8. This field shall be set to zero for RT_CLASS_UDP, RT_CLASS_1 and RT_CLASS_2 frames. For RT_CLASS_3 the value shall be set according to Table 169.

Table 169 – TransferStatus for RT_CLASS_3

Bit position	Name	Value	Meaning
0	AlignmentOrFrameChecksumError	0	No frame checksum error or alignment error
		1	A frame checksum error or alignment error and no MAC receive buffer overflow has appeared
1	WrongLengthError	0	No length error
		1	A length error has appeared
2	MACReceiveBufferOverflow	0	No MAC receive buffer overflow
		1	A MAC receive buffer overflow has appeared
3	RT_CLASS_3 Error	0	No RT_CLASS_3 error
		1	RT_CLASS_3 error E.g. the frame has not been received in time or has not had the expected frame type or has not had the expected frame ID or has not had the expected Source MAC address (this is optionally checked) or in case of an underrun, e.g. the bridge received a header, but the data have not arrived on time
Other	—	Reserved	Reserved

4.7.2.1.5 Coding section related to decentralized periphery**4.7.2.1.5.1 Coding of the field Data**

The data types defined in IEC 61158-5-10, Clause 5 shall be used for Data.

4.7.2.1.5.2 Coding section related to Subframe**4.7.2.1.5.2.1 Coding of the field SFCRC16**

This field shall be coded as data type Unsigned16.

The SFCRC16 is generated by Formula (38) as a 16 Bit-polynomial.

$$\text{SFCRC16} = 1 + x^2 + x^{15} + x^{16} \quad (38)$$

where

x is the content of the octet

SFCRC16 16 Bit-polynomial

Each SFCRC16 starts with value 0 and calculates all octets beginning with first octet of destination MAC address up to the SFCRC16.

The 16 bit value representation in a frame is little endian (which ensures that the value of CRC is 0 after the SFCRC16 octets of the previous Subframe).

NOTE The design could be checked with <http://www.easics.be/webtools/crctool>.

The generation of the first SFCRC16 starts with the first octet of the destination MAC address, ignores the VLAN tag (and other tags) and ends with the last octet of the FrameID.

The generation of the Subframe SFCRC16, as shown in Figure 51 starts with the first octet of the SFPosition and ends with the last octet of the field C_SDU.

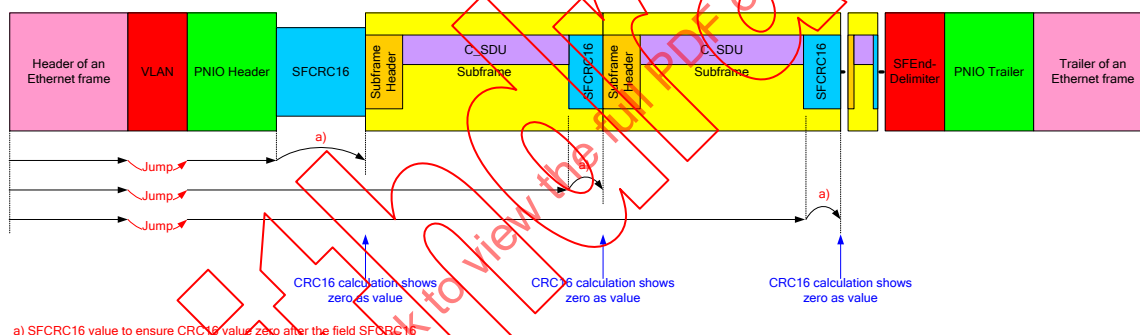


Figure 51 – SFCRC16 generation rule

4.7.2.1.5.2.2 Coding of the field SFPosition

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0-6: SFPosition.Position

This field shall be coded with the values according to Table 170.

Table 170 – SFPosition.Position

Value (hexadecimal)	Meaning
0x00	End of Subframe coding
0x01 – 0x3F	Useable for packetizing and unpacking of Subframes during transport and end to end
0x40 – 0x7F	Useable for packetizing and unpacking of Subframes end to end

NOTE End to end means, that the source of data creates and concatenates the Subframes for a frame. The conveyance is done without any change on the frame. The sink of the data unpacks the Subframes and uses the data.

Bit 7: SFPosition.Reserved

This field shall be coded with the values according to Table 171.

Table 171 – SFPosition.Reserved

Value (hexadecimal)	Meaning
0x00	Reserved

4.7.2.1.5.2.3 Coding of the field SFDataLength

This field shall be coded as data type Unsigned8 with the values according to Table 172.

Table 172 – SFDataLength

Value (hexadecimal)	Meaning
0x00	End of Subframe coding
0x01 – 0xFF	Number of octets of the C_SDU

4.7.2.1.5.2.4 Coding of the field SFCycleCounter

This field shall be coded as data type Unsigned8. Each Subframe contains its SFCycleCounter incremented by one for each transmission from the PPM.

The receiver of the Subframe shall use this field to detect repetitions, loss of Subframes and timeliness of data. For this reason the SFCycleCounter value range shall be divided in the ratio 15/16 for valid and 1/16 for invalid as shown in Figure 52.

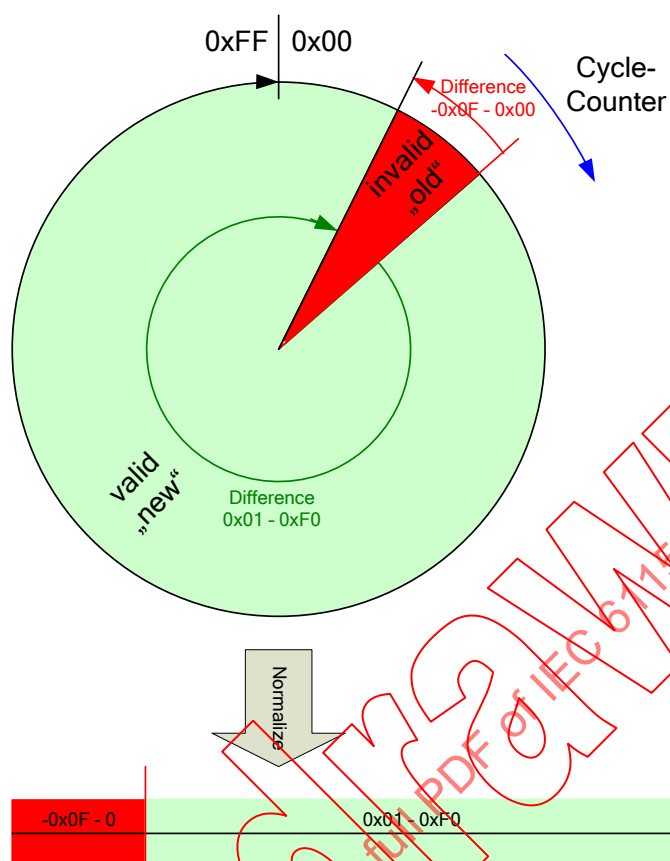


Figure 52 – SFCycleCounter value range

Table 173 defines a maximum time window between the old and the new SFCycleCounter and the action of the consumer according to the difference between the SFCycleCounter of the last received Subframe and the value of the current received Subframe.

Table 173 – SFCycleCounter Difference

Value (hexadecimal)	Time range RTC-PDU [s]	Time range UDP-RTC- PDU [s]	Meaning	Use
-0x0F– 0x00	—	—	Red sector according to Figure 52 1/16 of the SFCycleCounter range The consumer votes the received Subframe as older than the stored Subframe	Subframe is dropped
0x01 – 0xF0	—	—	Green sector according to Figure 52 15/16 of the SFCycleCounter range The consumer votes the received Subframe as newer than the stored Subframe	Subframe is processed

NOTE The same SFCycleCounter value, which means the difference is zero, is assigned to older.

If MRPD is used, the SFCycleCounter of the adjunctive subframes shall be set identical by the sender.

4.7.2.1.5.3 Coding section related to IOxS (IOPS, IOCS, SubstituteDataValid)

4.7.2.1.5.3.1 Coding of the field IOPS

The coding of these fields shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0: IOxS.Extension

This field shall be coded with the values according to Table 174.

Table 174 – IOxS.Extension

Value (hexadecimal)	Meaning
0x00	No IOxS octet follows
0x01	One more IOxS octet follows

Bit 1 to 4: IOxS.reserved

This field shall be set according to 3.4.2.2.

Bit 5 to 6: IOxS.Instance

This field shall be coded with the values according to Table 175 if the IOxS.DataState bit is set to bad. If this bit is set to good the IOxS instance bits are don't care and shall be set to zero.

Table 175 – IOxS.Instance

Value (hexadecimal)	Meaning
0x00	Detected by subslot
0x01	Detected by slot
0x02	Detected by IO device
0x03	Detected by IO controller

Bit 7: IOxS.DataState

This field shall be coded with the values according to Table 176.

Table 176 – IOxS.DataState

Value (hexadecimal)	Meaning
0x00	Bad Data field may not contain valid user data – the receiver shall use its pre-configured values (zero, last valid value, or default value) instead of transmitted data field values SubstituteDataValid = FALSE
0x01	Good Data field shall contain valid user data SubstituteDataValid = TRUE

4.7.2.1.5.3.2 Coding of the field IOCS

The coding of these fields shall be according to 4.7.2.1.5.3.1.

4.7.2.1.5.3.3 Coding of the field SubstituteDataValid

The coding of these fields shall be according to 4.7.2.1.5.3.1.

4.7.3 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.7.4 Application Relationship Protocol Machines

4.7.4.1 Provider protocol machine

4.7.4.1.1 General

Figure 53 and Figure 54 shows the basic structure of the Provider protocol machine (PPM).

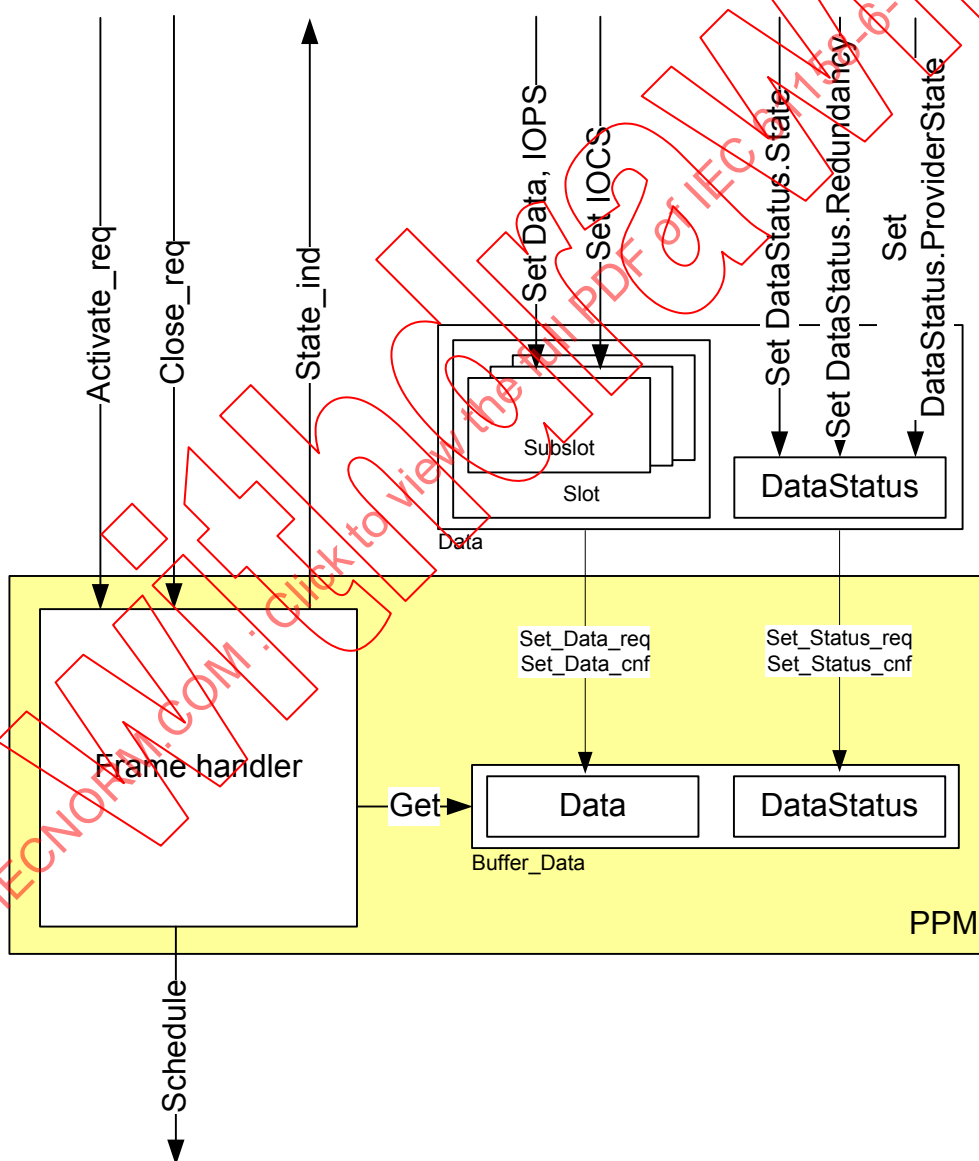


Figure 53 – Basic structure of a PPM with frame structure

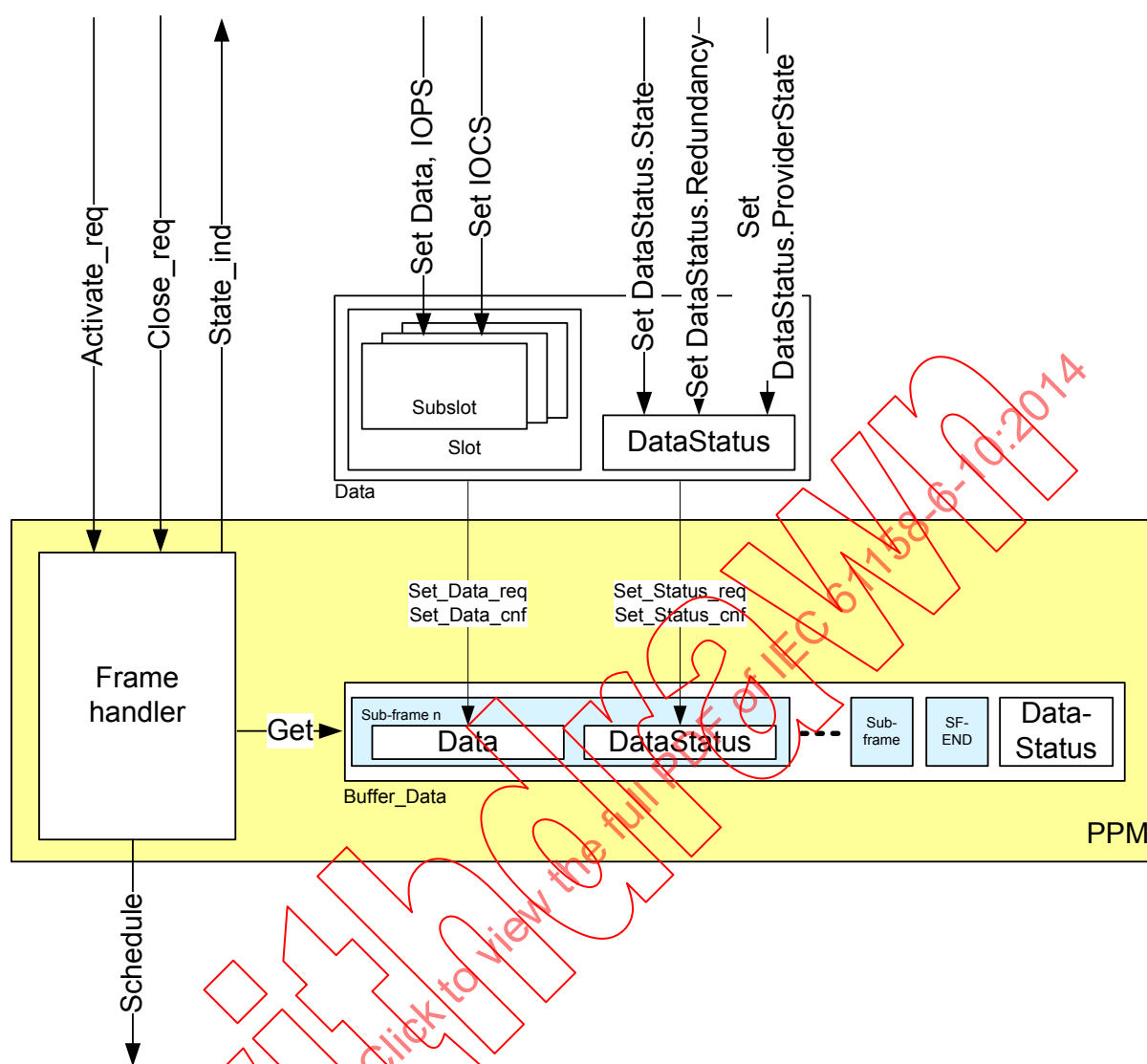


Figure 54 – Basic structure of a PPM with subframe structure

4.7.4.1.2 Primitive definitions

4.7.4.1.2.1 Primitives exchanged between remote machines

The remote service primitives including their associated parameters issued or received by PPM are described in the service definition and shown in Table 177.

Table 177 – Remote primitives issued or received by PPM

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.7.4.1.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by PPM are described in the service definition and shown in Table 178.

Table 178 – Local primitives issued or received by PPM

Primitive	Source	Destination	Associated parameters	Functions
PPM_Activate_cnf (-)	PPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
PPM_Activate_cnf (+)	PPM	CMSU CTLSU	CREP	—
PPM_Activate_req	CMSU CTLSU	PPM	CREP, DA, SA, FrameID, Prio, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status	—
PPM_Close_cnf (-)	PPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
PPM_Close_cnf (+)	PPM	CMSU CTLSU	CREP	—
PPM_Close_req	CMSU CTLSU	PPM	CREP	—
PPM_Error_ind	PPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
PPM_Set_Data_cnf (-)	PPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
PPM_Set_Data_cnf (+)	PPM	FSPMDEV FSPMCTL	CREP	—
PPM_Set_Data_req	FSPMDEV FSPMCTL	PPM	CREP, Data	—
PPM_Set_Status_cnf (-)	PPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
PPM_Set_Status_cnf (+)	PPM	FSPMDEV FSPMCTL	CREP	—
PPM_Set_Status_req	FSPMDEV FSPMCTL	PPM	CREP, Status	—
PPM_State_ind	PPM	local matter	CREP, State	—
Scheduler_add_cnf (-)	SCHEDULER	PPM	CREP	—
Scheduler_add_cnf (+)	SCHEDULER	PPM	CREP	—
SCHEDULER_add_req	PPM	SCHEDULER	CREP, ReductionRatio, Phase, Sequence, TxOption	—
Scheduler_remove_cnf (-)	SCHEDULER	PPM	CREP	—
Scheduler_remove_cnf (+)	SCHEDULER	PPM	CREP	—
SCHEDULER_remove_req	PPM	SCHEDULER	CREP	—
Scheduler_transmit_ind	SCHEDULER	PPM	CREP	—

4.7.4.1.3 State transition diagram

The state transition diagram of the PPM is shown in Figure 55.

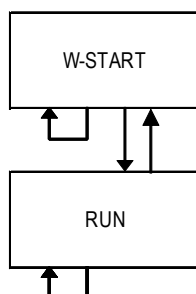


Figure 55 – State transition diagram of PPM

States of the PPM

W-START

Waiting for initialization and add the frame to the SCHEDULER

RUN

Running

4.7.4.1.4 State machine description

The W-START state indicates that an initialization is needed. The Activate service sets the machine in the RUN state waiting for Provider Data service requests and Time events. Receiving the confirmation will bring back the machine to the RUN state. An error or a Close service request is needed to re-enter the W-START state.

Each PPM handles up to two frames with one or two FrameIDs. For RTC3 compatibility two frames with the same FrameID and for MRPD two frames with two FrameIDs (FrameID and FrameID+1) are handled.

It is used to setup the frames which are transmitted by the Scheduler. The content (Buffer_Data) contains the local data if "frame structure" is used or the local data and the stored subframes (DFP_STORAGE) if "subframe structure" is used.

The Buffer_Data is fetched first from the INBOUND DFP frame and if this frame is too late from the DFP_STORAGE. If the concurrently received and transmitted subframe is detected as invalid, the frame should be shortened adding a TX-error (see Annex S) at the transmitting port.

NOTE The TagControlInformation.VLAN_Id is set to zero by the PPM but can be set different elsewhere according to IEEE 802.1Q.

4.7.4.1.5 PPM state table

Table 179 contains the description of the PPM state table.

Table 179 – PPM state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	PPM_Activate_req () => FirstTransmit := FALSE Buffer_Data := Default_Values Buffer_Status := Default_Status SCHEDULER_add.req () PPM_Activate_cnf (+)	RUN
2	W-START	PPM_Set_Data_req (Data) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Data_cnf (-)	W-START
3	W-START	PPM_Set_Status_req (Status) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Status_cnf (-)	W-START
4	W-START	PPM_Close_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Close_cnf (-)	W-START
5	W-START	Scheduler_transmit.ind () => ignore	W-START
6	W-START	Scheduler_add.cnf (+) => ignore	W-START
7	W-START	Scheduler_remove.cnf (+) => ignore	W-START
8	W-START	Scheduler_add.cnf (-) => ignore	W-START
9	W-START	Scheduler_remove.cnf (-) => ignore	W-START
10	RUN	PPM_Activate_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Activate_cnf (-)	RUN
11	RUN	PPM_Set_Data_req (Data) => Buffer_Data := Data PPM_Set_Data_cnf (+)	RUN
12	RUN	PPM_Set_Status_req (Status) => Buffer_Status := Status PPM_Set_Status_cnf (+)	RUN
13	RUN	PPM_Close_req () => SCHEDULER_remove.req () PPM_Close_cnf (+)	W-START
14	RUN	Scheduler_transmit.ind () /FirstTransmit == FALSE => FirstTransmit := TRUE PPM_State_ind (Start)	RUN

#	Current State	Event /Condition =>Action	Next State
15	RUN	Scheduler_transmit.ind () /FirstTransmit == TRUE => ignore	RUN
16	RUN	Scheduler_add.cnf (+) => ignore	RUN
17	RUN	Scheduler_remove.cnf (+) => ignore	RUN
18	RUN	Scheduler_add.cnf (-) => ERRCLS := CTXT ERRCODE := INVALID PPM_State_ind (Error) PPM_Error_ind (Error)	W START
19	RUN	Scheduler_remove.cnf (-) => ignore	RUN

4.7.4.1.6 Functions, Macros, Timers and Variables

Table 180 contains the functions, macros, timers and variables used by the PPM and their arguments and their descriptions.

Table 180 – Functions, Macros, Timers and Variables used by the PPM

Name	Type	Function/Meaning
FirstTransmit	Variable	This local variable contains the information whether the frame was at least one time conveyed.
Buffer_Data	Variable	This local variable contains the data (encoded as DataItem*) which shall be used as storage of the data dedicated for conveyance of the next Data-RTC-PDU or UDP-RTC-PDU.
Buffer_Status	Variable	This local variable contains the status (encoded as APDU_Status.DataStatus) which shall be used as storage of the status dedicated for conveyance of the next Data-RTC-PDU or UDP-RTC-PDU.
NOTE DataItem* includes all DataItems for the dedicated PDU.		

4.7.4.1.7 Truth tables

Table 181 contains the valid combinations, their meaning and their description of TxOption.

Table 181 – Truth table used by the PPM for TxOption

RT_CLASS_X	ARProperties. StartupMode	IRT	MRPD	DFP	Description
		FrameSendOffset, Port	FrameID, FrameSendOffset, Port	SFPosition	
RT_CLASS_1, RT_CLASS_2, RT_CLASS_UDP	—	—	—	—	PPM shall create one frame
RT_CLASS_3	Legacy	Values given	—	—	PPM shall create two frames. During startup (see Annex B) one frame with the given values from the connect service and one frame with RR=128 only send in the GREEN PERIOD. After ReadyForRTC3 the additional (GREEN) frame shall be deleted.
RT_CLASS_3	Legacy	Values given	Values given	—	Not supported
RT_CLASS_3	Legacy	Values given	—	Values given	Not supported
RT_CLASS_3	Legacy	Values given	Values given	Values given	Not supported
RT_CLASS_3	Advanced	Values given	—	—	PPM shall create one frame (see Annex A)
RT_CLASS_3	Advanced	Values given	Values given	—	PPM shall create two frames (see Annex A and Annex Q)
RT_CLASS_3	Advanced	Values given	—	Values given	PPM shall create one frame (see Annex A)
RT_CLASS_3	Advanced	Values given	Values given	Values given	PPM shall create two frames (see Annex A and Annex Q)

4.7.4.2 Consumer protocol machine

4.7.4.2.1 General

Figure 56 shows the basic structure of the Consumer protocol machine (CPM).

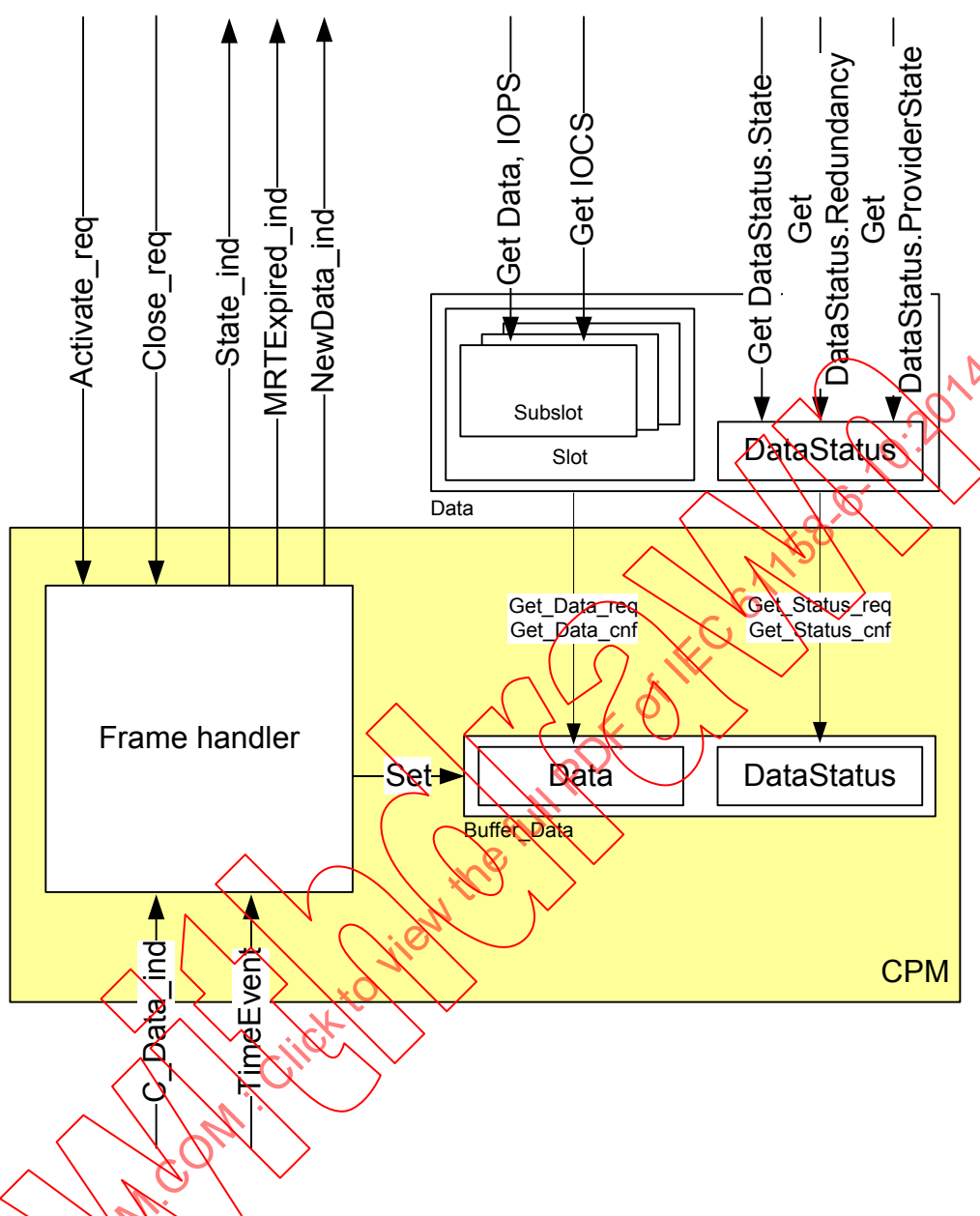


Figure 56 – Basic structure of a CPM

4.7.4.2.2 Primitive definitions

4.7.4.2.2.1 Primitives exchanged between remote machines

The remote service primitives including their associated parameters issued or received by CPM are described in the service definition and shown in Table 182.

Table 182 – Remote primitives issued or received by CPM

Primitive	Source	Destination	Associated parameters	Functions
LMPM_C_Data.ind	LMPM	CPM	CREP, Port, DA, SA, Prio, C_SDU, APDU_Status	—

Primitive	Source	Destination	Associated parameters	Functions
UDP_C_Data.ind	IP	CPM	CREP, IPPort, DA, SA, Prio, C_SDU, APDU_Status	—

4.7.4.2.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CPM are described in the service definition and shown in Table 185.

Table 183 – Local primitives issued or received by CPM

Primitive	Source	Destination	Associated parameters	Functions
CPM_Activate_cnf (-)	CPM	CMSU CMDMC CTLSU	CREP, ERRCLS, ERRCODE	—
CPM_Activate_cnf (+)	CPM	CMSU CMDMC CTLSU	CREP	—
CPM_Activate_req	CMSU CMDMC CTLSU	CPM	CREP, DA, SA, FrameID, RxOption, Exp_Length, DataHoldFactor, Default_Value, Default_Status	—
CPM_Close_cnf (+)	CPM	CMSU CMDMC CTLSU	CREP	—
CPM_Close_req	CMSU CMDMC CTLSU	CPM	CREP	—
CPM_Error_ind	CPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	Locally generated event if something strange happens
CPM_Get_Data_cnf (-)	CPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
CPM_Get_Data_cnf (+)	CPM	FSPMDEV FSPMCTL	CREP, Data, New_Flag	—
CPM_Get_Data_req	FSPMDEV FSPMCTL	CPM	CREP	—
CPM_Get_Status_cnf (-)	CPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
CPM_Get_Status_cnf (+)	CPM	FSPMDEV FSPMCTL	CREP, Status, RecvCounter	—
CPM_Get_Status_req	FSPMDEV FSPMCTL	CPM	CREP	—
CPM_MRTEExpired_ind	CPM	Diagnosis Module	info	—

Primitive	Source	Destination	Associated parameters	Functions
CPM_NewData_ind	CPM	FSPMDEV FSPMCTL CMIO CTLIO CMDMC	AREP, CREP, APDU_Status, data {Data, NoData}	—
CPM_State_ind	CPM	CMIO CTLIO CMDMC	state {Start, Stop}	—

4.7.4.2.3 State transition diagram

The state transition diagram of the CPM is shown in Figure 57.

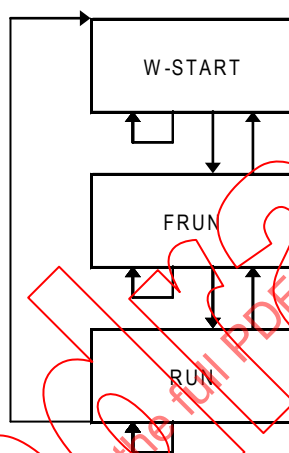


Figure 57 – State transition diagram of CPM

States of the CPM

W-START

The W-START state indicates that the initialization is needed. The Activate service sets the machine in the RUN state or in the FRUN (first run) state.

FRUN

The state machine is waiting for the first C_Data_ind service indication in these states. The DataHoldTime is ignored in this state. After passing the Start indication to the user the RUN state is entered.

RUN

In this state the monitoring using the DataHoldtime is active. A timeout will force a state transition back to FRUN (combined with a Stop indication). A Close service request is needed to reenter the W-START state.

4.7.4.2.4 State machine description

The W-START state indicates that the initialization is needed. The Activate service sets the machine in the RUN state or in the FRUN (first run) state. The state machine is waiting for the first LMPM_C_Data service indication in these states. After passing the Start indication to the user the RUN state is entered.

A timeout will force a state transition back to FRUN (combined with a Stop indication). A Close service request is needed to reenter the W-START state.

Each time receiving valid C_Data.ind the time is read and stored in a local variable together with a flag.

Each CPM handles up to two frames with one or two FrameIDs. For RTC3 compatibility two frames with the same FrameID and for MRPD two frames with two FrameIDs (FrameID and FrameID+1) are handled.

4.7.4.2.5 CPM state table

Table 184 contains the complete description of the CPM state table.

Table 184 – CPM state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	CPM_Activate_req () => Store DA, SA, FrameID, RxOption, DataHoldFactor, Default_Value, Default_Status New_Data := FALSE for all ports of interface do: RxA(Port, FrameID) := Invalid Dht := 0 Cycle := 0 RecvCnt := 0 Buffer_Data := Default_Value Buffer_Status := Default_Status Primary := TRUE ControllInterval := SendClockFactor x ReductionRatio x 31,25 µs StartTimer (ControllInterval) CPM_Activate_cnf (+)	FRUN
2	W-START	CPM_Close_req () => StopTimer (ControllInterval) CPM_Close_cnf (+)	W-START
3	W-START	CPM_Get_Status_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Get_Status_cnf (-)	W-START
4	W-START	CPM_Get_Data_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Get_Data_cnf (-)	W-START
5	W-START	C_Data_ind => ignore	W-START
6	FRUN	CPM_Activate_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Activate_cnf (-)	FRUN
7	FRUN	CPM_Close_req () => StopTimer () CPM_Close_cnf (+)	W-START
8	FRUN	CPM_Get_Status_req () => Status := Cycle, Buffer_Status RecvCounter := RecvCnt RecvCnt := 0 CPM_Get_Status_cnf (+)	FRUN
9	FRUN	CPM_Get_Data_req () => Data := Buffer_Data New_Flag := New_Data New_Data := FALSE	FRUN

#	Current State	Event /Condition =>Action	Next State
		CPM_Get_Data_cnf (+)	
10	FRUN	C_Data_ind /FrameGuard () == INVALID => ignore	FRUN
11	FRUN	C_Data_ind /FrameGuard () == VALID, DHTReload, UpdateData => New_Data := TRUE RxA(Port, FrameID) := APDU_Status.Cycle_Counter Dht := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Data := Filter (Data) Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_State_ind (Start) CPM_NewData_ind (Data) If MRPD active then StartTimer (MRPDt) //NOTE DataStatus.State == don't care and DataStatus.DataValid == Valid	RUN
12	FRUN	C_Data_ind /FrameGuard () == VALID, DHTReload => RxA(Port, FrameID) := APDU_Status.Cycle_Counter Dht := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_State_ind (Start) CPM_NewData_ind (NoData) If MRPD active then StartTimer (MRPDt) //NOTE DataStatus.State == Backup	RUN
13	FRUN	TimerExpired (ControlInterval) => ignore	FRUN
14	FRUN	TimerExpired (MRWDt) => ignore	FRUN
15	RUN	CPM_Activate_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Activate_cnf (-)	RUN
16	RUN	CPM_Close_req () => StopTimer (ControlInterval) CPM_Close_cnf (+)	W- START
17	RUN	CPM_Get_Status_req () => Status := Cycle, Buffer_Status, RecvCounter := RecvCnt, RecvCnt := 0 CPM_Get_Status_cnf (+)	RUN
18	RUN	CPM_Get_Data_req () => Data := Buffer_Data, New_Flag := New_Data New_Data := FALSE CPM_Get_Data_cnf (+)	RUN
19	RUN	C_Data_ind /FrameGuard () == INVALID =>	RUN

#	Current State	Event /Condition =>Action	Next State
		ignore	
20	RUN	C_Data_ind /FrameGuard () == VALID, DHTReload, UpdateData ==> New_Data := TRUE, RxA(Port, FrameID) := APDU_Status.Cycle_Counter DHt := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Data := Filter (Data) Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_NewData_ind (Data) //NOTE DataStatus.State == don't care and DataStatus.DataValid == Valid	RUN
21	RUN	C_Data_ind /FrameGuard () == VALID, DHTReload ==> RxA(Port, FrameID) := APDU_Status.Cycle_Counter DHt := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_NewData_ind (NoData) //NOTE DataStatus.State == Backup	RUN
22	RUN	TimerExpired (ControllInterval) /DHt > (DataHoldFactor-1) ==> DHt := 0 CPM_State_ind (Stop) //NOTE DHt expired	FRUN
23	RUN	TimerExpired (ControllInterval) /DHt < (DataHoldFactor) ==> DHt := DHt + 1	RUN
24	RUN	TimerExpired (MRWDt) /MRPD active && MRWDt expired ==> CPM_MRTEexpired_ind (Fault)	RUN
25	RUN	TimerExpired (MRWDt) /MRPD active && /MRPDGuard () == Adjunctive frame received ==> for all ports, FrameIDs RxA(Port, FrameID) := Invalid StopTimer (MRWDt) CPM_MRTEexpired_ind (Ok)	RUN
26	RUN	TimerExpired (MRWDt) /MRPD active && /MRPDGuard () == No adjunctive frame received ==> for all ports, FrameIDs RxA(Port, FrameID) := Invalid StartTimer (MRWDt)	RUN

4.7.4.2.6 Functions, Macros, Timers and Variables

Table 185 contains the functions, macros, timers and variables used by the CPM and their arguments and their descriptions.

Table 185 – Functions, Macros, Timers and Variables used by the CPM

Name	Type	Function/Meaning
Filter	Function	This local function filters from the received data the to be stored portion for the Buffer_Data.
FrameGuard	Function	This local function checks the received frame against the expected structure and semantics according the coding and the truth tables. VALID / INVALID: This flag shows whether the received frame is valid. DHTReload: This flag shows, that the DataHoldTimer shall be retrigged. UpdateData: This flag shows, that Buffer_Data and Buffer_Status shall be updated with the received data.
MRPDGuard	Function	This local function checks whether the adjunctive frame defined by the activation of MRPD is received or not. It uses the local variable RxA as database.
StartTimer	Function	This local function starts or restarts a timer.
StopTimer	Function	This local function stops a timer.
TimerExpired	Function	This local function is issued if a timer expires.
C_Data_ind	Macro	if (RxOption == RTC) LMPM_C_Data.ind (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) if (RxOption == UDP) UDP_C_Data.ind (IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status)
RxOption	Macro	Contains the - RT_CLASS_x - FrameID, - if MRPD - FrameID (redundant path), - Media redundancy watchdog (MRPD active) - if DFP - subaddress SFPosition
ControllInterval	Timer	This local timer contains the monitoring interval for the DataHoldFactor. $\text{ControlInterval} = \text{SendClockFactor} \times \text{ReductionRatio} \times 31,25 \mu\text{s}$ This timer shall use a autoreload function to start itself after it expires. It shall be aligned to the PTCP or local driven SendClock.
MRWDt	Timer	This local timer contains the monitoring interval for the Media Redundancy WatchDog time. It is used to monitor the receiving of the adjunctive frames defined by MRPD and issue an event if expired.
Buffer_Data	Variable	This local variable contains the last valid service data unit conveyed with a LMPM_C_Data indication.
Buffer_Status	Variable	This local variable contains the last valid APDU_Status conveyed with a LMPM_C_Data indication.
Cycle	Variable	This local variable contains the last valid Cycle Counter conveyed with a LMPM_C_Data indication with valid data.
Default_Status	Variable	The CycleCounter is set to invalid, the TransferStatus and the DataStatus is set to zero.
Default_Value	Variable	The Data is set to zero and the IOxS to BAD.
DHt	Variable	This local variable contains the DataHoldFactor counter value of the time events since last LMPM_C_Data indication.
ERRCLS	Variable	This local variable contains the entity signaling the error.
ERRCODE	Variable	This local variable contains the error reason in the context of the ERRCLS.
New_Data	Variable	This local variable signals the receipt of a valid LMPM_C_Data indication (set to FALSE by a GetData request).
RecvCnt	Variable	This local variable contains the counter value of the received cyclic PDU since last GetStatus request.

Name	Type	Function/Meaning
RxA	Variable	This local variable contains the stored information about the received frames for the MRPD checking. Principle: $RxA(m, n) == RxA(o, n+1)$ // Stored APDU_Status.CycleCounter are equal // m, o element {1..MaxPort} // n, n+1 element {FrameID, FrameID+1}

Table 186 contains the valid combinations, their meaning and their description for RxOption.

Table 186 – Truth table used by the CPM for RxOption

RT_CLASS_X	ARProperties. StartupMode	MRPD	DFP	Description
		FrameID, Redundancy watchdog	SFPosition	
RT_CLASS_1, RT_CLASS_2, RT_CLASS_UDP	—	—	—	CPM shall create one consumer for one frame
RT_CLASS_3	FALSE	—	—	CPM shall create one consumer for one frame. During startup (see Annex B) the DHT values are for RR=256 and after ReadyForRTC3 as given during the connect service.
RT_CLASS_3	FALSE	Values given	—	Not supported
RT_CLASS_3	FALSE	—	Values given	Not supported
RT_CLASS_3	FALSE	Values given	Values given	Not supported
RT_CLASS_3	TRUE	—	—	CPM shall create one consumer for one frame (see Annex A)
RT_CLASS_3	TRUE	Values given	—	CPM shall create one consumer for two frames (see Annex A and Annex Q)
RT_CLASS_3	TRUE	—	Values given	CPM shall create one consumer for one frame (see Annex A)
RT_CLASS_3	TRUE	Values given	Values given	CPM shall create one consumer for two frames (see Annex A and Annex Q)

4.7.4.2.7 Truth tables

4.7.4.2.7.1 CPM checking rules for one frame

Table 187, Table 188, Table 189 and Table 190 contains the truth table used by the CPM for frames.

Table 187 – Truth table for one frame using RT_CLASS_x

Input					Output
Received frame					Local
Received-InRED ^c	APDU_Status. TransferStatus	SourceMAC- Address ^b	FrameID, redundant FrameID ^a	C_SDU. Length	Frame structure
FALSE	—	—	—	—	Invalid

Input					Output
Received frame					Local
TRUE	InValid	—	—	—	Invalid
TRUE	Valid	Wrong	—	—	Invalid
TRUE	Valid	Ok	Wrong	—	Invalid
TRUE	Valid	Ok	Ok	Wrong	Invalid
TRUE	Valid	Ok	Ok	Ok	Valid

^a The SFPosition.Position (different frame layout) is managed by the DFP_RELAY.

^b For frames using DFP optional.

^c Only checked for RT_CLASS_3.

Table 188 – Truth table for one frame using RT_CLASS_UDP

Input					Output
Received frame					Local
APDU_Status. TransferStatus	SourceIP- Address	SourceIPPort	FrameID, redundant FrameID	C_SDU Length	Frame structure
Invalid	—	—	—	—	Invalid
Valid	Wrong	—	—	—	Invalid
Valid	Ok	Wrong	—	—	Invalid
Valid	Ok	Ok	Wrong	—	Invalid
Valid	Ok	Ok	Ok	Wrong	Invalid
Valid	Ok	Ok	Ok	Ok	Valid

Table 189 – Truth table for the C_SDU

Input	Output
Received frame	Local
APDU_Status.CycleCounter	C_SDU structure
Older ^a	Invalid
Newer ^b	Valid

^a See 4.7.2.1.2 and Table 161. When activating a CPM, the stored CycleCounterValue is set to invalid to make the next received frame in all cases "Newer".

^b See 4.7.2.1.2 and Table 161.

Table 190 – Truth table for arranging Dht and data

Input				Output		
Received frame				Local		
Frame structure	C_SDU structure	APDU_Status. DataStatus. DataValid	APDU_Status. DataStatus. State	Datahold timer	Update data	FrameGuard
Invalid	—	—	—	—	No	Invalid
Valid	Invalid	—	—	—	No	Invalid
Valid	Valid	Invalid	Primary	—	No	Invalid
Valid	Valid	Invalid	Backup	Reload	No	Valid
Valid	Valid	Valid	Primary	Reload	Yes	Valid
Valid	Valid	Valid	Backup	Reload	Yes	Valid

4.7.4.2.7.2 CPM checking rules for one subframe

Table 191, Table 192, Table 193 and Table 194 contains the truth table used by the CPM for subframes.

Table 191 – Truth table for the subframe – frame check

Input			Output
FrameID	SourceMAC-address	Header Subframe. SFCRC16	Frame check
FALSE	—	—	Invalid
TRUE	FALSE	—	Invalid
TRUE	TRUE	Invalid	Invalid
TRUE	TRUE	Valid	Valid
TRUE	TRUE	Don't care	Valid

Table 192 – Truth table for the subframe – sub frame check

Input			Output
Subframe. Position	Subframe. Length	Subframe. SFCRC16	Sub frame check
FALSE	—	—	Invalid
TRUE	FALSE	—	Invalid
TRUE	TRUE	Invalid	Invalid
TRUE	TRUE	Valid	Valid
TRUE	TRUE	Don't care	Valid

Table 193 – Truth table for the subframe – sub frame data check

Input		Output
Subframe. DataStatus. Ignore	Subframe. SFCycleCounter	Sub frame data check
Ignore	—	Invalid
Ok	OLD	Invalid
Ok	NEW	Valid

Table 194 – Truth table for the subframe – Dht and data

Input					Output		
Received frame					Local		
Frame check	Sub frame check	Sub frame data check	Subframe. DataStatus. Valid	Subframe. DataStatus. State	Datahold timer	Update data	Frame- Guard
Invalid	—	—	—	—	—	No	Invalid
Valid	Invalid	—	—	—	—	No	Invalid
Valid	Valid	Invalid	—	—	—	No	Invalid
Valid	Valid	Valid	Invalid	Backup	Reload	No	Valid
Valid	Valid	Valid	Valid	Primary	Reload	Yes	Valid
Valid	Valid	Valid	Valid	Backup	Reload	Yes	Valid

4.7.5 DLL Mapping Protocol Machines

There is no DLL Mapping Protocol Machine (DMPM) defined for this Protocol.

4.8 Real time acyclic

4.8.1 RTA syntax description

4.8.1.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.8.1.2 RTA APDU abstract syntax

Table 195 defines the abstract syntax of the Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 195 shall represent the content of the DLSDU in Table 5.

Table 195 – RTA APDU syntax

APDU name	APDU structure
RTA-PDU	DATA-RTA-PDU ^ ACK-RTA-PDU ^ NACK-RTA-PDU ^ ERR-RTA-PDU
UDP-RTA-PDU	IPHeader, UDPHeader, FrameID, RTA-PDU
NOTE The DLPDU_Padding applies for the RTA-PDU and the UDP-RTA-PDU.	

Table 196 defines structures for substitutions of elements of the APDU structures shown in Table 195.

Table 196 – RTA substitutions

Substitution name	Structure
Reference	DestinationServiceAccessPoint, SourceServiceAccessPoint
Destination-ServiceAccess-Point	AlarmEndpoint
Source-ServiceAccess-Point	AlarmEndpoint
FlagsSequence	AddFlags, SendSeqNum, AckSeqNum
VendorDevice-ErrorInfo	VendorID, DeviceID, Data*
DATA-RTA-PDU	Reference, PDUType (=1), FlagsSequence, VarPartLen (1 – 1 432) ^a , RTA-SDU
ACK-RTA-PDU	Reference, PDUType (=3), FlagsSequence, VarPartLen (=0)
NACK-RTA-PDU	Reference, PDUType (=2), FlagsSequence, VarPartLen (=0)
ERR-RTA-PDU	Reference, PDUType (=4), FlagsSequence, VarPartLen (=4), PNIStatus, [VendorDeviceErrorInfo] ^b
^a The maximum VarPartLen is calculated to fit in the RTA-PDU and in the UDP-RTA-PDU.	
^b The number of octets of this substitution shall not exceed the DLPDU Padding. See minimum frame size in IEEE 802.3	

4.8.2 RTA transfer syntax

4.8.2.1 Coding section related to RTA-PDUs specific fields

4.8.2.1.1 Coding of the field AlarmEndpoint

This field shall be coded as data type Unsigned16. It identifies, in combination with the Host-Address, the service access points of a bidirectional connection as shown in Figure 58. The values for source and destination shall be exchanged during a context management operation.

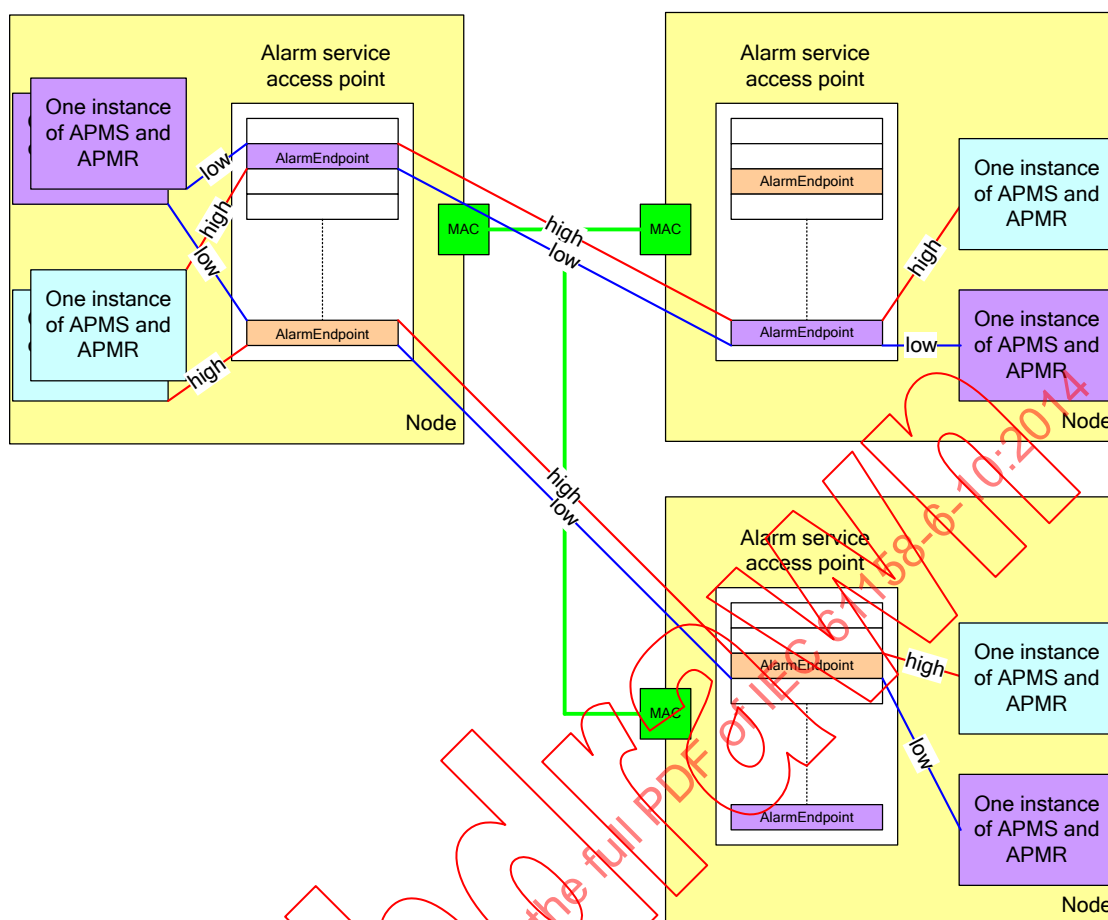


Figure 58 – Addressing scheme of RTA

NOTE The IO controller send its AlarmEndpoint during the connect to the IO device using the field LocalAlarmReference in the AlarmCRBlockReq. The IO device responds with its AlarmEndpoint using the field LocalAlarmReference in the AlarmCRBlockRes.

The IO controller uses its AlarmEndpoint as SourceServiceAccessPoint and the AlarmEndpoint of the IO devices as DestinationServiceAccessPoint. The IO device uses its AlarmEndpoint as SourceServiceAccessPoint and the AlarmEndpoint of the IO controller as DestinationServiceAccessPoint.

4.8.2.1.2 Coding of the field PDUType

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0 – 3: PDUType.Type

This field shall be coded with the values according to Table 197.

Table 197 – PDUType.Type

Value (hexadecimal)	Meaning	Usage
0x00	Reserved	—
0x01	RTA_TYPE_DATA	Shall only be used to encode the DATA-RTA-PDU
0x02	RTA_TYPE_NACK	Shall only be used to encode the NACK-RTA-PDU
0x03	RTA_TYPE_ACK	Shall only be used to encode the ACK-RTA-PDU

Value (hexadecimal)	Meaning	Usage
0x04	RTA_TYPE_ERR	Shall only be used to encode the ERR-RTA-PDU
0x05 – 0x0F	Reserved	—

Bit 4 – 7: PDUType.Version

This field shall be coded with the values according to Table 198.

Table 198 – PDUType.Version

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Version 1 of the protocol
0x02 – 0x0F	Reserved

4.8.2.1.3 Coding of the field AddFlags

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0 – 3: AddFlags.WindowSize

This field shall be coded with the values according to Table 199

Table 199 – AddFlags.WindowSize

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Window size one
0x02 – 0x07	Reserved

Bit 4: AddFlags.TACK

This field shall be coded with the values according to Table 200 by the sender within a RTA-PDU to control the acknowledge behavior of the receiver.

Table 200 – AddFlags.TACK

Value (hexadecimal)	Meaning	Usage
0x00	No immediate acknowledge	Shall be set for ERR-RTA-PDU, ACK-RTA-PDU and NACK-RTA-PDU and not checked by the receiver.
0x01	Immediate acknowledge	Shall be set for immediate acknowledge for DATA-RTA-PDU.

Bit 5 – 7: AddFlags.reserved

This field shall be set according to 3.4.2.2.

4.8.2.1.4 Coding of the field SendSeqNum

This field shall be coded as data type Unsigned16 with values according to Table 201.

Table 201 – SendSeqNum

Value (hexadecimal)	Usage
0x0000 – 0x7FFF	The first issued DATA-RTA-PDU shall contain the value zero.
other	Reserved
0xFFFFE	This value shall be used to synchronize sender and receiver after establishment of the application relationship. This value indicates that there was no reception of a DATA-RTA-PDU before.
0xFFFF	This value shall be used to synchronize sender and receiver after establishment of the application relationship. This value indicates the first DATA-RTA-PDU.

This field contains the number of the DATA-RTA-PDU. The incrementing and comparison of this number is done using modulo 2^{15} operations.

NOTE The first DATA-RTA-PDU sets SendSeqNum=0xFFFF and AckSeqNum=0xFFFF. It is acknowledged with SendSeqNum=0xFFFFE and AckSeqNum=0xFFFF. The second DATA-RTA-PDU sets SendSeqNum=0 and AckSeqNum=0xFFFFE. The synchronization is necessary because the acyclic protocol does not define any connection monitoring.

4.8.2.1.5 Coding of the field AckSeqNum

This field shall be coded as data type Unsigned16 with values according to Table 202.

Table 202 – AckSeqNum

Value (hexadecimal)	Usage
0x0000 – 0x7FFF	The first issued DATA-RTA-PDU shall contain the value zero.
other	Reserved
0xFFFFE	This value shall be used to synchronize sender and receiver after establishment of the application relationship. This value indicates that there was no DATA-RTA-PDU received before.
0xFFFF	This value shall be used to synchronize sender and receiver after establishment of the application relationship. This value acknowledges the reception of the first DATA-RTA-PDU.

This field contains the number of the DATA-RTA-PDU that is expected to be acknowledged or which is acknowledged.

4.8.2.1.6 Coding of the field VarPartLen

This field shall be coded as data type Unsigned16 according to Table 203. The VarPartLen shall not contain the DLPDU_Padding.

Table 203 – VarPartLen

Value (hexadecimal)	Meaning
0x0000	No RTA-SDU exists
0x0001 – 0x0598	A RTA-SDU with VarPartLen octets exists
0x0599 – 0xFFFF	Reserved

4.8.3 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.8.4 Application Relationship Protocol Machines

4.8.4.1 General

The symmetrical RTA transport is used by the symmetrical Acyclic Protocol Machine (APM). The APM is built as shown in Figure 59. An ALPMI / ALPMR channel is addressed by Source-Node, Destination-Node, Source-SAP, Destination-SAP and FrameID high or low.

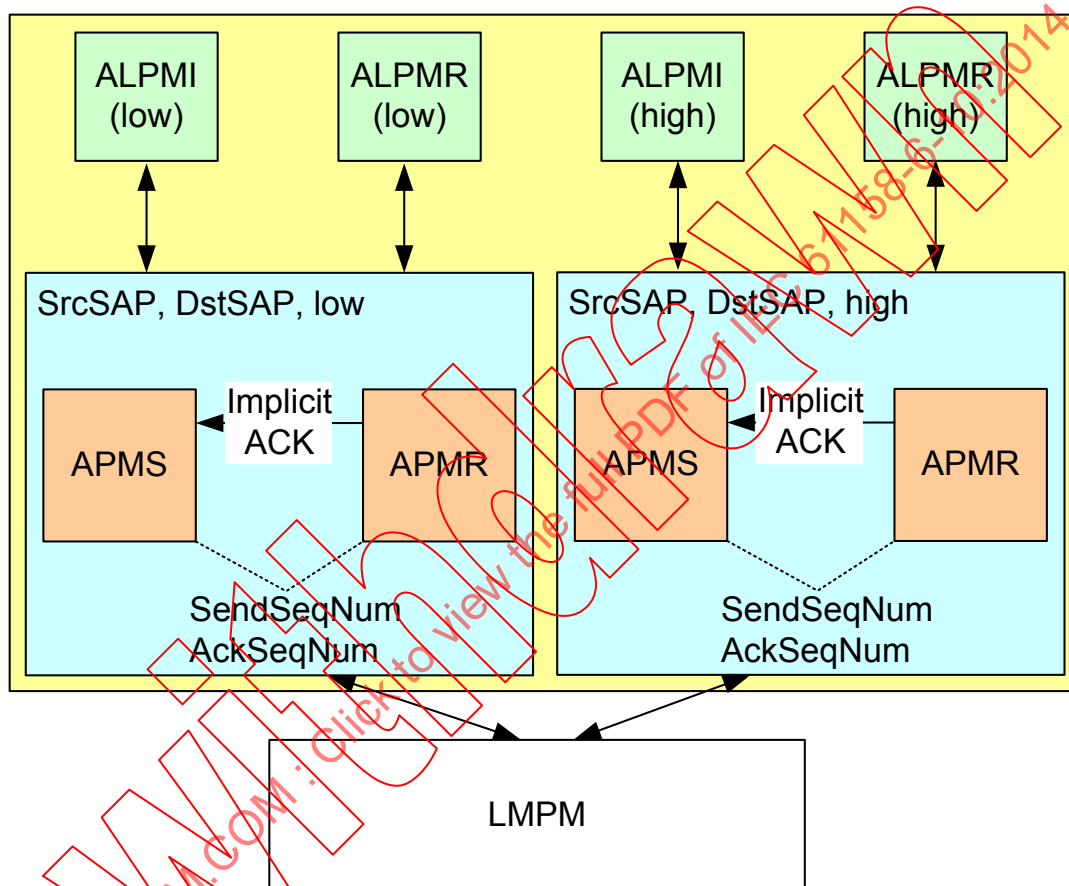


Figure 59 – Structure of the APM

4.8.4.2 Acyclic Protocol Machine Sender

4.8.4.2.1 General

In the following the structure of the Acyclic Protocol Machine Sender (APMS) is shown (see Figure 60).

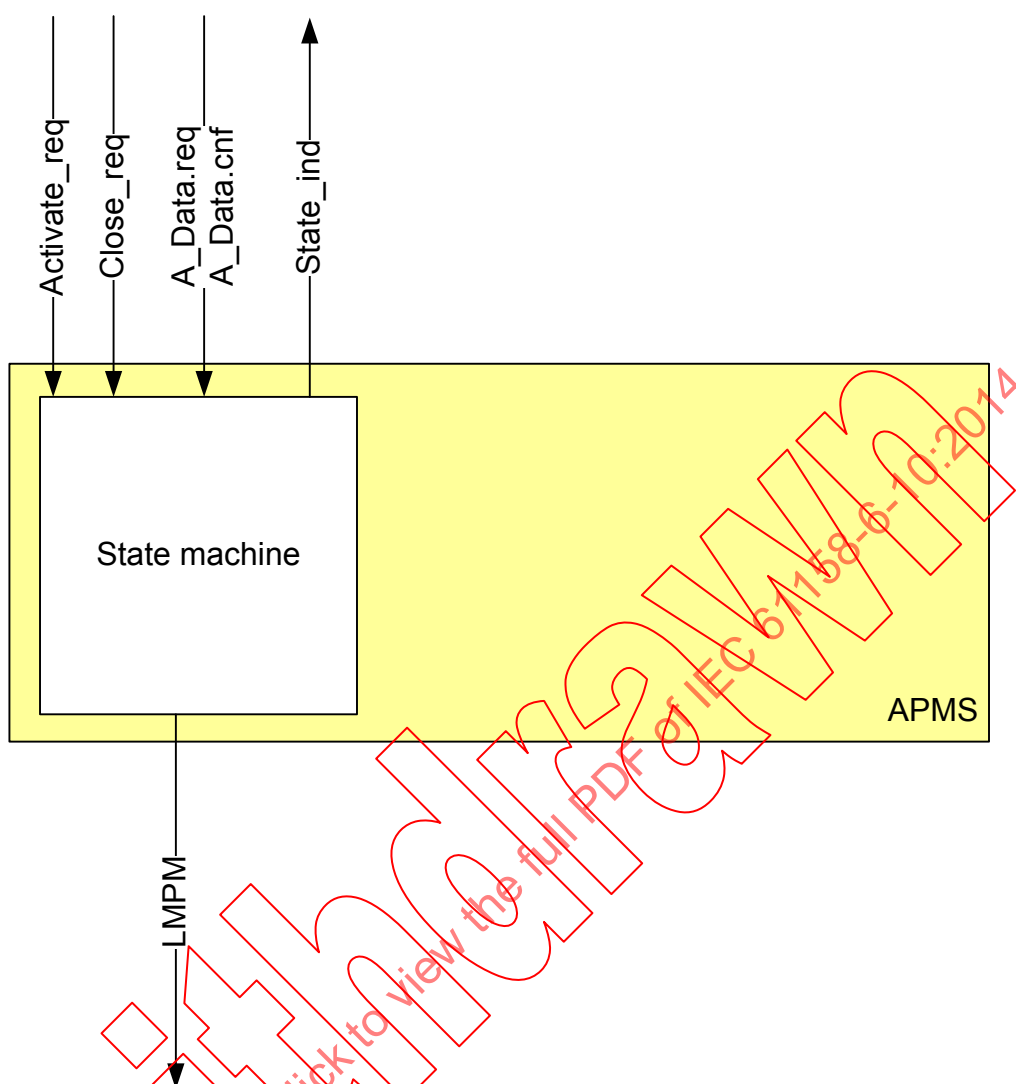


Figure 60 – Structure of the APMS

4.8.4.2.2 Primitive definitions

4.8.4.2.2.1 Primitives exchanged between remote machines

The remote service primitives including their associated parameters issued or received by APMS are described in the service definition and shown in Table 204.

Table 204 – Remote primitives issued or received by APMS

Primitive	Source	Destination	Associated parameters	Functions
APMS_A_Data.cnf (-)	APMS	ALPMI ALPMR	CREP, ERRCLS, ERRCODE	—
APMS_A_Data.cnf (+)	APMS	ALPMI ALPMR	CREP	—
APMS_A_Data.req	ALPMI ALPMR	APMS	CREP, Data	—
LMPM_A_Data.cnf	LMPM	APMS	CREP, LMPM_status	

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.ind	LMPM	APMS	CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_N_Data.cnf	LMPM	APMS	CREP, LMPM_status	—
LMPM_N_Data.ind	LMPM	APMS	CREP, DA, SA, VLANPrio, VLANId, N_SDU	—
LMPM_A_Data.req	APMS	LMPM	CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_N_Data.req	APMS	LMPM	CREP, DA, SA, VLANPrio, VLANId, N_SDU	—

4.8.4.2.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by APMS are described in the service definition and shown in Table 205.

Table 205 – Local primitives issued or received by APMS

Primitive	Source	Destination	Associated parameters	Functions
APMS_Activate_req	ALPMI ALPMR	APMS	CREP, DA, SA, FrameID, VLANPrio, VLANID, RTATimeoutFactor, RTARetries, Transport, DstIP, SrcIP	—
APMS_Close_req	ALPMI ALPMR	APMS	CREP, ErrCode	—
APMS_Close_cnf	APMS	ALPMI ALPMR	CREP	—
APMS_Activate_cnf (-)	APMS	ALPMI ALPMR	CREP, ERRCLS, ERRCODE	—
APMS_Activate_cnf (+)	APMS	ALPMI ALPMR	CREP	—
APMS_Error_ind	APMS	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—

4.8.4.2.3 State transition diagram

The state transition diagram of the APMS is shown in Figure 61.

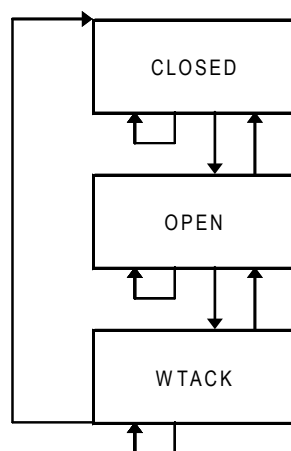


Figure 61 – State transition diagram of APMS

States of the APMS

CLOSED	The CLOSED state indicates that an initialization is needed. Activate service sets the machine in the OPEN state.
OPEN	Waiting for A-Data service request.
WTACK	After issuing the transmission of data the WTACK state is used to wait for a transport acknowledge. Receiving the acknowledge will set back the machine to the OPEN state.

4.8.4.2.4 State machine description

This state machine, addressed by Source-Host, Source-ServiceAccessPoint, Destination-Host, DestinationAccessPoint and Priority, handles the A-Data service. Each priority (High and Low) is handled independent. This state machine is part of the APMS and APMR pair which is used as sender and responder for the transport acknowledge A-Data services. The A-Data service are used for the Alarm transport.

4.8.4.2.5 APMS state table

Table 206 contains the complete description of the APMS state machine. A LMPM_A_Data.ind primitive will be accepted if the value of the PDUType.Type field is DATA-RTA-PDU, ACK-RTA-PDU or NACK-RTA-PDU and the PDUType.Version is RTA_VERS=1.

The APMS state machine specify the abort sequence and shows the corresponding ERR-RTA-PDU. These PDU shall only be sent from the APMS for low priority alarms from both sides of the connection.

Table 206 – APMS state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	APMS_Activate.req () => AlarmInstance[SrcSAP, DstSAP, Prio] := DA, RTATimeoutFactor, RTARetries, ... // NOTE Information used for ALPMI, ALPMR, APMS and APMR Send_Seq_Count:=0xffff Send_Seq_CountO:=0xfffe Use CREP to identify the AlarmInstance APMS_Activate.cnf (+)	OPEN
2	CLOSED	APMS_Close.req () => APMS_Close.cnf ()	CLOSED
3	CLOSED	APMS_A_Data.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_A_Data.cnf (-)	CLOSED
4	CLOSED	A_Data_ind => ignore	CLOSED
5	CLOSED	A_Data_cnf => ignore	CLOSED
6	OPEN	APMS_Activate.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_Activate.cnf (-)	OPEN
7	OPEN	APMS_Close.req () /VLANPrio == Low => PDU.Version:=1 PDU.PDUType:=ERR PDU.AddFlags.TACK:=0 PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.PNIOStatus.ErrorCode = 0xCF PDU.PNIOStatus.ErrorDecode = 0x81 PDU.PNIOStatus.ErrorCode1 = 0xFD PDU.PNIOStatus.ErrorCode2 = ErrCode from APMS_Close request A_Data_req StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
8	OPEN	APMS_Close.req () /VLANPrio == High => //NOTE Error PDU only for the APMS instance "low" StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
9	OPEN	APMS_A_Data.req () => PDU.Version:=1 PDU.PDUType:=DATA PDU.AddFlags.TACK:=Tack PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.RTA-SDU:=Data Store PDU.Flags, PDU.RTA-SDU Retry:= RTARetries A_Data_req	WTACK

#	Current State	Event /Condition =>Action	Next State
10	OPEN	A_Data_ind /PDU.PDUType == DATA PDU.PDUType == NAK PDU.PDUType == ACK => ignore	OPEN
11	OPEN	A_Data_ind /PDU.Type == ERR => ERRCLS := PDU.PNIOStatus.ErrorCode1 ERRCODE := PDU.PNIOStatus.ErrorCode2 APMS_Error_ind ()	OPEN
12	OPEN	A_Data_cnf => ignore	OPEN
13	OPEN	TimerExpired => ignore	OPEN
14	WTACK	APMS_Activate.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_Activate.cnf (-)	WTACK
15	WTACK	APMS_Close.req () /VLANPrio == Low => PDU.Version:=1 PDU.PDUType:=ERR PDU.AddFlags:=0 PDU.Window:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.PNIOStatus.ErrorCode = 0xCF PDU.PNIOStatus.ErrorDecode = 0x81 PDU.PNIOStatus.ErrorCode1 = 0xFD PDU.PNIOStatus.ErrorCode2 = ErrCode from APMS_Close request A_Data_req StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
16	WTACK	APMS_Close.req () /VLANPrio == High => StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
17	WTACK	APMS_A_Data.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_A_Data.cnf (-)	WTACK
18	WTACK	A_Data_ind /(PDU.PDUType == DATA PDU.PDUType == ACK) && PDU.AckSeqNum == Send_Seq_Count => Send_Seq_CountO:=Send_Seq_Count Send_Seq_Count:= (Send_Seq_Count+1) & 0x7fff //NOTE TACK shall be done by ACK or DATA APMS_A_Data.cnf (+)	OPEN
19	WTACK	A_Data_ind /(PDU.PDUType == DATA PDU.PDUType == ACK) && PDU.AckSeqNum != Send_Seq_Count => ignore	WTACK
21	WTACK	A_Data_ind /PDU.Type == ERR => APMS_Error_ind ()	WTACK

#	Current State	Event /Condition =>Action	Next State
21	WTACK	A_Data_ind /PDU.PDUType == NAK => ignore	WTACK
22	WTACK	A_Data_cnf => StartTimer (RTATimeout)	WTACK
23	WTACK	TimerExpired / Retry != 0 => PDU.Version:=1 PDU.PDUType:=DATA PDU.AddFlags.TACK:=from Stored PDU.AddFlags PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.RTA-SDU:=from Stored PDU.RTA-SDU Retry:= Retry-1 A_Data_req	WTACK
24	WTACK	TimerExpired /Retry == 0 => ERRCLS := APMS ERRCODE := TIMEOUT APMS_Error_ind () APMS_A_Data.cnf (-)	OPEN

4.8.4.2.6 Functions, Macros, Timers and Variables

Table 207 contains the functions, macros, timers and variables used by the APMS, their arguments and their descriptions.

Table 207 – Functions, Macros, Timers and Variables used by the APMS

Name	Type	Function/Meaning
StartTimer	Function	This function is used to start or restart a timer.
StopTimer	Function	This function is used to stop a timer.
TimerExpired	Function	This function signals that a timer is expired.
A_Data_cnf	Macro	if (Transport == RTC) LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) if (Transport == UDP) LMPM_N_Data.cnf (CREP, D_Port, Tstamp, LMPM_status)
A_Data_ind	Macro	if (Transport == RTC) LMPM_A_Data.ind (CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU) PDU := A_SDU if (Transport == UDP) LMPM_N_Data.ind (CREP, DA, SA, VLANPrio, VLANId, N_SDU) PDU := N_SDU
A_Data_req	Macro	if (Transport == RTC) A_SDU := PDU LMPM_A_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, A_SDU) if (Transport == UDP) N_SDU := PDU LMPM_N_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, N_SDU)

Name	Type	Function/Meaning
RTATimeout	Timer	This timer monitors the transport acknowledge from the responder
ERRCLS	Variable	This local variable contains the entity signaling the error.
ERRCODE	Variable	This local variable contains the error reason in the context of the ERRCLS.
Expected_Seq_CountO	Variable	This local variable contains the previous counter value of the Seq_Count variable.
RTARetries	Variable	This local variable contains the maximum retry factor from RTA ASE. The default value is 3.
Retry	Variable	This local variable is used as a working counter for the retry.
RTATimeoutFactor	Variable	This local variable contains the value from the RTA ASE. The default value is 100 ms.
Send_Seq_Count	Variable	This local variable contains the counter value which shall be used at the conveyance of the next DATA-RTA-PDU.

4.8.4.3 Acyclic Protocol Machine Receiver

4.8.4.3.1 General

In the following the structure of the Acyclic Protocol Machine Receiver (APMR) is shown (see Figure 62).

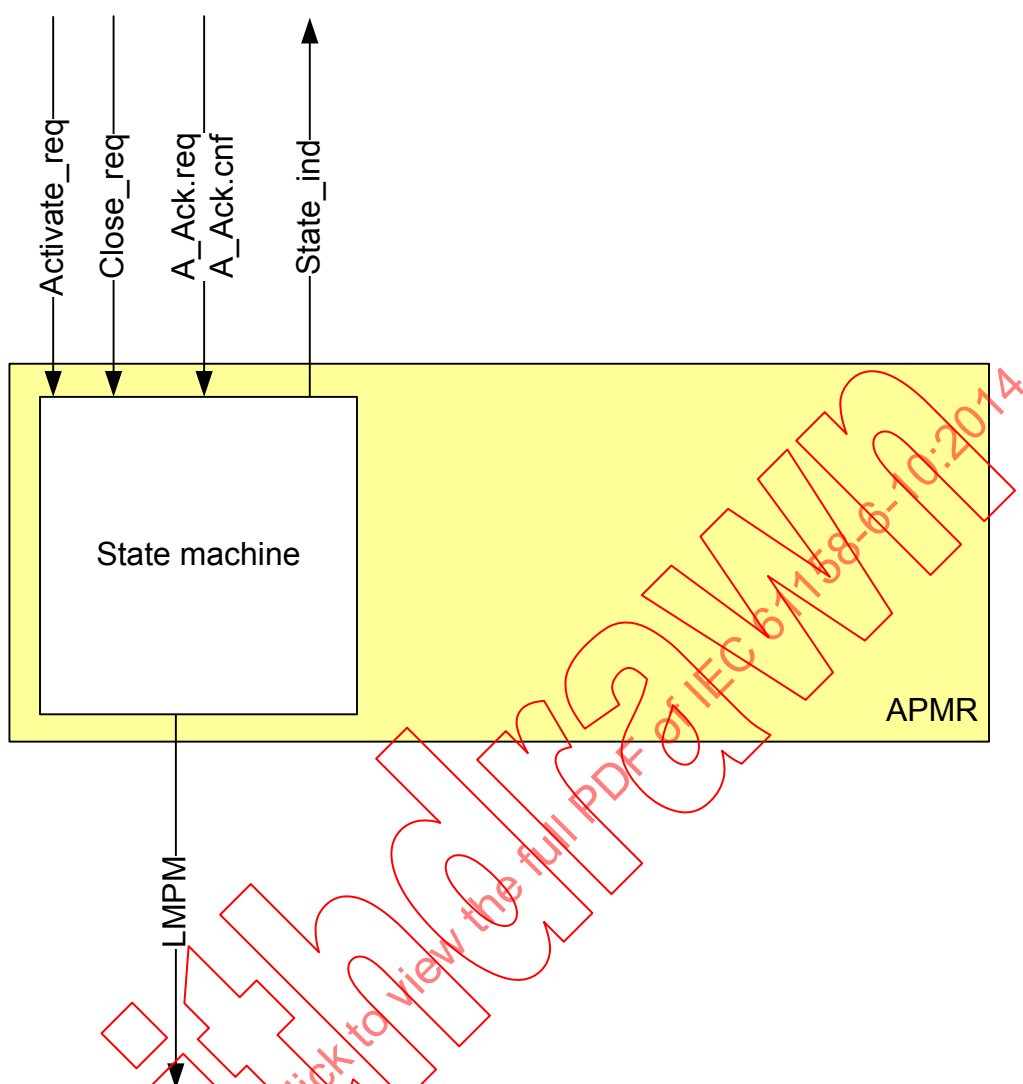


Figure 62 – Structure of the APMR

4.8.4.3.2 Primitive definitions

4.8.4.3.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by APMR are described in the service definition and shown in Table 209.

Table 208 – Remote primitives issued or received by APMR

Primitive	Source	Destination	Associated parameters	Functions
APMR_A_Data.ind	APMR	ALPMR	CREP, Data	Destination used if Data == AlarmNofication-PDU Data == unknown
APMR_A_Data.ind	APMR	ALPMI	CREP, Data	Destination used if Data == Alarm-Ack-PDU Data == unknown
LMPM_A_Data.cnf	APMR	LMPM	CREP, D_Port, Tstamp, LMPM_status	—

Primitive	Source	Destination	Associated parameters	Functions
LMPM_A_Data.ind	APMR	LMPM	CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_A_Data.req	APMR	LMPM	CREP, D_Port:=AUTO, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_N_Data.cnf	APMR	LMPM	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_N_Data.ind	APMR	LMPM	CREP, DA, SA, VLANPrio, VLANID, N_SDU	—
LMPM_N_Data.req	APMR	LMPM	CREP, D_Port:=AUTO, DA, SA, VLANPrio, VLANID, N_SDU	—

4.8.4.3.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CPM are described in the service definition and shown in Table 209.

Table 209 – Local primitives issued or received by APMR

Primitive	Source	Destination	Associated parameters	Functions
APMR_Activate_req	ALPMI	APMR	CREP, DA, SA, FrameID, VLANPrio, VLANID, Transport, DstIP, SrcIP	—
APMR_Close_req	ALPMI	APMR	CREP	—
APMR_Activate_cnf (-)	APMR	ALPMI	CREP, ERRCLS, ERRCODE	—
APMR_Activate_cnf (+)	APMR	ALPMI	CREP	—
APMR_Close_cnf (+)	APMR	ALPMI	CREP	—
APMR_Error_ind	APMR	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—

4.8.4.3.3 State transition diagram

The state transition diagram of the APMR is shown in Figure 63.

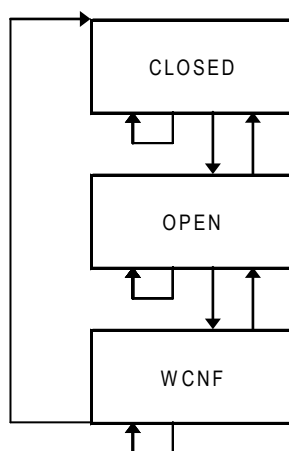


Figure 63 – State transition diagram of APMR

States of the APMR

CLOSED	The CLOSED state indicates that an initialization is needed. The Activate service sets the machine in the OPEN state.
OPEN	Waiting for LPM_A_Data or a UDP_A_Data service indication and convey a transport acknowledge.
WCNF	Wait for the confirmation of the transport acknowledge and issue a APMR_Data indication.

4.8.4.3.4 State machine description

This state machine, addressed by Source-Host, Source-ServiceAccessPoint, Destination-Host, DestinationAccessPoint and Priority, handles the A-Data service. Each priority (High and Low) is handled independent. This state machine is part of the APMS and APMR pair which is used as sender and responder for the transport acknowledge A-Data services. The A-Data service are used for the Alarm transport.

4.8.4.3.5 APMR state table

Table 210 contains the complete description of the APMR state machine. A LPM_A_Data.ind primitive will be accepted:

- If PDUType.Type field is DATA-RTA-PDU and the PDUType.Version field is one and the VarPartLen is greater than zero and the AddFlagsWindow is one.
- If PDUType.Type field is ERR-RTA-PDU and the PDUType.Version field is one and the VarPartLen is 4.

Table 210 – APMR state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	APMR_Activate.req () => AlarmInstance[SrcSAP, DstSAP, Prio] := DA, RTATimeoutFactor, RTARetries, ... // NOTE Information used for ALPMI, ALPMR, APMS and APMR Expected_Seq_Count:=0xffff Expected_Seq_CountO:=0xfffe Use CREP to identify the AlarmInstance APMR_Activate.cnf (+)	OPEN
2	CLOSED	APMR_Close.req () => APMR_Close.cnf (+)	CLOSED
3	CLOSED	A_Data_ind => ignore	CLOSED
4	CLOSED	A_Data_cnf => ignore	CLOSED
5	OPEN	APMR_Activate.req () => ERRCLS := APMR ERRCODE :=INVALID_STATE APMR_Activate.cnf (-)	OPEN
6	OPEN	APMR_Close.req () => APMR_Close.cnf (+)	CLOSED
7	OPEN	A_Data_ind /PDU.Type == DATA && PDU.AddFlags.Tack && PDU.SendSeqNum == Expected_Seq_Count => Store Data := PDU.RTA-SDU Expected_Seq_CountO :=Expected_Seq_Count Expected_Seq_Count := (Expected_Seq_Count+1) & 0x7fff PDU.Version :=1 PDU.Type := ACK PDU.AddFlags.TACK :=0 PDU.AddFlags.WindowSize :=1 PDU.SendSeqNum := Send_Seq_CountO (from the APMS) PDU.AckSeqNum := Expected_Seq_CountO //NOTE The PDU.SendSeqNum shall be set with the value from the corresponding APMS A_Data_req	WCNF
8	OPEN	A_Data_ind /PDU.Type == DATA && PDU.AddFlags.Tack && PDU.SendSeqNum == Expected_Seq_CountO => PDU.Version:=1 PDU.Type:= ACK PDU.AddFlags.TACK:=0 PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_CountO (from the APMS) PDU.AckSeqNum:=Expected_Seq_CountO A_Data_req	OPEN

#	Current State	Event /Condition =>Action	Next State
9	OPEN	A_Data_ind /PDU.Type == DATA && PDU.AddFlags.Tack && PDU.SendSeqNum != Expected_Seq_Count && PDU.SendSeqNum != Expected_Seq_CountO => ERRCLS := RTA_ERR_CLS_PROTOCOL ERRCODE := RTA_ERR_CODE_SEQ PDU.Version:=1 PDU.Type:= NAK PDU.AddFlags.TACK:=0 PDU.AddFlags.Window:=1 PDU.SendSeqNum:=Send_Seq_CountO (from the APMS) PDU.AckSeqNum:=Expected_Seq_CountO A_Data_req	OPEN
10	OPEN	A_Data_ind /PDU.Type == ERR => ERRCLS := PDU.PNIOStatus.ErrorCode1 ERRCODE := PDU.PNIOStatus.ErrorCode2 APMR_Error_ind ()	OPEN
11	OPEN	A_Data_ind /(PDU.Type == DATA && !PDU.AddFlags.Tack) => ignore	OPEN
12	OPEN	A_Data_ind /PDU.Type == ACK PDU.Type == NAK => ignore	OPEN
13	OPEN	A_Data_cnf => ignore	OPEN
14	WCNF	APMR_Activate.req () => ERRCLS := APMR ERRCODE :=INVALID_STATE APMR_Activate.cnf (-)	WCNF
15	WCNF	APMR_Close.req () => APMR_Close.cnf (+)	CLOSED
16	WCNF	A_Data_cnf => Retrieve Data := PDU.RTA-SDU APMR_A_Data.ind ()	OPEN
17	WCNF	A_Data_ind /PDU.Type != ERR => ignore	WCNF
18	WCNF	A_Data_ind /PDU.Type == ERR => APMR_Error_ind ()	WCNF

4.8.4.3.6 Functions, Macros, Timers and Variables

Table 211 contains the functions, macros, timers and variables used by the APMR and their arguments and their descriptions.

Table 211 – Functions, Macros, Timers and Variables used by the APMR

Name	Type	Meaning
A_Data_ind	Macro	if (Transport == RTC) LMPM_A_Data.ind(CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU) PDU := A_SDU if (Transport == UDP) LMPM_N_Data.ind(CREP, DA, SA, VLANPrio, VLANid, N_SDU) PDU := N_SDU
A_Data_cnf	Macro	if (Transport == RTC) LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) if (Transport == UDP) LMPM_N_Data.cnf (CREP, D_Port, Tstamp, LMPM_status)
A_Data_req	Macro	if (Transport == RTC) A_SDU := PDU LMPM_A_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, A_SDU) if (Transport == UDP) N_SDU := PDU LMPM_N_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, N_SDU)
PDU	Variable	Local variable representing the UDP-RTA-PDU or RTA-PDU depending on transport. Used to show the protocol specific setup and validation of RTA fields. The setup and validation of additional PDU-Parameters (e.g. IP-header) is not shown. The PDUType is shown in Table 202.
Expected_Seq_Count	Variable	This local variable contains the counter value which shall be used at the receipt of the next DATA PDU.
Expected_Seq_CountO	Variable	This local variable contains the previous counter value of the Seq_Count variable.

4.8.5 DLL Mapping Protocol Machines

There is no DLL Mapping Protocol Machine (DMPM) defined for this Protocol.

4.9 Fragmentation

4.9.1 General

The LengthOfPeriod is limited on the lower end by the maximum size of an Ethernet frame. The fragmentation reduces, with a peer to peer approach, the conveyed size of an Ethernet frame and allows to use a LengthOfPeriod down to 31,25 µs.

It uses the IEEE 802.1Q defined TagControlInformation.Priority to encode and decode up to 8 (at least one shall be supported) parallel fragmentation streams. Because of the bundling to the IEEE 802.1Q the granularity of the supported streams shall be one, two, four, or eight and the assignment to the TagControlInformation.Priority is shown in Table 212.

Table 212 – TagControlInformation.Priority vs. streams

Priority	One stream	Two streams	Four streams	Eight streams
7	Default All priorities are assigned to one stream	The upper four priorities are assigned to one stream	Two priorities are assigned to one stream	One priority is one stream
6			Two priorities are assigned to one stream	One priority is one stream
5				One priority is one stream
4				One priority is one stream
3		The lower four	Two priorities are	One priority is one stream

Priority	One stream	Two streams	Four streams	Eight streams
2		priorities are assigned to one stream	assigned to one stream	One priority is one stream
1			Two priorities are assigned to one stream	One priority is one stream
0				One priority is one stream

Even for the frames, generated by the fragmentation the lower frame size limit of the IEEE 802.3 applies. Thus limits the use of the principle of fragmentation according to Figure O.2, Formula (39), Formula (40), Formula (41) and Formula (42) as shown in Table 213.

The following Formula (39) shows the calculation for a single frame which is fragmented into a first and a last fragment.

NOTE The LowestFractionizeableFrameSize has two different values for frames with or without VLAN. Thus the fragmentation payload is defined and used as a simplification.

$$\text{LowestFractionizeableFrameSize} = \text{FrameDataFromFirstFragment} + \text{FrameDataFromLastFragment} + \text{ProtocolOverhead} \quad (39)$$

With VLAN

$$\text{LowestFractionizeableFrameSize} = 40 \text{ octets} + 38 \text{ octets} + 20 \text{ octets}$$

$$\text{LowestFractionizeableFrameSize} = 98 \text{ octets}$$

Without VLAN

$$\text{LowestFractionizeableFrameSize} = 48 \text{ octets} + 42 \text{ octets} + 16 \text{ octets}$$

$$\text{LowestFractionizeableFrameSize} = 106 \text{ octets}$$

where

LowestFractionizeableFrameSize	is the minimum length of frame to be able to be fractionized
FrameDataFromFirstFragment	is the amount of data of the original frame which is conveyed by the first fragment
FrameDataFromLastFragment	is the amount of data of the original frame which is conveyed by the last fragment
ProtocolOverhead	is the amount of data of the ethernet protocol frame

The following Formula (40) shows the calculation for the minimum leftover payload of a frame which is fragmented into a middle and a last fragment.

NOTE The LowestFractionizeableFrameSize has two different values for frames with or without VLAN. For simplification only the value with VLAN is used.

$$\text{LowestFractionizeableFrameSize} = \text{FrameDataFromMiddleFragment} + \text{FrameDataFromLastFragment} + \text{ProtocolOverhead} \quad (40)$$

With VLAN

$$\text{LowestFractionizeableFrameSize} = 40 \text{ octets} + 38 \text{ octets} + 26 \text{ octets}$$

$$\text{LowestFractionizeableFrameSize} = 104 \text{ octets}$$

Without VLAN

LowestFractionizeableFrameSize = 48 octets + 42 octets + 22 octets

LowestFractionizeableFrameSize = 112 octets

where

LowestFractionizeableFrameSize	is the minimum length of frame to be able to be fractionized
FrameDataFromMiddleFragment	is the amount of residual data of the original frame which is conveyed by a middle fragment
FrameDataFromLastFragment	is the amount of residual data of the original frame which is conveyed by the last fragment
ProtocolOverhead	is the amount of data of the ethernet protocol frame

Formula (41) shows a frame used for the first, middle or a last fragment with a VLAN tag.

$$\text{Lowest frame size(with VLAN)} = \text{DestinationMAC} + \text{SourceMAC} + \text{VLAN} + \text{FragmentationHeader} + \text{Payload} + \text{FCS} \quad (41)$$

$$\text{Payload(with VLAN)} = \text{Lowest frame size} - (\text{DestinationMAC} + \text{SourceMAC} + \text{VLAN} + \text{FragmentationHeader} + \text{FCS})$$

$$\text{Payload(with VLAN)} = 64 - (6 + 6 + 4 + 6 + 4)$$

$$\text{Payload(with VLAN)} = 38$$

according the rules for FragStatus.MoreFollows:=More fragments follows

$$\text{Payload(with VLAN)} = 40$$

according the rules for FragStatus.MoreFollows:=Last fragment

$$\text{Payload(with VLAN)} = 38$$

where

Lowest frame size	is the allowed minimum length of a frame (64 octets)
DestinationMAC	is the length of the MAC address (6 octets)
SourceMAC	is the length of the MAC address (6 octets)
VLAN	is the length of the VLAN tag (4 octets)
FragmentationHeader	is the length of the fragmentation header (6 octets)
Payload	is the length of the payload
FCS	is the length of the FCS (4 octets)

Formula (42) shows a frame used for the first, middle or a last fragment with a VLAN tag.

$$\text{Lowest frame size(without VLAN)} = \text{DestinationMAC} + \text{SourceMAC} + \text{FragmentationHeader} + \text{Payload} + \text{FCS} \quad (42)$$

$$\text{Payload(without VLAN)} = \text{Lowest frame size} - (\text{DestinationMAC} + \text{SourceMAC} + \text{FragmentationHeader} + \text{FCS})$$

$$\text{Payload(without VLAN)} = 64 - (6 + 6 + 6 + 4)$$

$$\text{Payload(without VLAN)} = 42$$

according the rules for FragStatus.MoreFollows:=More fragments follows

Payload(without VLAN) = 48

according the rules for FragStatus.MoreFollows:=Last fragment

Payload(without VLAN) = 42

where

Lowest frame size	is the allowed minimum length of a frame (64 octets)
DestinationMAC	is the length of the MAC address (6 octets)
SourceMAC	is the length of the MAC address (6 octets)
VLAN	is the length of the VLAN tag (4 octets)
FragmentationHeader	is the length of the fragmentation header (6 octets)
Payload	is the length of the payload
FCS	is the length of the FCS (4 octets)

Table 213 – Lower limit of fragments

Case	VLAN	Payload size of the fragments	Meaning
First + last	With	Payload first fragment := 40 octets Payload last fragment := 38 octets Protocol part := 20 octets	Smallest fragment able frame size := DA + SA + VLAN + Payload + FCS := 98 octets
	Without	Payload first fragment := 48 octets Payload last fragment := 42 octets Protocol part := 16 octets	Smallest fragment able frame size := DA + SA + Payload + FCS := 106 octets
Middle + last	With	Payload middle fragment := 40 octets Payload last fragment := 38 octets Protocol part := 26 octets	Smallest fragment able residual payload size := DA + SA + VLAN + FragHeader + Payload + FCS := 104 octets
	Without	Payload middle fragment := 48 octets Payload last fragment := 42 octets Protocol part := 22 octets	Smallest fragment able residual payload size := DA + SA + FragHeader + Payload + FCS := 112 octets

A fragmentation payload of 90 octets for all kind of frames (with and without VLAN) may be used to simplify the implementation of fragmentation. Thus the lowest frame size which could still be fragmented shall be 110 octets with VLAN and 106 octets without VLAN and the lowest residual fragmentation payload which could still be fragmented shall be 116 octets with VLAN and 112 octets without VLAN.

4.9.2 FRAG syntax description

4.9.2.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.9.2.2 FRAG APDU abstract syntax

Table 214 defines the abstract syntax of the Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 214 shall represent the content of the DLSDU in Table 5.

Table 214 – FRAG APDU syntax

APDU name	APDU structure
FRAG-PDU	FRAG-DATA-PDU

Table 215 defines structures for substitutions of elements of the APDU structures shown in Table 214.

Table 215 – FRAG substitutions

Substitution name	Structure
FRAG-DATA-PDU	FragDataLength, FragStatus, FragData

4.9.3 FRAG transfer syntax

4.9.3.1 Coding section related to FRAG-PDUs specific fields

4.9.3.1.1 Coding of the field FragDataLength

The field FragDataLength is coded as an Unsigned8 and contains the number of 8 octet portions of the fragment data. For the last fragment (FragStatus.MoreFollows := 0) up to 7 “leftover” octets may be added.

FragDataLength shall be calculated according to Formula (43).

$$\text{FragDataLength} = (\text{Length of FragData} \text{ DIV } 8) \quad (43)$$

where

FragDataLength	is the length of the field FragData in 8 octet portions
Length of FragData	is the length of the field FragData in octets
8	is the defined value of an increment of FragData

If the fragmentation of a frame into multiples of 8 octet portions has any leftover, then the size of the last fragment containing the FragData shall be the rest, containing the leftover octets.

The receiver shall accept a fragmented frame with this kind of FragData if FragStatus.MoreFollows := 0.

This field shall be coded as Unsigned8 with the values according to Table 216.

Table 216 – FragDataLength

Value (hexadecimal)	Meaning	Usage
0x00, ..., 0x04	—	Reserved
0x05 or 0x06 ^a	Minimum If the FragStatus.MoreFollows := 1, then the length of FragData is 0x30 octets If the FragStatus.MoreFollows := 0, then the length of FragData is 0x30 to 0x37 octets (up to 7 leftover octets)	Mandatory
0x08, ..., 0xBE	...	Mandatory
0xBF ^b	Maximum If the FragStatus.MoreFollows := 1, then the length of FragData is 0x5CF octets If the FragStatus.MoreFollows := 0, then the length of FragData is 0x5F8 to 0x5FF octets	Mandatory
0xC0, ..., 0xFF	Maximum length of FragData is 0x7FF	Optional
^a The minimum FragDataLength which shall be supported by the DEFRAG is 0x05 with VLAN and 0x06 without VLAN. ^b The maximum FragDataLength which shall be supported by the DEFRAG is defined by the use of at least two fragments for a maximum IEEE 802.3 frame. The support of one fragment fragmentation is not intended.		

4.9.3.1.2 Coding of the field FragStatus

The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0 – 5: FragStatus.FragmentNumber

This field contains the fragment number and shall be coded according to Table 217.

Table 217 – FragStatus.FragmentNumber

Value (hexadecimal)	Meaning
0x00	First fragment of a frame
0x01	Second fragment of a frame
...	...
0x3F	64 th fragment of a frame

Bit 6: FragStatus.Reserved

This field shall be coded according to Table 218.

Table 218 – FragStatus.Reserved

Value (hexadecimal)	Meaning
0x00	Reserved

Bit 7: FragStatus.MoreFollows

This field shall be coded according to Table 219.

Table 219 – FragStatus.MoreFollows

Value (hexadecimal)	Meaning
0x00	Last fragment
0x01	More fragments follows

4.9.3.1.3 Coding of the field FragData

This field shall be coded as data type OctetString and shall contain the transmitted portion of data of the fragmented frame.

4.9.4 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.9.5 Application Relationship Protocol Machines

There is no Application Relationship Protocol Machine (ARPM) defined for this Protocol.

4.9.6 DLL Mapping Protocol Machines**4.9.6.1 Fragmentation****4.9.6.1.1 Fragmentation dynamic****4.9.6.1.1.1 Primitive definitions****4.9.6.1.1.1.1 Primitives exchanged between remote machines**

The service primitives including their associated parameters issued or received by FRAG_D are described in the service definition and shown in Table 220.

Table 220 – Remote primitives issued or received by FRAG_D

Primitive	Source	Destination	Associated parameters	Functions
QueueHandler.req	FRAG_D	QueueHandler	Queue, Frame	—

4.9.6.1.1.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by FRAG_D are described in the service definition and shown in Table 221.

Table 221 – Local primitives issued or received by FRAG_D

Primitive	Source	Destination	Associated parameters	Functions
SCHEDULER_StartOfPeriod	SCHEDULER	FRAG	—	This local function is issued if a new communication cycle starts.

4.9.6.1.1.2 State transition diagram

The state transition diagram of the FRAG_D is shown in Figure 64.

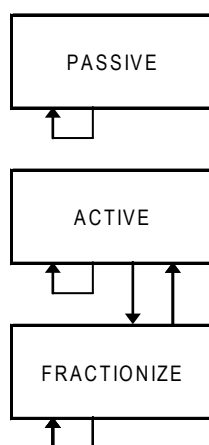


Figure 64 – State transition diagram of FRAG_D

States of the FRAG_D

PASSIVE

The FRAG unit is not used. Frames passes unaffected.

ACTIVE

The FRAG unit for the defined stream is active and waits for frames to fractionize them if needed.

FRACTIONIZE

A frame is fragmented and conveyed. High priority frames which do not need fragmentation may overtake the fragments. After the last fragment is sent, the state ACTIVE is reached. If residential time is available the fragmentation starts immediately with the next frame.

4.9.6.1.1.3 State machine description

The FRAG is responsible for fragmentation of frames according to the residual GREEN period. It gets the frames from the QueueHandler, breaks them into fragments, if necessary and allowed for the protocol, and stores the fragments in the lower queues of the QueueHandler.

In the inactive state FRAG shall be bypassed by the QueueHandler.

The fragmentation may be done by two different strategies. The first one is the dynamic fragmentation, which uses the residential GREEN phase and the second one is the static fragmentation, which uses a predefined fragment size.

4.9.6.1.1.4 FRAG_D state table (dynamic fragmentation)

Table 222 contains the FRAG_D state table.

Table 222 – FRAG_D state table (dynamic)

#	Current State	Event /Condition =>Action	Next State
1	PASSIVE	=> ignore	PASSIVE
2	ACTIVE	QueuesNotEmpty () /Period == RED => ignore	ACTIVE

#	Current State	Event /Condition =>Action	Next State
3	ACTIVE	QueuesNotEmpty () /Period == GREEN => GetFrameFromQueue () StartFragStream ()	FRACTIONIZE
4	FRACTIONIZE	QueuesNotEmpty () => ignore	FRACTIONIZE
5	FRACTIONIZE	StartFragStream () /Period == GREEN && FrameFitsIntoResidualPeriod == TRUE => QueueHandler.req () //NOTE Send frame	ACTIVE
6	FRACTIONIZE	StartFragStream () /Period == GREEN && FrameFitsIntoResidualPeriod == FALSE && FrameSizeAllowsFragmentation == TRUE && ResidualPeriodShorterThanMin == TRUE => ignore //NOTE Wait until the next GREEN period starts	FRACTIONIZE
7	FRACTIONIZE	SCHEDULER_StartOfPeriod () /Period == GREEN && ResidualPeriodShorterThanMin == TRUE => ignore	FRACTIONIZE
8	FRACTIONIZE	StartFragStream () /Period == GREEN && FrameFitsIntoResidualPeriod == FALSE && FrameSizeAllowsFragmentation == TRUE && ResidualPeriodShorterThanMin == FALSE && LastFragment == FALSE => CalculateFragSize () CreateFrag () QueueHandler.req () //NOTE Send frame fragment	FRACTIONIZE
9	FRACTIONIZE	SCHEDULER_StartOfPeriod () /Period == GREEN && ResidualPeriodShorterThanMin == FALSE && LastFragment == TRUE => CalculateFragSize () CreateFrag () QueueHandler.req () //NOTE Send frame fragment	ACTIVE

#	Current State	Event /Condition =>Action	Next State
10	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameFitsIntoResidualPeriod == TRUE => GetFrameFromQueue () QueueHandler.req ()	FRACTIONIZE
11	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameFitsIntoResidualPeriod == FALSE => ignore	FRACTIONIZE

4.9.6.1.1.5 Functions, Macros, Timers and Variables

Table 223 contains the functions, macros, timers and variables used by the FRAG_D and their arguments and their descriptions.

Table 223 – Functions, Macros, Timers and Variables used by the FRAG_D (dynamic)

Name	Type	Function/Meaning
CalculateFragSize	Function	This local function calculates the fragment size due to the residual GREEN period and the fragmentation rules. If dynamic fragmentation is not supported, use a default length given by the engineering according the minimal GREEN period of all phases.
CheckForFrameWithHigherPriority	Function	This local function tests whether any frame with a higher priority than the fragmented one is stored in the queues.
CreateFrag	Function	This local function creates a fragment according the coding and the fragmentation rules. Either first, next or last. Keep in mind, that the minimum and the maximum frame size of a IEEE 802.3 frame are still valid, even for fragments.
FrameFitsIntoResidualPeriod	Function	This local function checks whether the residual GREEN period is longer than the frame needs for conveyance
FrameSizeAllowsFragmentation	Function	This local function checks whether the frame is big enough for fragmentation
GetFrameFromQueue	Function	This local function fetches the next frame according the amount of FRAG streams and priorities.
QueuesNotEmpty	Function	This local function is issued if any entry is stored in the queues.
ResidualPeriodShorterThanMin	Function	This local function checks whether the residual GREEN period is shorter than an minimum frame needs for conveyance
StartFragStream	Function	This local function triggers the fragment creation of FRAG.
LastFragment	Macro	This local macro contains information whether the actual fragment is the last fragment. With the conveyance of the last fragment ends the FRAG stream
Period	Macro	This local macro contains the information what period (RED or GREEN) is active.

4.9.6.1.2 Fragmentation static

4.9.6.1.2.1 Primitive definitions

4.9.6.1.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by FRAG_S are described in the service definition and shown in Table 224.

Table 224 – Remote primitives issued or received by FRAG_S

Primitive	Source	Destination	Associated parameters	Functions
QueueHandler.req	FRAG_S	QueueHandler	Queue, Frame	—

4.9.6.1.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by FRAG_S are described in the service definition and shown in Table 225.

Table 225 – Local primitives issued or received by FRAG_S

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.9.6.1.2.2 State transition diagram

The state transition diagram of the FRAG_S is shown in Figure 65.

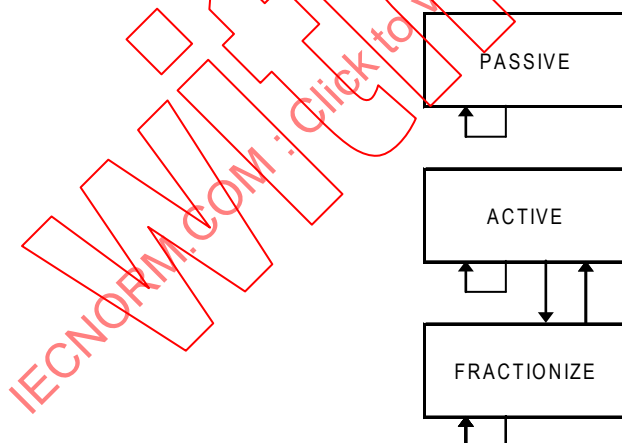


Figure 65 – State transition diagram of FRAG_S

States of the FRAG_S

PASSIVE

The FRAG unit is not used. Frames pass unaffected.

ACTIVE

The FRAG unit for the defined stream is active and waits for frames to fractionize them if needed.

FRACTIONIZE

A frame is fragmented and conveyed. High priority frames which do not need fragmentation may overtake the fragments. After the last fragment is sent, the state ACTIVE is reached. If residual time is available the fragmentation starts immediately with the next frame.

4.9.6.1.2.3 State machine description

The FRAG is responsible for fragmentation of frames according to the residual GREEN period. It gets the frames from the QueueHandler, breaks them into fragments, if necessary and allowed for the protocol, and stores the fragments in the lower queues of the QueueHandler.

In the inactive state FRAG shall be bypassed by the QueueHandler.

The fragmentation may be done by two different strategies. The first one is the dynamic fragmentation, which uses the residential GREEN phase and the second one is the static fragmentation, which uses a predefined fragment size.

4.9.6.1.2.4 FRAG_S state table (static fragmentation)

Table 226 contains the FRAG_S state table.

Table 226 – FRAG_S state table (static)

#	Current State	Event /Condition =>Action	Next State
1	PASSIVE	=> ignore	PASSIVE
2	ACTIVE	QueuesNotEmpty () /Period == RED => ignore	ACTIVE
3	ACTIVE	QueuesNotEmpty () /Period == GREEN && FrameSizeAllowsFragmentation == FALSE => GetFrameFromQueue () // Adapt the assigned queues if more than one Frag stream exists QueueHandler.req () //NOTE Send frame	ACTIVE
4	ACTIVE	QueuesNotEmpty () /Period == GREEN && FrameSizeAllowsFragmentation == TRUE => GetFrameFromQueue () // Adapt the assigned queues if more than one Frag stream exists StartFragStream ()	FRACTIONIZE
5	FRACTIONIZE	QueuesNotEmpty () => ignore	FRACTIONIZE
6	FRACTIONIZE	StartFragStream () => CalculateFragSize () CreateAllFrag () For all fragment frames QueueHandler.req () //NOTE Send fragment frame	FRACTIONIZE
7	FRACTIONIZE	LastFragmentTransmitted () =>	ACTIVE

#	Current State	Event /Condition =>Action	Next State
8	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameSizeAllowsFragmentation == FALSE => GetFrameFromQueue () // Adapt the assigned queues if more than one Frag stream exists QueueHandler.req () //NOTE Send frame	FRACTIONIZE
9	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameSizeAllowsFragmentation == TRUE => ignore	FRACTIONIZE

4.9.6.1.2.5 Functions, Macros, Timers and Variables

Table 227 contains the functions, macros, timers and variables used by the FRAG_S and their arguments and their descriptions.

Table 227 – Functions, Macros, Timers and Variables used by the FRAG_S (static)

Name	Type	Function/Meaning
CalculateFragSize	Function	This local function calculates the fragment size due to the given expected fragment size and the fragmentation rules. The default length is given by the engineering according the minimal GREEN period of all phases.
CheckForFrameWithHigherPriority	Function	This local function tests whether any frame with a higher priority than the fragmented one is stored in the queues.
CreateAllFrag	Function	This local function creates all fragments according the coding and the fragmentation rules. Either first, next or last. Keep in mind, that the minimum and the maximum frame size of a IEEE 802.3 frame are still valid, even for fragments.
FrameSizeAllowsFragmentation	Function	This local function checks whether the frame is big enough for fragmentation
GetFrameFromQueue	Function	This local function fetches the next frame according the amount of FRAG streams and priorities.
QueuesNotEmpty	Function	This local function is issued if any entry is stored in the queues.
StartFragStream	Function	This local function triggers the fragment creation of FRAG.
LastFragmentTransmitted	Function	This local function is issued if the last fragment is conveyed.
Period	Macro	This local macro contains the information what period (RED or GREEN) is active.

4.9.6.2 Defragmentation

4.9.6.2.1 Primitive definitions

4.9.6.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DEFRAG are described in the service definition and shown in Table 228.

Table 228 – Remote primitives issued or received by DEFRAG

Primitive	Source	Destination	Associated parameters	Functions
DMUX_DEFRAG.ind	DEMUX	DEFRAG	Port, Tstamp, DA, SA, VLAN, N_SDU	—
DMUX_MAC_RELAY.ind	DEFRAG	MAC_RELAY	Port, Tstamp, DA, SA, N_SDU	—

4.9.6.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DEFRAG are described in the service definition and shown in Table 229.

Table 229 – Local primitives issued or received by DEFRAG

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.9.6.2.2 State transition diagram

The state transition diagram of the DEFRAG is shown in Figure 66.

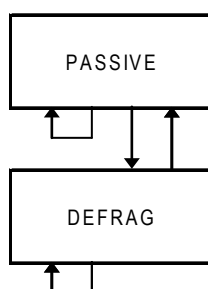


Figure 66 – State transition diagram of DEFRAG

States of the DEFRAG

PASSIVE

Waiting for the first fragment of a stream. For each supported steam exists a DEFRAG unit.

DEFRAG

A stream was started by the first fragment. After receiving the last fragment of the current stream, the machine reaches PASSIVE again. If the amount of data received during DEFRAG exceeds the maximum frame size according to the IEEE 802.3, the defragmentation is

aborted and the state PASSIVE entered.

4.9.6.2.3 State machine description

A DEFRAG state machine shall exist for every supported stream. At least one stream shall be supported, if fragmentation is supported. In this case the DEFRAG is waiting for a FRAGMENT after the link is active.

4.9.6.2.4 DEFRAG state table

Table 230 contains the DEFRAG state table which exists for each port.

Table 230 – DEFRAG state table

#	Current State	Event /Condition =>Action	Next State
1	PASSIVE	DMUX_DEFRAG.ind () /DefragGuard () == FirstFragment => Store fragment	DEFRAG
2	PASSIVE	DMUX_DEFRAG.ind () /DefragGuard () == Invalid => ignore	PASSIVE
3	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == NextFragment && FrameLimitReached => Delete stored fragments	PASSIVE
4	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == NextFragment && ! FrameLimitReached => Store fragment	DEFRAG
5	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == LastFragment && FrameLimitReached => Delete stored fragments	PASSIVE
6	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == LastFragment && ! FrameLimitReached => Create frame Delete stored fragments DMUX_MAC_RELAY.ind ()	PASSIVE
7	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == FirstFragment => Delete stored fragments Store fragment	DEFRAG
8	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == Invalid => Delete stored fragments	PASSIVE

4.9.6.2.5 Functions, Macros, Timers and Variables

Table 231 contains the functions used by the DEFRAG, their arguments and their descriptions.

Table 231 – Functions, Macros, Timers and Variables used by the DEFRAG

Name	Type	Function/Meaning
DefragGuard	Function	This local function checks the fragments against truth tables.
Store fragment	Macro	This local macro store the fragment for further defragmentation
Delete stored fragment	Macro	This local macro deletes the previous stored fragments
Create frame	Macro	This local macro creates a frame out of the stored fragments
FrameLimitReached	Macro	This local macro checks whether the IEEE 802.3 frame limits are exceeded.

4.9.6.2.6 DefragGuard

The DefragGuard checks the received frame against Table 232, Table 233 and Table 234. The used table depends on the local supported functionality.

The defragmentation shall accept a frame independent from the value of FrameID.FragSequence in combination with FragStatus.FragmentNumber:=0 as start of a new defragmentation sequence.

Table 232 – Truth table for the DefragGuard – first fragment

Input				Output
FragmentationFrameID. FragSequence	FragStatus. FragmentNumber	FragStatus. MoreFollows	FragDataLength	Local
Invalid	—	—	—	Invalid
Valid	Invalid	—	—	Invalid
Valid	Valid	FALSE ^a	—	Invalid
Valid	Valid	TRUE	Invalid	Invalid
Valid	Valid	TRUE	Valid	Valid

^a The MoreFollows for the first fragment shall never be set to FALSE by the sender.

Table 233 – Truth table for the DefragGuard – next fragment

Input				Output
FragmentationFrameID. FragSequence	FragStatus. FragmentNumber	FragStatus. MoreFollows	FragDataLength	Local
Invalid	—	—	—	Invalid
Valid ^a	Invalid	—	—	Invalid
Valid	Valid ^b	FALSE	—	Invalid
Valid	Valid ^b	TRUE	Invalid	Invalid
Valid	Valid	TRUE	Valid	Valid

^a Identical to the store one from the first fragment.
^b The stored one from the last received fragment of this FragSequence (FragmentNumber =+ 1).

Table 234 – Truth table for the DefragGuard – last fragment

Input				Output
FragmentationFrameID. FragSequence	FragStatus. FragmentNumber	FragStatus. MoreFollows	FragDataLength	Local
Invalid	—	—	—	Invalid
Valid	Invalid	—	—	Invalid
Valid	Valid	TRUE	—	Invalid
Valid	Valid	FALSE ^a	Invalid	Invalid
Valid	Valid	FALSE ^a	Valid	Valid

^a The MoreFollows for the last fragment shall be set to FALSE by the sender.

4.10 Remote procedure call

4.10.1 RPC syntax description

4.10.1.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.10.1.2 RPC APDU abstract syntax

Table 235 shows the utilization of the Publication C706 of The Open Group and defines the abstract syntax of the Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 235 shall represent the content of the DLSDU in Table 5.

Table 235 – RPC APDU syntax

APDU name	APDU structure
CL-RPC-PDU	IPHeader, UDPHeader, RPCHeader, NDRData
NDRData	NDRDataRequest ^ NDRDataResponse ^ NDREPMAPLookupReq ^ NDREPMAPLookupRes ^ NDREPMAPLookupFreeReq ^ NDREPMAPLookupFreeRes ^ NDRAck ^ NDRQuAck ^ NDRQuit ^ NDRFack ^ NDRFault ^ NDRNoCall ^ NDRWorking ^ NDRPing ^ NDRReject

Table 236 defines structures for substitutions of elements of the APDU structures shown in Table 235.

Table 236 – RPC substitutions

Substitution name	Structure
RPCHeader	RPCVersion (4), RPCPacketType, RPCFlags, RPCFlags2, RPCDRep, RPCSerialHigh, RPCObjectUUID ^a , RPCInterfaceUUID ^b , RPCActivityUUID, RPCServerBootTime, RPCInterfaceVersion, RPCSequenceNmb, RPCOperationNmb ^c , RPCInterfaceHint, RPCActivityHint, RPCLengthOfBody, RPCFragmentNmb, RPCAuthenticationProtocol, RPCSerialLow
NDRDataRequest	ArgsMaximum, ArgsLength, MaximumCount, Offset(0), ActualCount, PROFINETIOServiceReqPDU
NDRDataResponse	PNIOStatus, ArgsLength, MaximumCount, Offset(0), ActualCount, [PROFINETIOServiceResPDU]
NDRFault	RPCNCAFaultStatus
NDRAck	NULL
NDRQuAck	NULL ^ (RPCCancelVersion(0), RPCCancelID, RPCServerIsAccepting)

Substitution name	Structure
NDRQuit	NULL ^ (RPCCancelVersion(0), RPCCancelID)
NDRFack	NULL ^ (RPCVersionFack(0), RPCPad1, RPCWindowSize, RPCMaxTsdu, RPCMaxFragSize, RPCSerialNumber, RPCSelAckLen, RPCArrayOfSelAck*)
NDRNoCall	NULL ^ NDRFack
NDRWorking	NULL
NDRPing	NULL
NDRReject	RPCNCARrejectStatus
NDREMapLookupReq	RPCInquiryType, RPCObjectReference(1), RPCObjectUUID, RPCInterfaceReference(2), RPCInterfaceUUID, RPCInterfaceVersionMajor, RPCInterfaceVersionMinor, RPCVersionOption(1), RPCEntryHandleAttribute(0), RPCEntryHandleUUID, RPCMaxEntries
NDREMapLookupRes	RPCEntryHandleAttribute(0), RPCEntryHandleUUID, RPCNumberOfEntries, RPCMaxEntries, RPCEntriesOffset, RPCEntriesCount, [(RPCObjectUUID, RPCTowerReference, RPCAnnotationOffset, RPCAnnotationLength, RPCAnnotation, [RPCGap*], RPCTowerLength, RPCTowerOctetStringLength, [RPCTowerOctetString*], [RPCGap*])*], RPCEMapStatus
RPCTowerOctetString	RPCFloorCount, [RPCFloor*]
RPCFloor	RPCLHSByteCount, [RPCProtocolIdentifierString*], RPCRHSByteCount, [RPCRelatedData*]
RPCProtocolIdentifier-String	(RPCID, RPCInterfaceUUID ^a , RPCInterfaceVersionMajor ^a) ^ (RPCID, RPCDataRepresentationUUID ^a , RPCInterfaceVersionMajor) ^ RPCProtocolIdentifier ^ RPCServerUDPPort ^ RPCHostAddress ^a Shall be coded in little endian format
RPCRelatedData	RPCInterfaceVersionMinor ^a ^ RPCPortNumber ^ RPCIPAddress ^a Shall be coded in little endian format
NDREMapLookupFreeReq	RPCEntryHandleAttribute, RPCEntryHandleUUID
NDREMapLookupFreeRes	RPCEntryHandleAttribute, RPCEntryHandleUUID, RPCEMapStatus
RPCAnnotation	DeviceType, Blank, OrderID, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision, EndTerm
^a To identify IO device, IO controller, IO supervisor. ^b To identify PNIO interface type. ^c To identify the service type e.g. Connect, Release, Read, Write, Control.	

4.10.2 RPC Transfer syntax

4.10.2.1 Coding section related to CL-RPC-PDU

4.10.2.1.1 Coding of the field RPCVersion

This field shall be coded as data type Unsigned8 with values according to Table 237.

Table 237 – RPCVersion

Value (hexadecimal)	Meaning
0x04	Used RPC version
Other	Reserved

4.10.2.1.2 Coding of the field RPCPacketType

This field shall be coded as data type Unsigned8 with values according to Table 238.

Table 238 – RPCPacketType

Value (hexadecimal)	Meaning
0x00	Request
0x01	Ping
0x02	Response
0x03	Fault
0x04	Working
0x05	No call, response to ping
0x06	Reject
0x07	Acknowledge
0x08	Connectionless cancel
0x09	Fragment acknowledge (FACK-PDU)
0x0A	Cancel acknowledge
0x0B – 0xFF	Reserved

The decoder shall only check the 5 least significant bits.

4.10.2.1.3 Coding of the field RPCFlags

This field shall be coded as data type according to 3.4.2.3 with values according to Table 239.

Table 239 – RPCFlags

Bit	Meaning if set to 1
Bit 0	Implementation specific, shall be set to zero
Bit 1	Last fragment
Bit 2	Fragment
Bit 3	No fragment acknowledge requested
Bit 4	Maybe
Bit 5	Idempotent
Bit 6	Broadcast
Bit 7	Implementation specific, shall be set to zero

4.10.2.1.4 Coding of the field RPCFlags2

This field shall be coded as data type according to 3.4.2.3 with values according to Table 240.

Table 240 – RPCFlags2

Bit	Meaning if set to 1
Bit 0	Implementation specific, shall be set to 0
Bit 1	Cancel was pending at call end
Bit 2	Reserved
Bit 3	Reserved
Bit 4	Reserved
Bit 5	Reserved

Bit	Meaning if set to 1
Bit 6	Reserved
Bit 7	Reserved

4.10.2.1.5 Coding of the field RPCDRep

This field shall be coded as data type OctetString[3].

RPCDRep Octet 1

The coding of the first octet shall be according to 3.4.2.3 and the individual bits shall have the meaning defined in Table 241.

Table 241 – RPCDRep.Character- and IntegerEncoding

Bit	Field Name	Value	Meaning
0 – 3	RPCDRep.CharacterEncoding ^a	0	ASCII
		1	EBCDIC
4 – 7	RPCDRep.IntegerEncoding ^b	0	Big endian ^c
		1	Little endian ^c

^a Not used within Type 10.

^b As an exception to 3.4.2.5 the RPC encoding rules are applied to the RPCHeader, NDR substitutions and NDREMap substitutions within Type 10. It only uses Unsigned8, Unsigned16, or Unsigned32 there. The user data of the Type 10 service definitions are not influenced because everything is an opaque OctetString there.

^c See The Open Group — Publication C706.

RPCDRep Octet 2

The values of the second octet shall be encoded according to Table 242.

Table 242 – RPCDRep Octet 2 – Floating Point Representation

Value (hexadecimal)	Meaning
0x00	IEEE
0x01	VAX
0x02	CRAY
0x03	IBM
0x04 – 0xFF	Reserved

RPCDRep Octet 3

The value of the third octet shall be zero.

4.10.2.1.6 Coding of the field RPCSerialHigh

This field shall be coded as data type Unsigned8. The value contains the high octet of the fragment number of the call.

4.10.2.1.7 Coding of the field RPCSerialLow

This field shall be coded as data type Unsigned8. The value contains the low octet of the fragment number of the call.

NOTE The value of the field RPCSerial is incremented which each transmission even with a transmission repetition in distinction to the field FragmentNmb.

4.10.2.1.8 Coding of the field RPCObjectUUID

This field shall be coded as data type structure containing the following elements:

- Data1 as Unsigned32
- Data2 as Unsigned16
- Data3 as Unsigned16
- Data4 as array of eight Unsigned8 (Octet 1 to Octet 8)

The octet ordering shall be according to the value of the field RPCDRep (little endian or big endian).

The ordering of octets of Data4 is shown in Table 243.

Table 243 – RPCObjectUUID.Data4

Octet	Meaning
1	Values see Table 244
2	
3	Instance High
4	Instance Low
5	DeviceID High
6	DeviceID Low
7	VendorID High
8	VendorID Low

Defined values for PNIO items are shown in Table 244 and Table 245. These values shall be used in conjunction with the values from Table 246.

Table 244 – RPCObjectUUID for PNIO

Value	Description
UUID_IO_ObjectInstance_XYZ DEA00000-6C97-11D1-8271-{xxxxyyyyzzzz}	Identifies an object instance within a physical device in case there are more than one instance, where
xxxx	Represents the instance or node number See Table 245
yyyy	Identify the Device ID as a vendor specific number for the device class
zzzz	Represents the Vendor ID as a central administrative number assigned by the responsible user organisation

Table 245 – RPCObjectUUID for PNIO with multiple interfaces

Value	Description
xxxx	Identifies an object instance within a physical device in case there are more than one instance, where
Bit 0 – Bit 11	Represents the instance or node number
Bit 12 – Bit 15	Identify the interface 0x0: Interface 0 (Default) ... 0xF: Interface 15 See “I” within Formula (45)

4.10.2.1.9 Coding of the field RPCInterfaceUUID

This field shall be coded as data type structure containing the following elements:

- Data1 as Unsigned32
- Data2 as Unsigned16
- Data3 as Unsigned16
- Data4 as array of eight Unsigned8

The octet ordering shall be according to the value of the field RPCDRep (little endian or big endian).

Defined values for PNIO items are shown in Table 246.

Table 246 – RPCInterfaceUUID for PNIO

Value	Description
UUID_IO_DeviceInterface DEA00001-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO device uniquely. RPCInterfaceVersion shall be: Major version: 1 Minor version: 0
UUID_IO_ControllerInterface DEA00002-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO controller uniquely. RPCInterfaceVersion shall be: Major version: 1 Minor version: 0
UUID_IO_SupervisorInterface DEA00003-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO supervisor uniquely. RPCInterfaceVersion shall be: Major version: 1 Minor version: 0
UUID_IO_ParameterServerInterface DEA00004-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO parameter server uniquely. RPCInterfaceVersion shall be: Major version: 1 Minor version: 0

Defined values for the RPC end point mapper are shown in Table 247.

Table 247 – RPCInterfaceUUID for the RPC end point mapper

Value	Description
UUID_EPMap_Interface E1AF8308-5D1F-11C9-91A4-08002B14A0FA	Identifies the interface of the endpoint mapper.
UUID_EPMap_Object 00000000-0000-0000-0000-000000000000	The RPCInterfaceVersion shall be: Major version: 3 Minor version: 0

4.10.2.1.10 Coding of the field RPCActivityUUID

This field shall be coded as data type structure containing the following elements:

- Data1 as Unsigned32
- Data2 as Unsigned16
- Data3 as Unsigned16
- Data4 as array of eight Unsigned8

NOTE The response mirrors the content of the field of the request.

The endianness is according to the value in the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The IOC and the IOD generates an RPCActivityUUID for each AR and use them as long as the AR exist.

4.10.2.1.11 Coding of the field RPCServerBootTime

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

NOTE Idempotent functions do not need to maintain this value. Only the endpointmapper is allowed to answer with zero.

4.10.2.1.12 Coding of the field RPCInterfaceVersion

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 15: RPCInterfaceVersion.Major

This field shall be set according to Table 248.

Table 248 – RPCInterfaceVersion.Major

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001 – 0xFFFF	Valid version information

Bit 16 – 31: RPCInterfaceVersion.Minor

This field shall be set according to Table 249.

Table 249 – RPCInterfaceVersion.Minor

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Valid version information

The endianness is according to the value in the field “RPCDRep.IntegerEncoding” (little endian or big endian).

4.10.2.1.13 Coding of the field RPCSequenceNmb

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

4.10.2.1.14 Coding of the field RPCOperationNmb

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The RPCOperationNmb identifies the PNIO service supported by the PNIO interfaces

- IO device,
- IO controller and
- IO supervisor.

The values for PNIO services are defined in Table 250.

Table 250 – RPCOperationNmb (IO device, controller and supervisor)

Value (decimal)	Service	Usage
0	Connect	—
1	Release	—
2	Read	Only valid with ARUUUID<>0
3	Write	Only valid with ARUUUID<>0
4	Control	—
5	Read Implicit	Only valid with ARUUUID=0
6 – 65 535	Reserved	—

The values for endpoint mapper services according to Table 251 shall be used.

Table 251 – RPCOperationNmb for endpoint mapper

Value (decimal)	Usage	Service
0	Optional	Insert
1	Optional	Delete
2	Mandatory	Lookup
3	Optional	Map
4	Mandatory	LookupHandleFree
5	Optional	InqObject
6	Optional	MgmtDelete

Value (decimal)	Usage	Service
7 – 65 535	Reserved	—

4.10.2.1.15 Coding of the field **RPCInterfaceHint**

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value should be set to no hint (0xFFFF) for this version.

The client shall start with “no hint” for the first call and should use the server response for the following calls to allow a server optimization.

4.10.2.1.16 Coding of the field **RPCActivityHint**

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value should be set to no hint (0xFFFF) for this version.

The client shall start with “no hint” for the first call and should use the server response for the following calls to allow a server optimization.

4.10.2.1.17 Coding of the field **RPCLengthOfBody**

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to the number of octets of NDRData of the current frame.

The conveyance of NDRData may require more than one frame. In this case RPCLengthOfBody contains only the number of octets for the current frame.

4.10.2.1.18 Coding of the field **RPCFragmentNmb**

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to the number of the current fragment.

4.10.2.1.19 Coding of the field **RPCAuthenticationProtocol**

This field shall be coded as data type Unsigned8.

The value shall be set to zero for no authentication.

4.10.2.1.20 Coding of the field **RPCCancelVersion**

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to zero.

4.10.2.1.21 Coding of the field RPCCancelID

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.1.22 Coding of the field RPCServerIsAccepting

This field shall be coded as data type Unsigned8.

4.10.2.1.23 Coding of the field RPCVersionFack

This field shall be coded as data type Unsigned8.

The value shall be set to zero.

4.10.2.1.24 Coding of the field RPCPad1

This field shall be coded as data type Unsigned8.

4.10.2.1.25 Coding of the field RPCWindowSize

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field RPCDRep (little endian or big endian).

4.10.2.1.26 Coding of the field RPCMaxTsdU

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.1.27 Coding of the field RPCMaxFragSize

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.1.28 Coding of the field RPCSerialNumber

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.1.29 Coding of the field RPCSelAckLen

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.1.30 Coding of the field RPCArrayOfSelAck

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.1.31 Coding of the field RPCDataRepresentationUUID

This field shall be coded as data type structure containing the following elements:

- Data1 as Unsigned32
- Data2 as Unsigned16
- Data3 as Unsigned16
- Data4 as array of eight Unsigned8

The octet ordering shall be little endian with the value according to Table 252.

Table 252 – RPCDataRepresentationUUID – defined values

Value [UUID]	Value	Description
8A885D04-1CEB-11C9-9FE8-08002B104860	Data representation	Identifies the RPC data representation uniquely.

4.10.2.2 Coding section related to NDREPMaPDU

4.10.2.2.1 Coding of the field DeviceType

This field shall be coded as data type VisibleString[25]. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 25 octets.

NOTE See also the field DeviceVendorValue.

4.10.2.2.2 Coding of the field OrderID

This field shall be coded as data type VisibleString[20]. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 20 octets.

4.10.2.2.3 Coding of the field HWRevision

This field shall be coded as data type VisibleString[5]. The value shall be set manufacturer specific using the character “0”-“9” and “<Blank>”. The range is from “<Blank><Blank><Blank><Blank>0” to “99999”.

NOTE As an example HWRevision could be “<Blank><Blank><Blank><Blank>2”, “<Blank>5714”, or “61214”.

This field shall show the same revision than the field IM_Hardware_Revision.

4.10.2.2.4 Coding of the field SWRevisionPrefix

This field shall be coded as data type VisibleString[1]. The value shall be set manufacturer specific using the character:

- “V” for an officially released version
- “R” for Revision
- “P” for Prototype
- “U” for Under Test (Field Test)
- “T” for Test Device

It is recommended to use “V” in conjunction with the field SWRevision containing “<Blank><Blank>0<Blank><Blank>0<Blank><Blank>0” if a submodule contains no software / firmware.

4.10.2.2.5 Coding of the field SWRevision

This field shall be coded as data type VisibleString[9]. The value shall be set manufacturer specific using the character “0”-“9” and “<Blank>”. The range is from “<Blank><Blank>0<Blank><Blank>0<Blank><Blank>0” to “999999999”.

As an example SWRevision could be “<Blank><Blank>1<Blank><Blank>0<Blank><Blank>0”. The presentation layer may insert a “.” after each group of three octets for better visualization.

4.10.2.2.6 Coding of the field Blank

This field shall be coded as data type OctetString with 1 octet.

The value shall be set to 0x20.

4.10.2.2.7 Coding of the field **RPCInquiryType**

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The coding shall be according to Table 253.

Table 253 – RPCInquiryType

Value (hexadecimal)	Meaning	Using
0x00000000	Read all registered interfaces with objects.	Mandatory
0x00000001	Read all objects for one registered interface.	Optional
0x00000002	Read all interfaces including a dedicated object.	Optional
0x00000003	Read one dedicated interface with one dedicated object.	Optional
0x00000004 – 0xFFFFFFFF	Reserved	—

4.10.2.2.8 Coding of the field **RPCObjectReference**

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to 1.

4.10.2.2.9 Coding of the field **RPCInterfaceReference**

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field RPCDRep (little endian or big endian).

The value shall be set to 2.

4.10.2.2.10 Coding of the field **RPCInterfaceVersionMajor**

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

4.10.2.2.11 Coding of the field **RPCInterfaceVersionMinor**

This field shall be coded as data type Unsigned16. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

4.10.2.2.12 Coding of the field **RPCVersionOption**

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to 1.

4.10.2.2.13 Coding of the field **RPCEntryHandleAttribute**

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to 0.

4.10.2.2.14 Coding of the field RPCEntryHandleUUID

This field shall be coded as data type UUID. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.2.15 Coding of the field RPCMaxEntries

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set at least to 1.

4.10.2.2.16 Coding of the field RPCNumberOfEntries

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.2.17 Coding of the field RPCEntriesOffset

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set to 0.

4.10.2.2.18 Coding of the field RPCEntriesCount

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.2.19 Coding of the field RPCTowerReference

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set to 3.

4.10.2.2.20 Coding of the field RPCAnnotationOffset

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set to 0.

4.10.2.2.21 Coding of the field RPCAnnotationLength

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set from 0 to 64.

4.10.2.2.22 Coding of the field EndTerm

This field shall be coded as data type Unsigned8. The value shall be set to 0.

4.10.2.2.23 Coding of the field RPCGap

This field shall be coded as data type Unsigned8.

4.10.2.2.24 Coding of the field RPCTowerLength

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.2.25 Coding of the field RPCTowerOctetStringLength

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

4.10.2.2.26 Coding of the field RPCEPMapStatus

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The coding shall be according to Table 254.

Table 254 – RPCEPMapStatus

Value (hexadecimal)	Meaning
0x00000000	RPC okay
0x16C9A0D6	Endpoint not registered
others	Reserved

4.10.2.2.27 Coding of the field RPCFloorCount

This field shall be coded as data type Unsigned16. The encoding shall be independently of the field RPCDRep always little endian.

The value shall be set to 5.

4.10.2.2.28 Coding of the field RPCLHSByteCount

This field shall be coded as data type Unsigned16. The encoding shall be independently of the field RPCDRep always little endian.

4.10.2.2.29 Coding of the field RPCRHSByteCount

This field shall be coded as data type Unsigned16. The encoding shall be independently of the field RPCDRep always little endian.

4.10.2.2.30 Coding of the field RPCID

This field shall be coded as data type Unsigned8.

The value shall be set to 0x0D.

4.10.2.2.31 Coding of the field RPCProtocolIdentifier

This field shall be coded as data type Unsigned8.

The value shall be set to 0x0A.

4.10.2.2.32 Coding of the field RPCServerUDPPort

This field shall be coded as data type Unsigned8.

The value shall be set to 0x08.

4.10.2.2.33 Coding of the field RPCHostAddress

This field shall be coded as data type Unsigned8.

The value shall be set to 0x09.

4.10.2.2.34 Coding of the field RPCPortNumber

This field shall be coded as data type Unsigned16. The encoding shall be independently of the field RPCDRep always big endian.

4.10.2.2.35 Coding of the field RPCIPAddress

This field shall be coded as data type Unsigned32. The encoding shall be independently of the field RPCDRep always big endian.

The value may be zero.

4.10.2.3 Coding section related to NDRData

4.10.2.3.1 Coding of the field ArgsMaximum

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set to the maximum buffer size available for the response.

4.10.2.3.2 Coding of the field ArgsLength

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set to the number of octets within the PROFINETIOServiceReqPDU or PROFINETIOServiceResPDU.

4.10.2.3.3 Coding of the field MaximumCount

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

In case of a request the value shall be set to the same value as in the field ArgsMaximum. Within the response the value shall be taken from the field ArgsMaximum of the appropriate request.

In case of an inconsistency in the coding of the "Uni-dimensional Conformant-varying Array" the packet should be rejected on the server side and an error should be reported locally on the client side, as stated in The Open Group — Publication C706.

4.10.2.3.4 Coding of the field Offset

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field "RPCDRep.IntegerEncoding" (little endian or big endian).

The value shall be set to zero.

4.10.2.3.5 Coding of the field ActualCount

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The value shall be set to the same value as in the field ArgsLength.

4.10.2.3.6 Coding of the field PNIOStatus

This field shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian) within the first field of the NDRDataResponse. In all other cases the octet ordering shall be big endian.

The content is defined in 5.2.6. The PNIOStatus shall be calculated according to the following formula.

$$\begin{aligned} \text{PNIOStatus} = & \text{ErrorCode} \times 16\,777\,216 + \\ & \text{ErrorDecode} \times 65\,536 + \\ & \text{ErrorCode1} \times 256 + \\ & \text{ErrorCode2} \end{aligned} \quad (44)$$

where

ErrorCode is the error code value
ErrorDecode is the error decode value
ErrorCode1 is the error code1 value
ErrorCode2 is the error code2 value

4.10.2.4 Coding section related to RPC (NCA Codes)

4.10.2.4.1 Coding of the field RPCNCAFaultStatus

The RPC specific NCA error codes shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The values shall be according to Table 255.

Table 255 – Values of NCAFaultStatus

Value (hexadecimal)	Definition
0x1C000001	NCA_s_fault_int_div_by_zero
0x1C000002	NCA_s_fault_addr_error
0x1C000003	NCA_s_fault_fp_div_zero
0x1C000004	NCA_s_fault_fp_underflow
0x1C000005	NCA_s_fault_fp_overflow
0x1C000006	NCA_s_fault_invalid_tag
0x1C000007	NCA_s_fault_invalid_bound
0x1C000008	NCA_s_rpc_version_mismatch
0x1C000009	NCA_s_unspec_reject
0x1C00000A	NCA_s_bad_actid

Value (hexadecimal)	Definition
0x1C00000B	NCA_s_who_are_you_failed
0x1C00000C	NCA_s_manager_not_entered
0x1C00000D	NCA_s_fault_chancel
0x1C00000E	NCA_s_fault_ill_inst
0x1C00000F	NCA_s_fault_fp_error
0x1C000010	NCA_s_fault_int_overflow
0x1C000012	NCA_s_fault_unspec
0x1C000013	NCA_s_fault_remote_comm_failure
0x1C000014	NCA_s_fault_pipe_empty
0x1C000015	NCA_s_fault_pipe_closed
0x1C000016	NCA_s_fault_pipe_order
0x1C000017	NCA_s_fault_pipe_discipline
0x1C000018	NCA_s_fault_pipe_comm_error
0x1C000019	NCA_s_fault_pipe_memory
0x1C00001A	NCA_s_fault_context_mismatch
0x1C00001B	NCA_s_fault_remote_no_memory
0x1C00001C	NCA_s_invalid_pres_context_id
0x1C00001D	NCA_s_unsupported_authn_level
0x1C00001F	NCA_s_invalid_checksum
0x1C000020	NCA_s_invalid_crc
0x1C000021	NCA_s_fault_user_defined
0x1C000022	NCA_s_fault_tx_open_failed
0x1C000023	NCA_s_fault_codeset_conv_error
0x1C010001	NCA_s_comm_failure
0x1C010015	NCA_s_fault_string_too_long

4.10.2.4.2 Coding of the field RPCNCARejectStatus

The RPC specific NCA error codes shall be coded as data type Unsigned32. The octet ordering shall be according to the value of the field “RPCDRep.IntegerEncoding” (little endian or big endian).

The values shall be according to Table 256.

Table 256 – Values of NCARejectStatus

Value (hexadecimal)	Definition
0x1C000008	NCA_rpc_version_mismatch
0x1C000009	NCA_unspec_reject
0x1C00000A	NCA_s_bad_actid
0x1C00000B	NCA_who_are_you_failed
0x1C00000C	NCA_manager_not_entered
0x1C010002	NCA_op_rng_error
0x1C010003	NCA_unk_if

Value (hexadecimal)	Definition
0x1C010006	NCA_wrong_boot_time
0x1C010009	NCA_s_you_crashed
0x1C01000B	NCA_proto_error
0x1C010013	NCA_out_args_too_big
0x1C010014	NCA_server_too_busy
0x1C010017	NCA_unsupported_type
0x1C00001D	NCA_unsupported_authn_level
0x1C00001F	NCA_invalid_checksum
0x1C000020	NCA_invalid_crc

4.10.3 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.10.4 Application Relationship Protocol Machines

4.10.4.1 RPC Protocol Machine

4.10.4.1.1 Primitive definitions

4.10.4.1.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by RPC are described in the service definition and shown in Table 257.

Table 257 – Remote primitives issued or received by RPC

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.10.4.1.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by RPC are described in the service definition and shown in Table 258.

Table 258 – Local primitives issued or received by RPC

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.10.4.1.2 State transition diagram

The state transition diagram of RPC is defined in the Technical standard DCE1.1.

4.10.4.1.3 State machine description

The state machine description is defined in the Technical standard DCE1.1.

4.10.4.1.4 RPC state table

The state table is defined in the Technical standard DCE1.1.

4.10.4.1.5 Functions, Macros, Timers and Variables

The functions are defined in the Technical standard DCE1.1.

4.10.4.2 Monitoring of services

The Ping service initiated by a service requester is called to monitor the health of the responder. The Cancel service may be repeated up to three times.

Timeout Ping

The value shall be 2 s.

NOTE 1 The ping service is used if the request is completely sent and the response is still pending.

Timeout FRAG

The value shall be 2 s.

NOTE 2 The transmission of a fragment will be repeated up to three times after 2 s if there is no acknowledge.

Timeout Cancel

The value shall be 1 s.

NOTE 3 The Cancel service will be repeated up to three times after 1 s if there is no reaction.

Timeout Resend, Timeout Ack, Timeout Broadcast, Timeout IDLE, Timeout WAIT

The values don't care.

NOTE 4 The above described parameters are defined with their values in DCE RPC.

4.10.5 DLL Mapping Protocol Machines

There is no DLL Mapping Protocol Machine (DMPM) defined for this Protocol.

4.11 Link layer discovery

4.11.1 FAL common syntax description

4.11.1.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.11.1.2 LLDP APDU abstract syntax

Table 259 defines the abstract syntax of the LLDP PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 259 shall represent the content of the DLSDU in Table 5.

Table 259 – LLDP APDU syntax

APDU name	APDU structure
LLDP-PDU	LLDPChassis, LLDPPort, LLDP TTL, LLDP-PNIO-PDU, LLDPEnd
LLDP-PNIO-PDU	{ LLDP_PNIO_DELAY ^a , LLDP_PNIO_PORTSTATUS, [LLDP_PNIO_ALIAS], LLDP_PNIO_MRPPORTSTATUS ^b , LLDP_PNIO_CHASSIS_MAC, LLDP8023MACPHY ^d , LLDPManagement, LLDP_PNIO_PTCPSTATUS ^c , [LLDPOption*] ^e , [LLDP8021*], [LLDP8023*] }
^a Shall only exist, if LineDelay measurement is supported. ^b Shall only exist, if MRP is activated for this port. ^c Shall only exist, if PTCP is activated by means of the PDSyncData Record. ^d Shall only exist, if IEEE 802.3 is used. ^e Other LLDP options may be used concurrently.	

Table 260 defines structures for substitutions of elements of the APDU structures shown in Table 259.

Table 260 – LLDP substitutions

Substitution name	Structure
LLDPChassis with MultipleInterfaceMode.- NameOfDevice:=0	LLDPChassisStationName ^ LLDPChassisMacAddress ^a
LLDPChassis with MultipleInterfaceMode.- NameOfDevice:=1	LLDPChassisStationName ^ LLDPChassisMacAddress ^c
LLDPChassisStationName	LLDP_TLVHeader ^b , LLDP_ChassisIDSubType(7) ^b , LLDP_ChassisID
LLDPChassisMacAddress	LLDP_TLVHeader ^b , LLDP_ChassisIDSubType(4) ^b , (CMResponderMacAdd ^ CMInitiatorMacAdd) ^d
LLDPPort	LLDP_TLVHeader ^b , LLDP_PortIDSubType(7) ^b , LLDP_PortID
LLDPTTL	LLDP_TLVHeader ^b , LLDP_TimeToLive(20) ^b
LLDP_PNIO_Header	LLDP_TLVHeader ^b , LLDP_OUI(00-0E-CF)
LLDP_PNIO_DELAY	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x01), PTCP_PortRxDelayLocal, PTCP_PortRxDelayRemote PTCP_PortTxDelayLocal, PTCP_PortTxDelayRemote, CableDelayLocal
LLDP_PNIO_PORTSTATUS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x02), RTClass2_PortStatus, RTClass3_PortStatus
LLDP_PNIO_ALIAS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x03), AliasNameValue
LLDP_PNIO_MRPPORTSTATUS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x04), MRP_DomainUUID, MRRT_PortStatus
LLDP_PNIO_CHASSIS_MAC	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x05), (CMResponderMacAdd ^ CMInitiatorMacAdd) ^d
LLDP_PNIO_PTCPSTATUS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x06), PTCP_MasterSourceAddress ^e , PTCP_SubdomainUUID ^f , IRDataUUID ^g , LLDP_LengthOfPeriod ^g , LLDP_RedOrangePeriodBegin ^g , LLDP_OrangePeriodBegin ^g , LLDP_GreenPeriodBegin ^g
LLDPEnd	LLDP_TLVHeader ^b (0)
LLDP8021	LLDP_TLVHeader ^b , LLDP_OUI(00-80-C2) ^h , LLDP_8021_SubType ^h , Data ^h
LLDP8023	LLDP_TLVHeader ^b , LLDP_OUI(00-12-0F) ⁱ , LLDP_8023_SubType ⁱ , Data ⁱ
LLDP8023MACPHY	LLDP_TLVHeader ^b , LLDP_OUI(00-12-0F) ⁱ , LLDP_8023_SubType(1) ⁱ , LLDP_8023_AUTONEG ⁱ , LLDP_8023_PMDCAP ⁱ , LLDP_8023_OPMAU ⁱ
LLDPManagement	LLDP_TLVHeader ^b , LLDP_ManagementData ^j

^a LLDPChassisMacAddress shall be used if no NameOfStation is assigned.
^b The encoding of the fields shall be according to IEEE 802.1AB-2005.
^c LLDPChassisMacAddress shall only be used if the NameOfDevice is equal to the NameOfStation and no NameOfStation is assigned.
^d Shall be the interface MAC address of the transmitting node.
^e Shall be set zero, if unknown.
^f Shall be PTCP_SubdomainUUID of PTCP_SyncID := 0. Otherwise the value shall be zero.
^g Shall be set zero, if unknown.
^h Shall be set according to IEEE 802.1AB-2005 Annex F.
ⁱ Shall be set according to IEEE 802.1AB-2005 Annex G.
^j Shall be set according to IEEE 802.1AB-2005, 9.5.9. It is recommended to set the object identifier according to 4.17.2.

NOTE There are different kinds of MAC addresses. The port MAC address used as SourceAddress and the interface MAC address used as CMResponderMacAdd or CMInitiatorMacAdd.

4.11.2 LLDP transfer syntax

4.11.2.1 General

As an extension to IEEE 802.1AB-2005, 10.5.3.1 this standard defines that value of the LLDP field txDelayWhile shall be zero. In this case, every node transmits LLDP PDUs on data change.

4.11.2.2 Coding of the field LLDP_ChassisID

This field shall be coded as data type OctetString according to Table 261.

NOTE The field LLDP_ChassisID is not terminated by zero.

Table 261 – LLDP_ChassisID in conjunction with MultipleInterfaceMode.NameOfDevice and NameOfStation

Multiple-InterfaceMode. NameOfDevice	NameOfStation exist	Data type	Meaning	
0	Yes	OctetString[]	This field shall be coded with 1 to 240 octets according to 4.3.1.4.15.	
0	No	—	—	
1	Yes	OctetString[]	Mandatory ^a	This field should be coded with “DeviceType, Blank, OrderID, Blank, IM_Serial_Number, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision”
			Optional ^b	This field shall be coded with 1 to 240 octets according to 4.3.1.4.15.
1	No	OctetString[]	Mandatory ^a	This field should be coded with “DeviceType, Blank, OrderID, Blank, IM_Serial_Number, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision”
			—	—

^a Mandatory for nodes with multiple interfaces and recommended for all other nodes.

^b Optional for nodes with single interface which will never have multiple interfaces.

4.11.2.3 Coding of the field LLDP_PortID

This field shall be coded as data type OctetString according to Table 262.

NOTE The field LLDP_PortID is not terminated by zero.

Table 262 – LLDP_PortID in conjunction with MultipleInterfaceMode.NameOfDevice

Multiple-InterfaceMode.NameOfDevice	Data type	Meaning
0	OctetString[8] or OctetString[14]	The field LLDP_PortID shall be coded according to 4.3.1.4.16
1	OctetString[]	The field LLDP_PortID shall be coded according to 4.3.1.4.17.1

4.11.2.4 Coding of the field LLDP_PNIO_SubType

This field shall be coded as data type Unsigned8 and shall be set according to Table 263.

Table 263 – LLDP_PNIO_SubType

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Measured delay values
0x02	Port Status
0x03	Alias
0x04	MRP Port Status
0x05	Interface MAC address
0x06	PTCP Status
0x07 – 0xFF	Reserved

4.11.2.5 Coding of the field PTCP_PortRxDelayLocal

This field shall be coded as data type Unsigned32, the time base shall be 1 ns and shall be set according to Table 264.

Table 264 – PTCP_PortRxDelayLocal

Value (hexadecimal)	Meaning
0x00	Unknown
0x01 – 0x00000FFF	Local RX port delay
0x00001000 – 0xFFFFFFFF	Reserved

4.11.2.6 Coding of the field PTCP_PortRxDelayRemote

This field shall be coded as data type Unsigned32, the time base shall be 1 ns and shall be set according to Table 265.

Table 265 – PTCP_PortRxDelayRemote

Value (hexadecimal)	Meaning
0x00	Unknown
0x01 – 0x00000FFF	Remote RX port delay
0x00001000 – 0xFFFFFFFF	Reserved

4.11.2.7 Coding of the field PTCP_PortTxDelayLocal

This field shall be coded as data type Unsigned32, the time base shall be 1 ns and shall be set according to Table 266.

Table 266 – PTCP_PortTxDelayLocal

Value (hexadecimal)	Meaning
0x00	Unknown
0x01 – 0x00000FFF	Local TX port delay
0x00001000 – 0xFFFFFFFF	Reserved

4.11.2.8 Coding of the field PTCP_PortTxDelayRemote

This field shall be coded as data type Unsigned32, the time base shall be 1 ns and shall be set according to Table 267.

Table 267 – PTCP_PortTxDelayRemote

Value (hexadecimal)	Meaning
0x00	Unknown
0x01 – 0x00000FFF	Remote TX port delay
0x00001000 – 0xFFFFFFFF	Reserved

4.11.2.9 Coding of the field CableDelayLocal

This field shall be coded as data type Unsigned32, the time base shall be 1 ns and shall be set according to Table 268.

Table 268 – CableDelayLocal

Value (hexadecimal)	Meaning
0x00	Unknown
0x01 – 0x000FFFFF	Measured cable delay
0x00100000 – 0xFFFFFFFF	Reserved

4.11.2.10 Coding of the field RTClass2_PortStatus

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: RTClass2_PortStatus.State

This field shall be set according to Table 269 or Table 270.

Table 269 – RTClass2_PortStatus.State with ARProperties.StartupMode == Legacy

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Not used or configured if LLDP_OrangePeriodBegin.Valid is equal one.
0x01	Reserved	—
0x02	RTCLASS2_RUN	ORANGE Phase activated for transmission and reception of RTClass2 Frames
0x03	Reserved	—

Table 270 – RTClass2_PortStatus.State with ARProperties.StartupMode == Advanced

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Not used
other	Reserved	—

Bit 2 – 15: RTClass2_PortStatus.reserved

This field shall be set according to 3.4.2.2.

4.11.2.11 Coding of the field RTClass3_PortStatus

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 2: RTClass3_PortStatus.State

This field shall be set according to Table 271.

Table 271 – RTClass3_PortStatus.State

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Not used or configured if LLDPRedOrangePeriodBegin. Valid is equal one. RED period is deactivated.
0x01	Reserved	—
0x02	RTCLASS3_UP	RED period activated for transmission of RTClass3 Frames. Expected next state: RTCLASS3_RUN
0x03	Reserved	—
0x04	RTCLASS3_RUN	RED period activated for transmission and reception of RTClass3 Frames
0x05 – 0x07	Reserved	—

Bit 3 – 11: RTClass3_PortStatus.reserved

This field shall be set according to 3.4.2.2.

Bit 12: RTClass3_PortStatus.Fragmentation

This field shall be set according to Table 272.

Table 272 – RTClass3_PortStatus.Fragmentation

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Default The fragmentation mode for this port is disabled.
0x01	ON	The fragmentation mode for this port is enabled.

The information whether the fragmentation is activated, shall be derived from the PDIRGlobalData and the SendClockFactor.

Bit 13: RTClass3_PortStatus.PreambleLength

This field shall be set according to Table 273.

Table 273 – RTClass3_PortStatus.PreambleLength

Value (hexadecimal)	Meaning	Usage
0x00	Seven octets	Default The PHY uses seven octets preamble when transmitting
0x01	One octet	The PHY uses one octet preamble when transmitting

It is recommended, that a PHY / MAC supports one octet preamble length when sending using the MAUType 100BASETXFD.

A PHY / MAC should support one octet preamble length when receiving using the MAUType 100BASETXFD.

Table 274 contains the truth table used for the shortening of the preamble.

Table 274 – Truth table for shortening of the preamble

Input		Output
RTC3PSM	AdjustPreambleLength. PreambleLength.Length	RTClass3_PortStatus. PreambleLength
EXPECTED_REMOTE(LLDP_ChassisID) == REMOTE(LLDP_ChassisID) AND EXPECTED_REMOTE(LLDP_PortID) == REMOTE(LLDP_PortID) AND EXPECTED_REMOTE(IRDataUUID) == REMOTE(LLDP_IRDataUUID) AND EXPECTED_LOCAL(MAUType) == LOCAL(MAUType)	Seven Octets One Octet	Seven Octets One Octet
! (EXPECTED_REMOTE(LLDP_ChassisID) == REMOTE(LLDP_ChassisID) AND EXPECTED_REMOTE(LLDP_PortID) == REMOTE(LLDP_PortID) AND EXPECTED_REMOTE(IRDataUUID) == REMOTE(LLDP_IRDataUUID) AND EXPECTED_LOCAL(MAUType) == LOCAL(MAUType))	—	Seven Octets

Bit 14: RTClass3_PortStatus.reserved

This field shall be set according to 3.4.2.2.

Bit 15: RTClass3_PortStatus.Optimized

This field shall be set according to Table 275.

Table 275 – RTClass3_PortStatus.Optimized

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Default
0x01	ON	Reserved

4.11.2.12 Coding of the field MRRT_PortStatus

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: MRRT_PortStatus.State

This field shall be set according to Table 276.

Table 276 – MRRT_PortStatus.State

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Not used / not configured
0x01 – 0x03	Reserved	—

Bit 2 – 15: MRRT_PortStatus.reserved

This field shall be set according to 3.4.2.2.

4.11.2.13 Coding of the field IRDataUUID

This field shall be coded as data type UUID according to Table 277.

Table 277 – IRDataUUID

Value (UUID)	Meaning	Usage
00000000-0000-0000-0000-000000000000	No RT_CLASS_3 domain	Initial value
00000000-0000-0000-0000-000000000001 – FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFFE	UUID for a RT_CLASS_3 domain	Generated by engineering and used by IO controller and IO device to identify a RT_CLASS_3 domain

4.11.2.14 Coding of the field LLDP_RedOrangePeriodBegin

As defined for RT_CLASS_3 the usage of a port is divided into different time periods. Also the usage of a port is divided into transmit and receive direction. The coding of this field shall be according to 3.4.2.5 and to Figure 21. The individual bits shall have the following meaning:

Bit 0 – 30: LLDP_RedOrangePeriodBegin.Offset

This field shall be set according to Table 278.

Table 278 – LLDP_RedOrangePeriodBegin.Offset

Value (hexadecimal)	Meaning	Usage
0x00000000 – 0x003D08FF	Offset relative to the begin of the cycle in nanoseconds	Begin of the RT_CLASS_3 period of the transmit direction of the port. If a RedPeriod is defined, the value shall be derived from the minimum of RedOrangePeriodBegin, else the value shall be derived from ReservedIntervalBegin.
0x003D0900 – 0x7FFFFFFF	Reserved	—

Bit 31: LLDP_RedOrangePeriodBegin.Valid

This field shall be set according to Table 279.

Table 279 – LLDP_RedOrangePeriodBegin.Valid

Value (hexadecimal)	Meaning	Usage
0x00	Invalid	The value of LLDP_RedOrangePeriodBegin.Offset is not valid. It shall be set to zero.
0x01	Valid	The value of LLDP_RedOrangePeriodBegin.Offset is valid.

4.11.2.15 Coding of the field LLDP_OrangePeriodBegin

NOTE Used only with ARProperties.StartupMode == Legacy.

The coding of this field shall be according to 3.4.2.5 and to Figure 21. The individual bits shall have the following meaning:

Bit 0 – 30: LLDP_OrangePeriodBegin.Offset

This field shall be set according to Table 280.

Table 280 – LLDP_OrangePeriodBegin.Offset

Value (hexadecimal)	Meaning	Usage
0x00000000 – 0x003D08FF	Offset relative to the begin of the cycle in nanoseconds	Begin of the RT_CLASS_2 period of the transmit direction of the port. If a RedPeriod is defined, the value shall be derived from the maximum of OrangePeriodBegin, else the value shall be derived from ReservedIntervalBegin.
0x003D0900 – 0x7FFFFFFF	Reserved	

Bit 31: LLDP_OrangePeriodBegin.Valid

This field shall be set according to Table 281 or Table 282.

Table 281 – LLDP_OrangePeriodBegin.Valid with ARProperties.StartupMode == Legacy

Value (hexadecimal)	Meaning	Usage
0x00	Invalid	The value of LLDP_OrangePeriodBegin.Offset is not valid. It shall be set to zero.
0x01	Valid	The value of LLDP_OrangePeriodBegin.Offset is valid.

Table 282 – LLDP_OrangePeriodBegin.Valid with ARProperties.StartupMode == Advanced

Value (hexadecimal)	Meaning	Usage
0x00	Invalid	LLDP_OrangePeriodBegin.Offset is not valid. It shall be set to zero.
other	—	Reserved

4.11.2.16 Coding of the field LLDP_GreenPeriodBegin

As defined for RT_CLASS_1, RT_CLASS_2, RT_CLASS_UDP and the other protocols the usage of a port is divided into different time periods. Also the usage of a port is divided into

transmit and receive. The coding of this field shall be according to 3.4.2.5 and to Figure 21. The individual bits shall have the following meaning:

Bit 0 – 30: LLDP_GreenPeriodBegin.Offset

This field shall be set according to Table 283.

Table 283 – LLDP_GreenPeriodBegin.Offset

Value (hexadecimal)	Meaning	Usage
0x00000000 – 0x003D08FF	Offset relative to the begin of the cycle in nanoseconds	Begin of the unrestricted period of the transmit direction of the port. The value shall be derived from ReservedIntervalEnd.
0x003D0900 – 0x7FFFFFFF	Reserved	—

Bit 31: LLDP_GreenPeriodBegin.Valid

This optional field shall be set according to Table 284.

Table 284 – LLDP_GreenPeriodBegin.Valid

Value (hexadecimal)	Meaning	Usage
0x00	Invalid	The value of LLDP_GreenPeriodBegin.Offset is not valid. It shall be set to zero.
0x01	Valid	The value of LLDP_GreenPeriodBegin.Offset is valid.

4.11.2.17 Coding of the field LLDP_LengthOfPeriod

A port is divided into different time periods. The duration of all periods is shown by this field. The coding of this field shall be according to 3.4.2.5 and to Figure 21. The individual bits shall have the following meaning:

Bit 0 – 30: LLDP_LengthOfPeriod.Length

This field shall be set according to Table 285.

Table 285 – LLDP_LengthOfPeriod.Length

Value (hexadecimal)	Meaning	Usage
0x00007A12 – 0x003D0900	Duration of a cycle in nanoseconds	The value shall be a multiple of 31 250 ns. See 5.2.4.62
0x003D0900 – 0x7FFFFFFF	Reserved	—

Bit 31: LLDP_LengthOfPeriod.Valid

This optional field shall be set according to Table 286.

Table 286 – LLDP_LengthOfPeriod.Valid

Value (hexadecimal)	Meaning	Usage
0x00	Invalid	LLDP_LengthOfPeriod.Length is not valid. It shall be set to zero.
0x01	Valid	LLDP_LengthOfPeriod.Length is valid.

4.11.2.18 Coding of the field LLDP_POINT

This field shall be coded as data type OctetString with one octet. The value shall be “.”.

NOTE The field LLDP_POINT is not terminated by zero.

4.11.2.19 Coding of the field LLDP_TimeToLive

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.20 Coding of the field LLDP_TLVHeader

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.21 Coding of the field LLDP_ChassisIDSubType

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.22 Coding of the field LLDP_PortIDSubType

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.23 Coding of the field LLDPOption

This field shall be coded according to IEEE 802.1AB-2005, 9.4.

4.11.2.24 Coding of the field LLDP_OUI

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.25 Coding of the field LLDP_8021_SubType

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.26 Coding section related to LLDP_Management**4.11.2.26.1 Coding of the field LLDP_ManagementData**

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.27 Coding section related to LLDP_8023**4.11.2.27.1 Coding of the field LLDP_8023_SubType**

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.27.2 Coding of the field LLDP_8023_AUTONEG

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.27.3 Coding of the field LLDP_8023_PMDCAP

This field shall be coded according to IEEE 802.1AB-2005.

4.11.2.27.4 Coding of the field LLDP_8023_OPMAU

This field shall be coded according to IEEE 802.1AB-2005.

4.11.3 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.11.4 Application Relation Protocol Machines

There is no Application Relation Protocol Machine (ARPM) defined for this Protocol.

4.11.5 DLL Mapping Protocol Machines

There is no DLL Mapping Protocol Machine (DMPM) defined for this Protocol.

4.12 MAC Bridges

4.12.1 Overview

The concepts according to IEEE 802.1D standard shall be applied.

According to this concept, every MAC Bridge shall not forward frames where the SourceMACAddress is its own one and shall support wire speed forwarding of frames.

Apart from IEEE 802.1D behavior, this protocol specification defines the following protocol machines to provide special forwarding actions:

- RT_CLASS_3 Forwarding Protocol Machine (RED_RELAY)
- DFP Forwarding Protocol Machine (DFP_RELAY)
- Time Synchronization Forwarding Protocol Machine (SYNC_RELAY)

Additionally the learning mechanisms of IEEE 802.1D shall be disabled according to Table 287, Table 288 and Table 289. As an optimization a previous learned entry of the FDB may be refreshed.

The using of the “Cut through” principle for the forwarding of frames needs a late error handling to avoid frame fragments in a switched network. Thus, whenever an invalid (by FCS or SFCRC16) frame is detected which is already partial forwarded, it should be shortened adding a TX-error (see Annex S) at the transmitting port.

4.12.2 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.12.3 Application Relation Protocol Machines

There is no Application Relation Protocol Machine (ARPM) defined for this Protocol.

4.12.4 DLL Mapping Protocol Machines

4.12.4.1 Forwarding Protocol Machine

4.12.4.1.1 Filtering database

According to IEEE 802 and this standard a switch contains a filtering data base (FDB) to optimize the bridging. Table 287, Table 288 and Table 289 show the list of entries.

The row ORG with an entry “YES” indicates whether this protocol is able to be received or transmitted, even if a port is excluded from the active topology by MRP or RSTP.

NOTE IEC 62439-2 if supported adds further entries in the FDB.

Table 287 – Unicast FDB entries

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Application	Bridge	ORG	Learning	Meaning
Vendor	n	Receive ^a	Filter	No	Enabled	Interface MAC address
Vendor	n+1 – n+m	Receive ^a	Filter	No	Enabled	Port MAC addresses
Others	Others	—	—	—	—	

^a If applicable

Table 288 – Multicast FDB entries

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Application	Bridge	ORG	Learning	Meaning
01-0E-CF	00-00-00	Receive ^a	Forward	No	Enabled	With FrameID=0xFEFE used for DCP-Identity-ReqPDU
01-0E-CF	00-00-01	Receive ^a	Forward	No	Enabled	With FrameID=0xFEFC used for DCP-Hello-ReqPDU
01-0E-CF	00-01-01	Receive ^a	Filter ^b	No	Disabled	RT_CLASS_3 destination multicast address
01-0E-CF	00-01-02	Receive ^a	Filter ^b	No	Disabled	RT_CLASS_3 invalid frame multicast address
01-0E-CF	00-02-00 – 00-02-1F	Receive ^a	Forward ^c	No	Enabled	RT_CLASS_2 multicast communication address
01-0E-CF	00-02-20 – 00-02-FF	Receive ^a	Forward	No	Enabled	RT_CLASS_2 multicast communication address
01-0E-CF	00-04-00	Receive ^a	Forward	No	Disabled	In conjunction with PTCP-AnnouncePDU and FrameID (=0xFF00) used for working clock synchronization
01-0E-CF	00-04-20	Receive ^a	Filter	Yes	Disabled	In conjunction with PTCP-RTSyncPDU with follow up and FrameID(=0x0020) used for working clock synchronization
01-0E-CF	00-04-40	Receive ^a	Filter	Yes	Disabled	In conjunction with PTCP-FollowUpPDU and FrameID(=0xFF20) used for working clock synchronization
01-0E-CF	00-04-80	Receive ^a	Filter	Yes	Disabled	In conjunction with PTCP-RTSyncPDU and FrameID(=0x0080) used for working clock synchronization
X3-XX-00	00-00-00	Receive ^a	Filter ^b	No	Disabled	Fast forwarding RT_CLASS_3 destination multicast address
01-80-C2	00-00-00	Receive ^a	Filter ^d	Yes	Disabled	IEEE 802.1D Rapid spanning tree protocol (RSTP)
01-80-C2	00-00-01 – 00-00-0D	Receive ^a	Filter	Yes	Disabled	IEEE 802.1D

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Applicati on	Bridge	ORG	Learning	Meaning
01-80-C2	00-00-0E	Receive ^a	Filter	Yes	Disabled	IEEE 802.1D, IEEE 802.1AS, IEEE 802.1AB (LLDP), and PTCP In conjunction with PTCP-DelayReqPDU and FrameID(=0xFF40), PTCP-DelayResPDU with follow up and FrameID(=0xFF41), PTCP-DelayFuResPDU and FrameID(=0xFF42) and PTCP-DelayResPDU without follow up and FrameID(=0xFF43) used for delay measurement
01-80-C2	00-00-0F	Receive ^a	Filter	Yes	Disabled	IEEE 802.1D
01-80-C2	00-00-10	Receive ^a	Filter	Yes	Disabled	IEEE 802.1D All LANS bridge management group address
01-00-5E	40-F8-00 – 40-FB-FF	Receive ^a	— ^e	No	Enabled	Used for multicast communication relations in conjunction with RT_CLASS_UDP
Others	Others	—	—	—	—	—
^a If applicable ^b This frames shall be discarded by the MAC Relay if RT_CLASS_3 is supported, otherwise should be discarded ^c Forwarding depends on the setting of MulticastBoundary ^d It is recommended to forward Rapid Spanning Tree Protocol (RSTP) frames in switches without RSTP support or switches with alternative media redundancy protocols like MRP ^e It is recommended to forward these frames unless RFC 4604 (IGMPv3) is supported						

Table 289 – Broadcast FDB entry

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Applicati on	Bridge	ORG	Learn- ing	Meaning
FF-FF-FF	FF-FF-FF	Receive ^a	Forward	No	Enabled	Broadcast
Others	Others	—	—	—	—	—
^a If applicable						

4.12.4.1.2 Primitive definitions

The service primitives including their associated parameters issued by MAC_RELAY user received by MAC_RELAY and vice versa are described in the MAC bridges ASE in the service definition.

4.12.4.1.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by MAC_RELAY are described in the service definition and shown in Table 290.

Table 290 – Remote primitives issued or received by MAC_RELAY

Primitive	Source	Destination	Associated parameters	Functions
MAC_RELAY_Data.req	LMPM	MAC_RELAY	Frame	—
MAC_RELAY_Data.cnf	LMPM	MAC_RELAY	Frame, Status	—
MAC_RELAY_Data.ind	MAC_RELAY	LMPM	Port, Frame, Timestamp	—
DEMUX_MAC_RELAY.ind	DEMUX	MAC_RELAY	Port, Frame	—
QueueHandler.req	MAC_RELAY	QueueHandler	Queue, Frame	—

4.12.4.1.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by MAC_RELAY are described in the service definition and shown in Table 291.

Table 291 – Local primitives issued or received by MAC_RELAY

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.12.4.1.3 State transition diagram

The state transition diagram of the MAC_RELAY shall be according to IEEE 802.1D.

4.12.4.1.4 State machine description

The state machine shall of the MAC_RELAY be according to IEEE 802.1D.

4.12.4.1.5 MAC_RELAY state table

The state table shall be according to IEEE 802.1D.

4.12.4.1.6 Functions, Macros, Timers and Variables

Table 292 contains the functions, macros, timers and variables used by the MAC_RELAY and their arguments and their descriptions.

Table 292 – Functions, Macros, Timers and Variables used by the MAC_RELAY

Name	Type	Function/Meaning
—	—	—

4.12.4.2 RT_CLASS_3 port state machine**4.12.4.2.1 General**

This part of the specification defines the utilization of the IEEE 802.1D standard.

It includes extensions for RT_CLASS_3 phase regarding forwarding of frames.

The routing mechanisms of IEEE 802.1D shall be temporarily disabled within the RT_CLASS_3 phase. In this case, the routing of RT_CLASS_3 frames shall be done according to the attributes defined by the appropriate ASE. The values of the attributes define a special routing table, which is used.

4.12.4.2.2 Primitive definitions

4.12.4.2.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Realtime Class 3 Port State Machine (RTC3PSM) are described in the service definition and shown in Table 293.

Table 293 – Remote primitives issued or received by RTC3PSM

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.12.4.2.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by RTC3PSM are described in the service definition and shown in Table 294.

Table 294 – Local primitives issued or received by RTC3PSM

Primitive	Source	Destination	Associated parameters	Functions
Remote_Systems_Data_Change_ind	LLDP	RTC3PSM	Local Port ID, List of Neighbors	—
MAU_Type_Change_ind	IEEE 802.3	RTC3PSM	Local Port ID, Local MAU Type, Local Link Status	—
Port_State_Change_ind	IEEE 802.1D	RTC3PSM	Local PortState	—
Sync_State_Change_ind	PTCP	RTC3PSM	Sync State	—
SetPortState	RTC3PSM	IEEE 802.1D	Local PortState	—

4.12.4.2.3 State transition diagram

The state transition diagram of the RTC3PSM is shown in Figure 67.

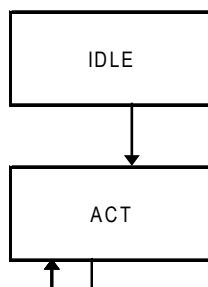


Figure 67 – State transition diagram of RTC3PSM

States of the RT3PSM

IDLE

Idle state

ACT

Active state

4.12.4.2.4 State machine description

The Realtime Class 3 Port State Machine (RTC3PSM) arranges the port states of realtime class 3.

4.12.4.2.5 RTC3PSM state table

The RTC3PSM state table is shown in Table 295 and shall exist for each port. The bandwidth allocation using the TXBeginEndAssignment and the RXBeginEndAssignment shall be coupled with the RTC3PSM port states.

Table 295 – RTC3PSM state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	=> EvaluateTruthTable() SetPortState ()	ACT
2	ACT	Remote_Systems_Data_Change_ind () => REMOTEPORTSTATE NEIGHBOR ENGINEERING EvaluateTruthTable() SetPortState ()	ACT
3	ACT	MAU_Type_Change_ind () => PORT EvaluateTruthTable() SetPortState ()	ACT
4	ACT	Sync_State_Change_ind () => SYNC EvaluateTruthTable() SetPortState ()	ACT
5	ACT	Port_State_Change_ind () => PORT EvaluateTruthTable() SetPortState ()	ACT

4.12.4.2.6 Functions, Macros, Timers and Variables

Table 296 contains the functions, macros, timers and variables used by the RTC3PSM and their arguments and their descriptions.

Table 296 – Functions, Macros, Timers and Variables used by the RTC3PSM

Name	Type	Function/Meaning
EvaluateTruthTable	Function	This function is used as a trigger to evaluate the corresponding truth table.
If activated	Function	This check is activated by a sub block of the PDPortDataCheck record.
CHECK_DATA	Macro	This function delivers an information whether the local stored data is valid.

Name	Type	Function/Meaning
DIFFERENCE_FAULT	Macro	This function checks whether the difference between the local and remote measured CableDelay is less than 50 ns.
EXPECTED_LOCAL	Macro	This function delivers the local stored information which is expected to be achieved from local.
EXPECTED_REMOTE	Macro	This function delivers the local stored information which is expected to be delivered from remote.
GET_DOMAIN_BOUNDARY_VALUE	Macro	This function delivers the local information, stored in PDIRGlobalData and PDIRBeginEndData, whether this port is used as a RT_CLASS_3 port or not.
LOCAL	Macro	This function delivers the local information.
NEIGHBOR	Macro	<pre> ! (// physical if activated: EXPECTED_REMOTE(LLDP_ChassisID) != REMOTE(LLDP_ChassisID) OR if activated: EXPECTED_REMOTE(LLDP_PortID) != REMOTE(LLDP_PortID) OR REMOTE(MAUType) != VALID ^a OR if activated: (EXPECTED_REMOTE(MAUType) != LOCAL(MAUType)) OR if activated: (LOCAL(LineDelay) > EXPECTED_LOCAL(LineDelay)) OR if activated: (EXPECTED_REMOTE(PreambleLength) != REMOTE(PreambleLength)) OR if activated: (EXPECTED_REMOTE(Fragmentation) != REMOTE(Fragmentation))) </pre> ^a This function returns for the parameter MAUType the value VALID, if the speed is greater or equal 100 Mbit/s and the duplexity is full according to 6.2.12.7.
PORT	Macro	<pre> ! (// physical LOCAL(MAUType) != VALID ^a OR if activated: (EXPECTED_LOCAL(MAUType) != LOCAL(MAUType)) OR LOCAL(PortState) == DISCARDING OR LOCAL(LinkStatus) != UP OR // logical LOCAL(GET_DOMAIN_BOUNDARY_VALUE()) == TRUE) </pre> ^a This function returns for the parameter MAUType the value VALID, if the speed is greater or equal 100 Mbit/s and the duplexity is full according to Table 551.
REMOTE	Macro	This function delivers the remote information stored locally.
REMOTEPORTSTATE	Macro	<pre> (// physical Does the remote RT_CLASS_3_Port_State exist and contain a value out of the following list {OFF, RTCLASS3_UP, RTCLASS3_RUN}) </pre>

Name	Type	Function/Meaning
SYNC	Macro	<pre> ! (// physical LOCAL(SYNC) == FALSE OR // logical if activated: (DIFFERENCE_FAULT(LOCAL(CableDelayLocal), REMOTE(CableDelayRemote)))) </pre>

4.12.4.2.7 Truth table

Table 297 and Table 298 contain the truth table used by the RTC3PSM.

Table 297 – Truth table for the RTC3PSM

Input				Output	
Conditions of the transmit port				Remote port state	Next local port state
PORT	NEIGHBOR	SYNC	ENGINEERING ^b	—	—
FALSE	—	—	—	—	OFF
TRUE	FALSE	—	—	—	OFF
TRUE	TRUE	FALSE	—	—	OFF
TRUE	TRUE	TRUE	FALSE	—	OFF
TRUE	TRUE	TRUE	TRUE	OFF	UP
TRUE	TRUE	TRUE	TRUE	UP	RUN
TRUE	TRUE	TRUE	TRUE	RUN	— ^a
^a The local state “OFF” in combination with remote port state “RUN” is not suitable, but possible as a transient state. Even in this case, the next local state should be “RUN”.					
^b PDIRData and if needed PDIRSubframeData exist and is valid.					

Table 298 – RXBeginEndAssignment and TXBeginEndAssignment

Input		Output	
Local port state	Remote port state	Local RXBeginEndAssignment	Local TXBeginEndAssignment
OFF	—	OFF	OFF
UP	—	OFF	ON ^a
RUN	UP	ON	ON ^a
RUN	RUN	ON	ON ^b
^a To protect the installation of a remote RXBeginEndAssignment the local TXBeginEndAssignment should start with the RedOrangePeriodBegin:= 0, even if the PDIRBeginEndData stated otherwise.			
^b After the installation of a remote RXBeginEndAssignment the local TXBeginEndAssignment should start with the RedOrangePeriodBegin from the PDIRBeginEndData.			

4.12.4.2.8 Generating of events

The RTC3PSM shows the state changes required to establish a RT_CLASS_3 peer to peer communication. Additional Figure 68 shows the diagnostic events, generated according to the state changes of the RTC3PSM. Table 299 defines the coding of the associated events.

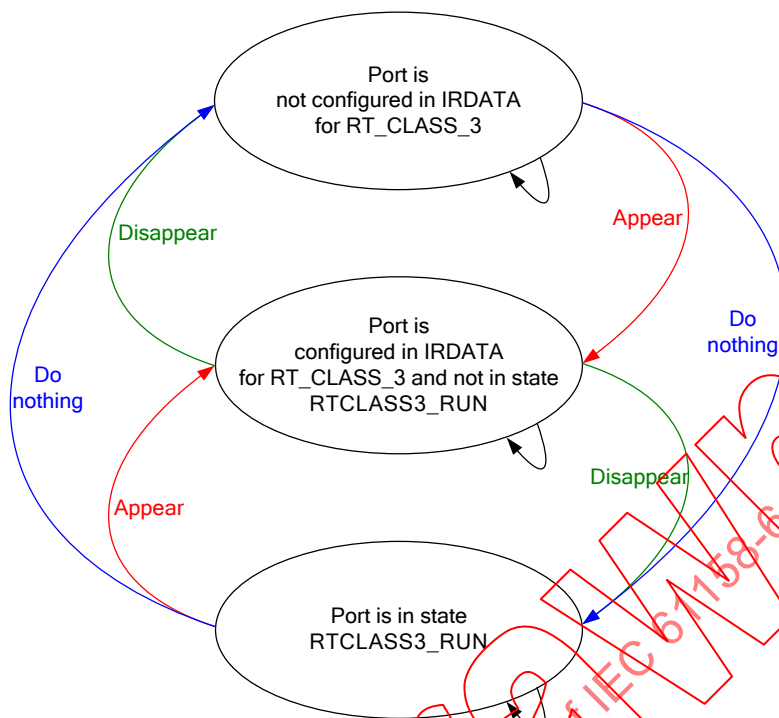


Figure 68 – State transition diagram for generating events

Table 299 – Event function table

Function name	Operations
Appear	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:=Port Data Change Notification Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelErrorType := "Remote Mismatch" ExtChannelErrorType := "RT Class3 mismatch" Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Disappear	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:=Port Data Change Notification Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Disappears ChannelErrorType := "Remote Mismatch" ExtChannelErrorType := "RT Class3 mismatch" Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Do nothing	This state change shall not indicate an event

4.12.4.3 RT_CLASS_3 Forwarding Protocol Machine

4.12.4.3.1 Primitive definitions

4.12.4.3.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Realtime Class 3 Forwarding Protocol Machine (RED_RELAY) are described in the service definition and shown in Table 300.

Table 300 – Remote primitives issued or received by RED_RELAY

Primitive	Source	Destination	Associated parameters	Functions
DMUX_RED_RELAY.ind	DFP_RELAY, DEMUX	RED_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—
RED_RELAY_Data.ind	RED_RELAY	LMPM	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—
QueueHandler.req	RED_RELAY	QueueHandler	RED	This function puts the frame in the RED queue of the QueueHandler. The MUX dequeues the frame and conveys it to the MAC.

4.12.4.3.1.2 Primitives exchanged between local machines

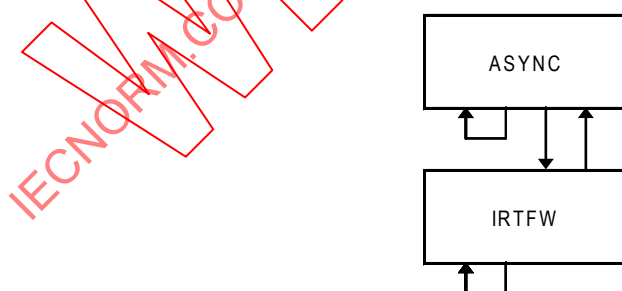
The local service primitives including their associated parameters issued or received by RED_RELAY are described in the service definition and shown in Table 301.

Table 301 – Local primitives issued or received by RED_RELAY

Primitive	Source	Destination	Associated parameters	Functions
RED_RELAY_Error_ind	RED_RELAY	Local Matter	CREP, ErrorCode	—
SyncStateChange_ind	PTCP	RED_RELAY	SYNC	—

4.12.4.3.2 State transition diagram

The state transition diagram of the RED_RELAY is shown in Figure 69.

**Figure 69 – State transition diagram of RED_RELAY**

States of the RED_RELAY

ASYNC

The node shall be synchronized, either local as a sync master or remote as a sync slave for the RED forwarding

IRTFW

If the node is synchronized, the forwarding and local reception for RED frames using the PDIRFrameData or the PDIRGlobalData and PDIRBeginEndData shall be done

4.12.4.3.3 State machine description

The Realtime Class 3 Forwarding Protocol Machine (RED_RELAY) arranges the forwarding and local reception of RED frames.

4.12.4.3.4 RED_RELAY state table

The RED_RELAY shall ensure that the forwarding of the RT_CLASS_3 frames is done in a timely correct way. This means that the frame shall be forwarded according to the given FrameSendOffset (see Figure 29 for the FrameSendOffset reference point), to work with all kinds of RedGuards.

Devices supporting the RedGuard with full check may generate surrogate frames (see Table 169 for the used values of APDU_Status.TransferStatus), if a RED_RELAY_Error_Ind is detected or a frame is missing.

The necessary data for the RED_RELAY is given by the PDIRFrameData or locally generated out of the PDIRGlobalData.

The RED_RELAY state table is shown in Table 302.

Table 302 – RED_RELAY state table

#	Current State	Event /Condition => Action	Next State
1	ASYNC	SyncStateChange_ind () /SYNC => ignore	IRTFW
2	ASYNC	SyncStateChange_ind () /!SYNC => ignore	ASYNC
3	ASYNC	TickEvent () => ignore	ASYNC
4	ASYNC	DMUX_RED_RELAY.ind () => ignore	ASYNC
5	IRTFW	SyncStateChange_ind () /SYNC => ignore	IRTFW
6	IRTFW	SyncStateChange_ind () /!SYNC => ignore	ASYNC
7	IRTFW	TickEvent (CycleCounter, CycleOffset) => Calculate Local_Phase as input for the RedGuard Calculate Local_FrameSendOffset as input for the RedGuard	IRTFW
8	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == WrongTxPortState => ErrorCode := WRONG_TX_STATE RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW

#	Current State	Event /Condition =>Action	Next State
9	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == Data length => ErrorCode := WRONG_LENGTH RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
10	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == FrameID OR Reason == Receive Port OR Reason == Phase => ErrorCode := WRONG_FRAMEID RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
11	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == FrameSendOffset => ErrorCode := WRONG_FrameSendOffset RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
12	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == FORWARD, List of TX Ports => StartTimer (IRT, Expected FrameSendOffset) For (TimerExpired (IRT), List of TX ports) if TX Port == 0 RED_RELAY_Data.ind () else QueueHandler.req ()	IRTFW

4.12.4.3.5 Functions, Macros, Timers and Variables

Table 303 contains the functions, macros, timers and variables used by the RED_RELAY and their arguments and their descriptions.

Table 303 – Functions, Macros, Timers and Variables used by the RED_RELAY

Name	Type	Function/Meaning
RedGuard	Function	This function checks the received frame against the local information.
StartTimer	Function	Starts the IRT Timer which controls the timely correct transmission of a frame
TickEvent	Function	CycleCounter, CycleOffset
TimerExpired	Function	Controls the timely correct transmission of a frame
Calculate Local_FrameSendOffset as input for the RedGuard	Macro	Calculate the input parameter Local_FrameSendOffset for the RedGuard
Calculate Local_Phase as input for the RedGuard	Macro	Calculate the input parameter Local_Phase for the RedGuard
IRT_Timer	Timer	Timer which is used to controls the timely correct transmission of a frame by generating the FrameSendOffset event
Expected FrameSendOffset	Variable	Expected or calculated FrameSendOffset of the frame
Local_FrameSendOffset	Variable	This local variable contains the actual value of the FrameSendOffset driven by TickEvent
Local_Phase	Variable	This local variable contains the actual value of the Phase driven by TickEvent

4.12.4.3.6 RedGuard

The RedGuard checks the received frame (which is “Received in RED” due to the DEMUX) against Table 304, Table 305 or Table 306. The used table depends on the local supported functionality.

Table 304 – Truth table for the RedGuard with full check

Input					Output		
Conditions of the receive port					Conditions of the transmit port		Local
FrameID ^a	Receive port	Phase	Data length	FrameSend Offset ^c	List of TX ports	Remote RTC3PSM	RED relay
FALSE	—	—	—	—	—	—	Discard ^b
TRUE	FALSE	—	—	—	—	—	Discard ^b
TRUE	TRUE	FALSE	—	—	—	—	Discard ^b
TRUE	TRUE	TRUE	FALSE	—	—	—	Discard ^b
TRUE	TRUE	TRUE	TRUE	FALSE	—	—	Discard ^b
TRUE	TRUE	TRUE	TRUE	TRUE	—	! RUN	Discard ^b
TRUE	TRUE	TRUE	TRUE	TRUE	Yes	RUN	Forward

^a The FrameID shall be checked against the PDIRFrameData and PDIRBeginEndData.
^b The bridge should discard the frame.
^c The forwarding of the frame could be done keeping its given FrameSendOffset (TRUE) or not (FALSE).

Table 305 – Truth table for the RedGuard with reduced check

Input		Output		
Conditions of the receive port		Conditions of the transmit port		Local
FrameID ^a	FrameSend Offset ^c	List of TX ports	Remote RTC3PSM	RED relay
FALSE	—	—	—	Discard ^b
TRUE	FALSE	—	—	Discard ^b
TRUE	TRUE	—	! RUN	Discard ^b
TRUE	TRUE	Yes	RUN	Forward

^a The FrameID shall be checked against PDIRFrameData and PDIRBeginEndData.
^b The bridge should discard the frame.
^c The forwarding of the frame could be done keeping its given FrameSendOffset (TRUE) or not (FALSE).

Table 306 – Truth table for the RedGuard with minimal check

Input	Output		
Conditions of the receive port	Conditions of the transmit port		Local
FrameID ^a	List of TX ports	Remote RTC3PSM	RED relay
FALSE	—	—	Discard ^b
TRUE	—	! RUN	Discard ^b

Input	Output		
Conditions of the receive port	Conditions of the transmit port		Local
FrameID ^a	List of TX ports	Remote RTC3PSM	RED relay
TRUE	Yes	RUN	Forward
^a The FrameID shall be checked against PDIRFrameData (for local received and not forwarded frames) and PDIRBeginEndData. ^b The bridge should discard the frame.			

4.12.4.4 DFP Forwarding Protocol Machine

4.12.4.4.1 General

The DFP_RELAY arranges frames with the attribute DFP. A DFP frame is a special form of an RT_CLASS_3 frame.

DFP offers two different principles, inbound and outbound, for the handling of this kind of frames. Figure 70, Figure 71 and Figure 72 shows the integration of DFP.

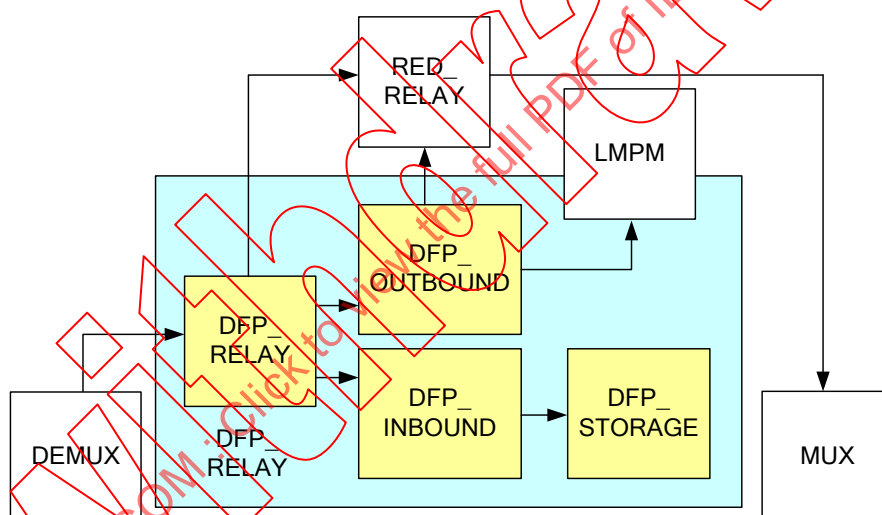


Figure 70 – Scheme of the DFP_RELAY

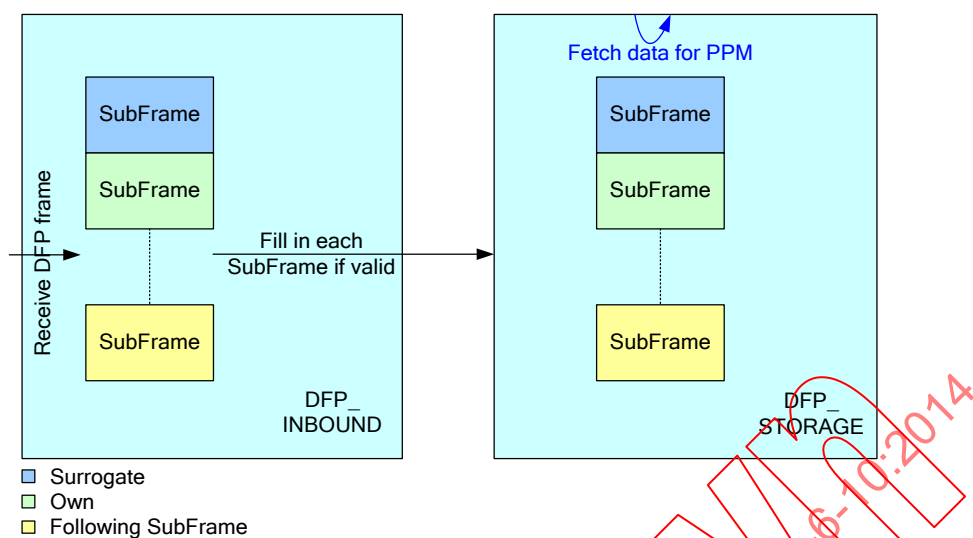


Figure 71 – Scheme of the DFP_INBOUND and DFP_STORAGE

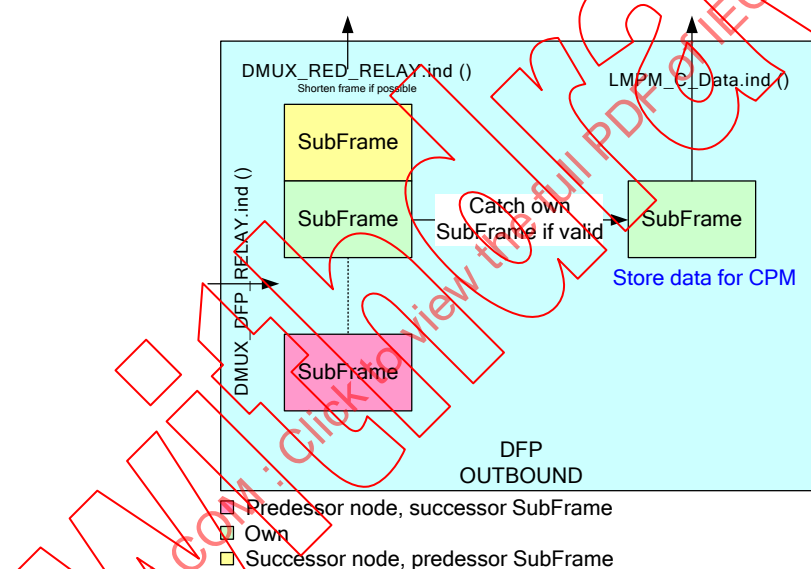


Figure 72 – Scheme of the DFP_OUTBOUND

4.12.4.4.2 DFP Protocol Machine

4.12.4.4.2.1 Primitive definitions

4.12.4.4.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DFP Protocol Machine (DFP_RELAY) are described in the service definition and shown in Table 307.

Table 307 – Remote primitives issued or received by DFP_RELAY

Primitive	Source	Destination	Associated parameters	Functions
DMUX_DFP_RELAY.ind	DEMUX	DFP_RELAY	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
DFP_INBOUND.ind	DFP_RELAY	DFP_INBOUND	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
DFP_OUTBOUND.ind	DFP_RELAY	DFP_OUTBOUND	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
DMUX_RED_RELAY.ind	DFP_RELAY	RED_RELAY	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—

4.12.4.4.2.1.2 Primitives exchanged between local machines

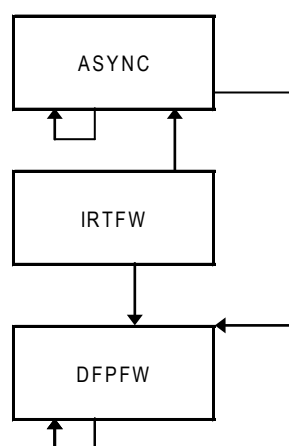
The local service primitives including their associated parameters issued or received by DFP_RELAY are described in the service definition and shown in Table 308.

Table 308 – Local primitives issued or received by DFP_RELAY

Primitive	Source	Destination	Associated parameters	Functions
SyncStateChange_ind	PTCP	DFP_RELAY	state {SYNC, ASYNC}	—

4.12.4.4.2.2 State transition diagram

The state transition diagram of the DFP_RELAY is shown in Figure 73.

**Figure 73 – State transition diagram of DFP_RELAY**

States of the DFP_RELAY

ASYNCR

The node shall be synchronized, either local as a sync master or remote as a sync slave for the DFP forwarding which is based on the RED forwarding

DFPFW

DFP forwarding active

4.12.4.4.2.3 State machine description

This state machine receives the possible DFP frames from the DEMUX and forwards them to the assigned state machine.

Unknown RT_CLASS_3 frames are forwarded to the RED_RELAY.

4.12.4.4.2.4 DFP_RELAY state table

The DFP_RELAY checks the received frame (which is “Received in RED” due to the DEMUX) against the state table shown in Table 309.

Table 309 – DFP_RELAY state table

#	Current State	Event /Condition => Action	Next State
1	ASYNCR	SyncStateChange_ind () /SYNCR => ignore	DFPFW
2	ASYNCR	SyncStateChange_ind () /!SYNCR => ignore	ASYNCR
3	DFPFW	SyncStateChange_ind () /SYNCR => ignore	DFPFW
4	DFPFW	SyncStateChange_ind () /!SYNCR => ignore	ASYNCR
5	DFPFW	DMUX_DFP_RELAY.ind () /DFPGuard () == OUTBOUND => DFP_OUTBOUND.ind ()	DFPFW
6	DFPFW	DMUX_DFP_RELAY.ind () /DFPGuard () == INBOUND => DFP_INBOUND.ind ()	DFPFW
7	DFPFW	DMUX_DFP_RELAY.ind () /DFPGuard () == RED_RELAY => DMUX_RED_RELAY.ind ()	DFPFW

4.12.4.4.2.5 Functions, Macros, Timers and Variables

Table 310 contains the functions, macros, timers and variables used by the DFP_RELAY and their arguments and their descriptions.

Table 310 – Functions, Macros, Timers and Variables used by the DFP_RELAY

Name	Type	Function/Meaning
DFPGuard	Function	This function checks the received frame against the local information.

4.12.4.4.2.6 DFPGuard

The DFPGuard checks the received frame against Table 311.

Table 311 – Truth table for the DFPGuard

Input			Output		
Local			Local		
FrameID ^a	Outbound	Inbound	RED_RELAY	OUTBOUND	INBOUND
FALSE	—	—	TRUE	—	—
TRUE	—	—	TRUE	—	—
TRUE	TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	TRUE ^b	TRUE ^b	—	—	—

^a Checked against the PDIRSubframeData.

^b Avoided by local checks.

4.12.4.4.3 DFP Inbound Protocol Machine**4.12.4.4.3.1 Primitive definitions****4.12.4.4.3.1.1 Primitives exchanged between remote machines**

The service primitives including their associated parameters issued or received by DFP Protocol Machine (DFP_RELAY_INBOUND) are described in the service definition and shown in Table 312.

Table 312 – Remote primitives issued or received by DFP_RELAY_INBOUND

Primitive	Source	Destination	Associated parameters	Functions
DFP_INBOUND.ind	DFP_RELAY	DFP_RELAY_INBOUND	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—

4.12.4.4.3.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DFP_RELAY_INBOUND are described in the service definition and shown in Table 313.

Table 313 – Local primitives issued or received by DFP_RELAY_INBOUND

Primitive	Source	Destination	Associated parameters	Functions
DFP_Activate_req	CMSU	DFP_RELAY_INBOUND	CREP	—
DFP_Close_req	CMSU	DFP_RELAY_INBOUND	CREP	—
DFP_Activate_cnf (+)	DFP_RELAY_INBOUND	CMSU	CREP	—
DFP_Close_cnf (+)	DFP_RELAY_INBOUND	CMSU	CREP	—
StoreSubframe_ind	DFP_RELAY_INBOUND	DFP_RELAY_IN_STORAGE	CREP, Subframe	Trigger the DFP STORAGE for further handling of the sub frame.

4.12.4.4.3.2 State transition diagram

The state transition diagram of the DFP_RELAY_INBOUND is shown in Figure 74.

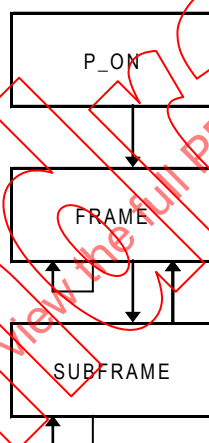


Figure 74 – State transition diagram of DFP_RELAY_INBOUND

States of the DFP_RELAY_INBOUND

P_ON	Data initialization
FRAME	Check DFP frame
SUBFRAME	Check DFP sub frame

4.12.4.4.3.3 State machine description

This state machine receives the INBOUND DFP frames, identified by it's FrameID, checks them against the protocol definition supporting the two modes DistributedWatchDog active and passive.

For MRPD, two DFP layouts exist, identified by the FrameID.

4.12.4.4.3.4 DFP_RELAY_INBOUND state table

The DFP_RELAY_INBOUND shall ensure that the reception of a DFP frames is done in a correct way and the state table is shown in Table 314.

Table 314 – DFP_RELAY_INBOUND state table

#	Current State	Event /Condition =>Action	Next State
1	P_ON	DFP_Activate_req () => DFP_Activate_cnf (+)	FRAME
2	FRAME	DFP_INBOUND.ind () /InboundGuard () == FrameStructureValid => DFP_Check_SubFrame ()	SUBFRAME
3	FRAME	DFP_INBOUND.ind () /InboundGuard () == INVALID => ignore	FRAME
4	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == SubFrameStructureValid, SubFrameValid => StoreSubFrame_ind () DFP_Check_SubFrame ()	SUBFRAME
5	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == SubFrameStructureValid, SubFrameValid, EndOfSubFrame => ignore	FRAME
6	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == SubFrameStructureValid, SubFrameDataInvalid => DFP_Check_SubFrame ()	FRAME
7	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == INVALID => ignore	FRAME
8	ANY	DFP_Close_req () => DFP_Close_cnf (+)	P_ON

4.12.4.4.3.5 Functions, Macros, Timers and Variables

Table 315 contains the functions, macros, timers and variables used by the DFP_RELAY_INBOUND and their arguments and their descriptions.

Table 315 – Functions, Macros, Timers and Variables used by the DFP_RELAY_INBOUND

Name	Type	Function/Meaning
InboundGuard	Function	This function checks the received frame against the local information.
DFP_Check_SubFrame	Function	Trigger for further frame parsing. The subframes are checked according to their sequence in the frame; each call of the function checks a particular subframe.

4.12.4.4.3.6 InboundGuard

The InboundGuard checks the received frame against Table 316, Table 317, Table 318 and Table 319.

Table 316 – Truth table for the InboundGuard – frame check

Input	Output
Header Subframe. SFCRC16	Frame check
Invalid	Invalid
Valid	Valid

Table 317 – Truth table for the InboundGuard – sub frame check

Input			Output
Subframe. Position ^a	Subframe. Length ^b	Subframe. SFCRC16 ^c	Sub frame check
Not found	—	—	Invalid
Found	FALSE	—	Invalid
Found	TRUE	Invalid	Invalid
Found	TRUE	Valid	Valid

^a Checks whether the next expected subframe is received
^b Checks whether the subframe has the expected length
^c Checks whether the SFCRC16 is valid

Table 318 – Truth table for the InboundGuard – sub frame data check

Input				Output
RestartFor-DistributedWdTime ^a	Distributed-WatchDog-Factor ^b	Subframe. DataStatus. Valid ^c	Subframe. SFCycleCounter ^d	Sub frame data check
OFF	OFF	—	—	Valid
Running	—	—	—	Invalid
Expired / OFF	ON	Invalid	—	Invalid
Expired / OFF	ON	Valid	Old	Invalid
Expired / OFF	ON	Valid	New	Valid

^a A running RestartForDistributedWdTime stops the storage of the subframe.
^b The check of the subframe data is done by the endpoint if the distributed watchdog is disabled.
^c The data is only stored, if it is valid.
^d Only newer subframes are stored.

Table 319 – Truth table for the InboundGuard – full check

Input			Output		
Frame check	Sub frame check	Sub frame data check	Frame-StructureValid	Subframe-StructureValid	Subframe-DataValid
Invalid	—	—	Invalid	—	—
Valid	Invalid	—	Valid	Invalid	—
Valid	Valid	Invalid	Valid	Valid	Invalid
Valid	Valid	Valid	Valid	Valid	Valid

4.12.4.4.4 DFP Inbound Storage Protocol Machine

4.12.4.4.4.1 General

From the consistency point of view there exists no difference between the rules for a frame or a subframe. The conveyance system shall insure that a frame is transported with frame consistency and a subframe is transported with subframe consistency.

4.12.4.4.4.2 Primitive definitions

4.12.4.4.4.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DFP Inbound Storage Protocol Machine (DFP_RELAY_IN_STORAGE) are described in the service definition and shown in Table 320.

Table 320 – Remote primitives issued or received by DFP_RELAY_IN_STORAGE

Primitive	Source	Destination	Associated parameters	Functions
AlarmNotification.ind	DFP_RELAY_IN_STORAGE	FAL	AREP, API, AlarmPriority, AlarmType, SlotNumber, SubslotNumber, AlarmSpecifier, ModuleIdentifier, SubmoduleIdentifier, AlarmItem	Inform the application that a concatenation fault takes place.

4.12.4.4.4.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DFP_RELAY_IN_STORAGE are described in the service definition and shown in Table 321.

Table 321 – Local primitives issued or received by DFP_RELAY_IN_STORAGE

Primitive	Source	Destination	Associated parameters	Functions
DFP_Activate_req	CMSU	DFP_RELAY_IN_STORAGE	CREP	—
DFP_Close_req	CMSU	DFP_RELAY_IN_STORAGE	CREP	—
DFP_Activate_cnf(+)	DFP_RELAY_IN_STORAGE	CMSU	CREP	—
DFP_Close_cnf	DFP_RELAY_IN_STORAGE	CMSU	CREP	—
DFP_Error_ind	DFP_RELAY_IN_STORAGE	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
StoreSubFrame_ind	DFP_RELAY_INBOUND	DFP_RELAY_IN_STORAGE	CREP, Subframe	Stores a received subframe on its storage

4.12.4.4.4.3 State transition diagram

The state transition diagram of the DFP_RELAY_IN_STORAGE is shown in Figure 75.

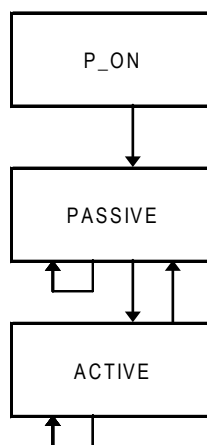


Figure 75 – State transition diagram of DFP_RELAY_IN_STORAGE

States of the DFP_RELAY_IN_STORAGE

P_ON	Data initialization
PASSIVE	Wait for first receive or for the timeout of the RestartForDistributedWDTime.
ACTIVE	Check receive of a DFP sub frame with distributed watchdog if activated. Additional check the concatenation of the subframes against the associated PPM time constraints and generate indications if necessary.

4.12.4.4.4 DFP_RELAY_IN_STORAGE state machine description

This state machine receives subframe and stores them for further use. The concatenation of the subframes to the local generated frame is done in parallel to the storing of the data.

If a fault is detected, then the transmission of the frame shall be aborted with a corresponding TX-error. In this case the last stored data of the subframe is still valid and useable.

4.12.4.4.5 DFP_RELAY_IN_STORAGE state table

The DFP_RELAY_IN_STORAGE shall control the reception of DFP frames. The corresponding state table is shown in Table 322.

Table 322 – DFP_RELAY_IN_STORAGE state table

#	Current State	Event /Condition =>Action	Next State
1	P_ON	DFP_Activate_req () => Create list of sub frames in surrogate state List of successors If activated List of distributed watchdogs DFP_Activate_cnf(+)	PASSIVE
2	PASSIVE	StoreSubFrame_ind () /RestartForDistributedWDTime (Position) != Running => Update content of the stored sub frame (Position) If activated StartTimer (Distributed WDT (Position))	ACTIVE

#	Current State	Event /Condition =>Action	Next State
3	PASSIVE	StoreSubFrame_ind () /RestartForDistributedWDTime (Position) == Running =>	PASSIVE
4	PASSIVE	TimerExpired (RestartForDistributedWDTime (Position)) => ignore	PASSIVE
5	PASSIVE	TimerExpired (Distributed WDT (Position)) => ignore	PASSIVE
6	ACTIVE	StoreSubFrame_ind () => Update content of the stored sub frame (Position) If activated StartTimer (Distributed WDT (Position))	ACTIVE
7	ACTIVE	TimerExpired (Distributed WDT (Position)) => Set stored sub frame in surrogate state (Position) StartTimer (RestartForDistributedWDTime (Position))	PASSIVE
8	ACTIVE	TimerExpired (RestartForDistributedWDTime (Position)) => ignore	ACTIVE
9	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && Sub frames updated in this Phase && !Appear sent => ignore	ACTIVE
10	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && !Sub frames updated in this Phase && Appear sent => ignore	ACTIVE
11	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && Sub frames updated in this Phase && Appear sent => Appear sent := FALSE Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= Dynamic Frame Packing problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Disappears ChannelErrorType := "Dynamic frame packing function mismatch" ExtChannelErrorType := "Frame late error" AlarmNotification.ind ()	ACTIVE
12	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && !Sub frames updated in this Phase && !Appear sent => Appear sent := TRUE Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= Dynamic Frame Packing problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelErrorType := "Dynamic frame packing function mismatch" ExtChannelErrorType := "Frame late error" AlarmNotification.ind ()	ACTIVE

#	Current State	Event /Condition =>Action	Next State
13	ANY	DFP_Close_req () => Remove list of sub frames Remove stored diagnosis from Diagnosis ASE DFP_Close_cnf ()	P_ON

4.12.4.4.4.6 Functions, Macros, Timers and Variables

Table 323 contains the functions, macros, timers and variables used by the DFP_RELAY_IN_STORAGE and their arguments and their descriptions.

Table 323 – Functions, Macros, Timers and Variables used by the DFP_RELAY_IN_STORAGE

Name	Type	Function/Meaning
Set stored sub frame in surrogate state	Function	This local function shall set the C_SDU of the sub frame to zero and the DataStatus.Ignore of the sub frame to TRUE; the other flags of the DataStatus and the SFCycleCounter should be zero.
StartTimer	Function	This local function is used to start or restart a timer.
TickEvent	Function	Local generated event from a timer. The timer shall be controlled by the PTP master or slave in case of synchronization or free running without synchronization. Parameters: CycleCounter, CycleOffset
TimerExpired	Function	This local function is issued if a timer expires.
Update content of the stored sub frame	Function	This local function updated the C_SDU, DataStatus and the SFCycleCounter of the sub frame
Phase and FrameSendOffset of the associated PPM	Macro	This local macro uses the timing information of the associated PPM for a check against the information from TickEvent.
Sub frames updated in this Phase	Macro	This local macro checks whether the sub frame was received in time to be concatenated to the local injected frame.
Distributed Watchdog Time	Timer	This timer is used to control the correct reception of valid data if activated.
RestartForDistributedWDTime	Timer	This timer is used to protect a surrogate sub frame for the detection of a fault by the IO controller. It is only active if the distributed watchdog is requested by the engineering.
Appear sent	Variable	This local variable is used to store the information whether an alarm request was already given or not.

4.12.4.4.5 DFP Outbound Protocol Machine

4.12.4.4.5.1 Primitive definitions

4.12.4.4.5.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by DFP Inbound Protocol Machine (DFP_RELAY_OUTBOUND) are described in the service definition and shown in Table 324.

Table 324 – Remote primitives issued or received by DFP_RELAY_OUTBOUND

Primitive	Source	Destination	Associated parameters	Functions
DFP_OUTBOUND.ind	DFP_RELAY_OUTBOUND	DFP_RELAY_OUTBOUND	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
LMPM_C_Data.ind	DFP_RELAY_OUTBOUND	—	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status, ReceivedInRED	Convey the own sub frame to the local interface for further handling in the associated CPM.
DMUX_RED_RELAY.ind	DFP_RELAY_OUTBOUND	RED_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—

4.12.4.4.5.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DFP_RELAY_OUTBOUND are described in the service definition and shown in Table 325.

Table 325 – Local primitives issued or received by DFP_RELAY_OUTBOUND

Primitive	Source	Destination	Associated parameters	Functions
DFP_Activate_cnf (+)	DFP_RELAY_OUTBOUND	CMSU	CREP	—
DFP_Close_cnf (+)	DFP_RELAY_OUTBOUND	CMSU	CREP	—
DFP_Activate_req	CMSU	DFP_RELAY_OUTBOUND	CREP	—
DFP_Close_req	CMSU	DFP_RELAY_OUTBOUND	CREP	—

4.12.4.4.5.2 State transition diagram

The state transition diagram of the DFP_RELAY_OUTBOUND is shown in Figure 76.

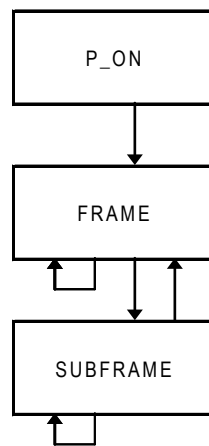


Figure 76 – State transition diagram of DFP_RELAY_OUTBOUND

States of the DFP_RELAY_OUTBOUND

P_ON	Data initialization
FRAME	Check the frame whether it has a valid structure and shall be shortened.
SUBFRAME	Search for the own subframe and shorten the frame if the structure is valid.

4.12.4.4.5.3 DFP_RELAY_OUTBOUND state machine description

This state machine receives a frame identified by the FrameID, checks whether the shortening is possible. If possible, the frame will be shortened before the own subframe and forwarded. The own subframe will be conveyed to the CPM. If not possible, the frame is forwarded to the RED_RELAY.

The DFP_RELAY_OUTBOUND shall ensure that the local portion of the DFP frames is given to the LMPM and the, if applicable, shortened frame is given to the RED_RELAY.

The frame should be shortened if SFIOCRProperties.SFCRC16 is set to one and the assigned subframe locally received after checking its SFCRC16.

The shortening shall be done by replacing the locally received subframe with a SFEndDelimiter, if needed Padding and an APDU_Status (CycleCounter:=actual local value, TransferStatus:=0, DataStatus.Ignore:=1, DataStatus.State:=1, DataStatus.DataValid:=1, DataStatus.StationProblemIndicator:=1, DataStatus.ProviderState:=1).

A locally assigned subframe shall be received after checking the FCS of the conveying frame if SFIOCRProperties.SFCRC16 is set to zero.

4.12.4.4.5.4 DFP_RELAY_OUTBOUND state table

The DFP_OUTBOUND state table is shown in Table 326.

Table 326 – DFP_RELAY_OUTBOUND state table

#	Current State	Event /Condition =>Action	Next State
1	P_ON	DFP_Activate_req () => DFP_Activate_cnf (+)	FRAME
2	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == FrameStructureValid, FrameShortenImpossible => DMUX_RED_RELAY.ind ()	FRAME
3	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == FrameStructureValid, FrameShortenPossible && FirstSubframeInFrame == OWN => DMUX_RED_RELAY.ind ()	FRAME
4	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == FrameStructureValid, FrameShortenPossible && ! FirstSubframeInFrame == OWN => DFP_Check_SubFrame ()	SUBFRAME
5	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == INVALID => ignore //NOTE The frame shall be shorten after the last valid subframe by the MUX using the appropriate TX-error	FRAME
6	SUBFRAME	DFP_Check_SubFrame () /OutboundGuard () == SubFrameStructureValid && SFPosition.Position == OWN => N_SDU := ShortenFrame (N_SDU) LMPM_C_Data.ind () DMUX_RED_RELAY.ind ()	SUBFRAME
7	SUBFRAME	DFP_Check_SubFrame () /OutboundGuard () == SubFrameStructureValid && SFPosition.Position != OWN => DFP_Check_SubFrame ()	SUBFRAME
8	SUBFRAME	DFP_Check_SubFrame () /OutboundGuard () == INVALID => ignore //NOTE The frame shall be shorten after the last valid subframe by the MUX using the appropriate TX-error	FRAME
9	ANY	DFP_Close_req () => DFP_Close_cnf (+)	P_ON

4.12.4.4.5.5 Functions, Macros, Timers and Variables

Table 327 contains the functions, macros, timers and variables used by the DFP_RELAY_OUTBOUND and their arguments and their descriptions.

Table 327 – Functions, Macros, Timers and Variables used by the DFP_RELAY_OUTBOUND

Name	Type	Function/Meaning
DFP_Check_SubFrame	Function	Trigger for further frame parsing. The subframes are checked according to their sequence in the frame; each call of the function checks a particular subframe.
OutboundGuard	Function	This local function checks the received frame against the local information.
ShortenFrame	Function	This local function shortens the N_SDU if the sequence of valid SFCRC16 is not broken.
FirstSubframeInFrame	Macro	This local macro returns the information of the first subframe in the frame. This information is needed to avoid shortening of a frame which contains only one subframe.
OWN	Variable	This local variable contains the information about the own subframe.

4.12.4.4.5.6 OutboundGuard

The OutboundGuard checks the received frame against Table 328 and Table 329.

Table 328 – Truth table for the OutboundGuard – frame check

Input			Output	
SFIOCRProperties. SFCRC16 ^a	Header Subframe. SFCRC16 ^b	Structure before own Subframe ^c	Frame structure	Frame shorten
FALSE	—	—	Valid	Impossible
TRUE	Invalid	—	Invalid	—
TRUE	Valid	Invalid	Invalid	Impossible
TRUE	Valid	Valid	Valid	Possible
^a The shortening of a frame is only possible if the SFCRC16 is valid. Thus, the subframes are only checked if the SFCRC16 shall exist. ^b The header SFCRC16 is check before the subframe checking starts. ^c The shortening is only possible if a consistent structure is found before the own subframe.				

Table 329 – Truth table for the OutboundGuard – sub frame check

Input				Output
Frame structure	Subframe. Position ^a	Subframe. Length ^b	Subframe. SFCRC16	Sub frame structure
Invalid	—	—	—	Invalid
Valid	Invalid	—	—	Invalid
Valid	Valid	Invalid	—	Invalid
Valid	Valid	Valid	Invalid	Invalid
Valid	Valid	Valid	Valid	Valid
^a Check against the protocol definition, optional against the structure from the PDIRSubframeData. ^b Check against the protocol definition, optional against the structure from the PDIRSubframeData.				

4.13 Virtual bridges

4.13.1 Overview

The concepts according to IEEE 802.1Q standard shall be applied. This part of the specification defines the utilization of the IEEE 802.1Q standard. It includes extensions for RT_CLASS_3 phase regarding forwarding of frames. The routing mechanisms of IEEE 802.1Q shall be temporary disabled within the RT_CLASS_3 phase. In this case, the routing of RT_CLASS_3 frames shall be done according to the attributes defined by the appropriate ASE. The values of the attributes define a special routing table, which is used.

Apart from IEEE 802.1Q standard behavior, this protocol specification defines the following protocol machines to provide special MAC actions:

- Queue handler Protocol Machine (QueueHandler)
- Multiplex MAC Protocol Machine (MUX)
- Demultiplex MAC Protocol Machine (DEMUX)

4.13.2 QueueHandler

The QueueHandler manages the priority queues of a port. It offers service primitives for en queuing and dequeuing of frames. Additionally information of the content of a queue is available.

4.13.3 FAL Service Protocol Machines

There is no FAL Service Protocol Machine (FSPM) defined for this Protocol.

4.13.4 Application Relation Protocol Machines

There is no Application Relation Protocol Machine (ARPM) defined for this Protocol.

4.13.5 DLL Mapping Protocol Machines

4.13.5.1 Network Access Sender protocol Machine

4.13.5.1.1 Primitive definitions

4.13.5.1.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Network Access Sender (MUX) are described in the service definition and shown in Table 330.

Table 330 – Remote primitives issued or received by MUX

Primitive	Source	Destination	Associated parameters	Functions
M_UNITDATA.req	MUX	MAC	frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, rif_information (optional), include_tag	—
M_UNITDATA.ind	MAC	MUX	time_stamp	—
LinkStateChange.ind	MAC	MUX	Port, Link_State	—

Primitive	Source	Destination	Associated parameters	Functions
PortStateChange.ind	Virtual Bridge	MUX	Port, Port_State	—

4.13.5.1.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by MUX are described in the service definition and shown in Table 331.

Table 331 – Local primitives issued or received by MUX

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.13.5.1.2 State transition diagram

The state transition diagram of the MUX is shown in Figure 77.

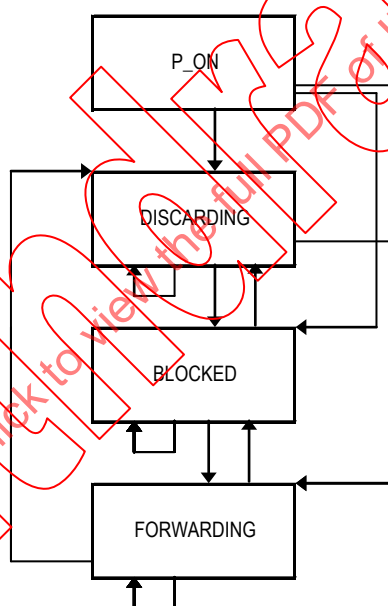


Figure 77 – State transition diagram of MUX

States of the MUX

P-ON	Wait for the startup Link and Port state and jump to the appropriate state
DISCARDING	All frames are discarded in this state
BLOCKED	Only network control or RT_CLASS_3 frames are forwarded in this state
FORWARDING	All frames are forwarded

4.13.5.1.3 State machine description

The MUX is responsible for mapping LMPM, RED_RELAY and MAC_RELAY(IEEE 802.1D) services to MAC according to IEEE 802.3.

Additional the Queue handler shown in Figure 79 is used to manage the priority queues.

4.13.5.1.4 MUX state table

The MUX state table is shown in Table 332.

Table 332 – MUX state table

#	Current State	Event /Condition =>Action	Next State
1	P_ON	/StateChecker () == FORWARDING => Period := GREEN	FORWARDING
2	P_ON	/StateChecker () == BLOCKED => Period := GREEN	BLOCKED
3	P_ON	/StateChecker () == DISCARDING => Period := GREEN	DISCARDING
4	P_ON	M_UNITDATA.cnf () => ignore	P_ON
5	DISCARDING	MUX_Schedule_req (CycleCounter, Tx_Period) => ignore	DISCARDING
6	DISCARDING	QueuesNotEmpty () => GET_FRAME_FROM_QUEUE(NwC_Hi, NwC_Lo, PRIOR7, RED, PRIOR6, ..., PRIOR2-1) DISCARD_FRAME()	DISCARDING
7	DISCARDING	M_UNITDATA.cnf () => ignore	DISCARDING
8	BLOCKED	QueuesNotEmpty () /Period == GREEN && FrameSizeFits (NwC_Hi, NwC_Lo) => GET_FRAME_FROM_QUEUE(NwC_Hi, NwC_Lo) M_UNITDATA.req ()	BLOCKED
9	BLOCKED	QueuesNotEmpty () /Period == GREEN && !FrameSizeFits (NwC_Hi, NwC_Lo) => ignore	BLOCKED
10	BLOCKED	M_UNITDATA.cnf () => ignore	BLOCKED
11	BLOCKED / FORWARDING	MUX_Schedule_req (CycleCounter, Tx_Period) => Period := Tx_Period	BLOCKED / FORWARDING
12	BLOCKED / FORWARDING	QueuesNotEmpty () /Period == RED => GET_FRAME_FROM_QUEUE(RED) M_UNITDATA.req ()	BLOCKED / FORWARDING
13	FORWARDING	QueuesNotEmpty () /Period == GREEN && FrameSizeFits (NwC_Hi, NwC_Lo, PRIOR7, PRIOR6, ..., PRIOR2-1) => GET_FRAME_FROM_QUEUE(NwC_Hi, NwC_Lo, PRIOR7, PRIOR6, ..., PRIOR2-1)	FORWARDING

#	Current State	Event /Condition =>Action	Next State
		M_UNITDATA.req ()	
14	FORWARDING	QueuesNotEmpty () /Period == GREEN && !FrameSizeFits (NwC_Hi, NwC_Lo, PRIO7, PRIO6, ..., PRIOR2-1) => ignore	FORWARDING
15	FORWARDING	M_UNITDATA.cnf () => ignore	FORWARDING
16	BLOCKED / FORWARDING	QueuesNotEmpty () /Period != RED && FrameEgressFilter () => GET_FRAME_FROM_QUEUE() DISCARD_FRAME()	BLOCKED / FORWARDING
17	ANY	LinkStateChange () /StateChecker () == DISCARDING => Period := GREEN	DISCARDING
18	ANY	PortStateChange () /StateChecker () == DISCARDING => Period := GREEN	DISCARDING
19	ANY	LinkStateChange () /StateChecker () == BLOCKED => ignore	BLOCKED
20	ANY	PortStateChange () /StateChecker () == BLOCKED => ignore	BLOCKED
21	ANY	LinkStateChange () /StateChecker () == FORWARDING => ignore	FORWARDING
22	ANY	PortStateChange () /StateChecker () == FORWARDING => ignore	FORWARDING

4.13.5.1.5 Functions, Macros, Timers and Variables

Table 333 contains the functions, macros, timers and variables used by the MUX and their arguments and their descriptions.

Table 333 – Functions, Macros, Timers and Variables used by MUX

Name	Type	Function/Meaning
FrameEgressFilter	Function	This local function checks whether the frame should be transmitted or dropped according to the IEEE 802.1D egress filtering
FrameSizeFits	Function	This local function checks if the found frame fits into the remaining period till the next StartOfRED. It was only searched in the (...) queues from highest to lowest priority. The first non fitting frame stops the search and FALSE was returned. The check includes the requirements of the corresponding truth table.

Name	Type	Function/Meaning
QueuesNotEmpty	Function	This local event is issued if an entry in any queue exists. An entry exists after a frame receive is detected and the CT or S&F switching has the transmit queue assigned.
StateChecker	Function	This local function checks the Link_State and Port_State for the possible next state and returns the result.
DISCARD_FRAME	Macro	Discards the selected frame
GET_FRAME_FROM_QUEUE	Macro	Delivers a frame out of (...) queues. This dequeuing always works from highest to lowest priority of the queues.
Period	Variable	This local variable stores the actual period for further use

4.13.5.1.6 Truth tables

4.13.5.1.6.1 FrameSizeFits rules truth table

Table 334 contains the truth table used by the MUX.

Table 334 – Truth table for FrameSizeFits

Input						Output
Period	Distance greater than MaxFrameSize	Distance greater than YellowTime	Well known protocols	Other protocols	Frame size	FrameSizeFits
GREEN	True	True	CT / S&F	CT / S&F	—	True
GREEN	False	True	CT / S&F	—	—	True
GREEN	False	True	—	S&F	Fits	True
GREEN	False	True	—	S&F	Too big	False
GREEN	False	False	S&F	S&F	Fits	True
GREEN	False	False	S&F	S&F	Too big	False
GREEN	False	False	CT	CT	—	False
RED	—	—	—	—	—	—

4.13.5.1.6.2 StateChecker truth table

Table 335 contains the truth table used by the MUX.

Table 335 – Truth table for StateChecker

Input		Output
Link_State	Port_State	StateChecker
DOWN	DISCARDING	DISCARDING
DOWN	BLOCKED	DISCARDING
DOWN	FORWARDING	DISCARDING
UP	DISCARDING	DISCARDING
UP	BLOCKED	BLOCKED
UP	FORWARDING	FORWARDING

4.13.5.2 Network Access Receiver Protocol Machine

4.13.5.2.1 Primitive definitions

4.13.5.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Network Access Sender (DEMUX) are described in the service definition and shown in Table 336.

Table 336 – Remote primitives issued or received by DEMUX

Primitive	Source	Destination	Associated parameters	Functions
M_UNITDATA.ind	MAC	DEMUX	Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier	—
DMUX_DEFRAG.ind	DEMUX	DEFRAG	Port, Tstamp, DA, SA, VLAN, N_SDU	—
DMUX_DFP_RELAY.ind	DEMUX	DFP_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—
DMUX_MAC_RELAY.ind	DEMUX		Port, Tstamp, DA, SA, N_SDU	—
DMUX_P_Data.ind	DEMUX	LMPM	Port, Tstamp, DA, SA, N_SDU	—
DMUX_RED_RELAY.ind	DEMUX	RED_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—

4.13.5.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by DEMUX are described in the service definition and shown in Table 337.

Table 337 – Local primitives issued or received by DEMUX

Primitive	Source	Destination	Associated parameters	Functions
DMUX_Schedule_req	SCHEDULER	DEMUX	Rx_Period	—
DMUX_Schedule_cnf	DEMUX	SCHEDULER	—	—

4.13.5.2.2 State transition diagram

The state transition diagram of the DEMUX is shown in Figure 78.

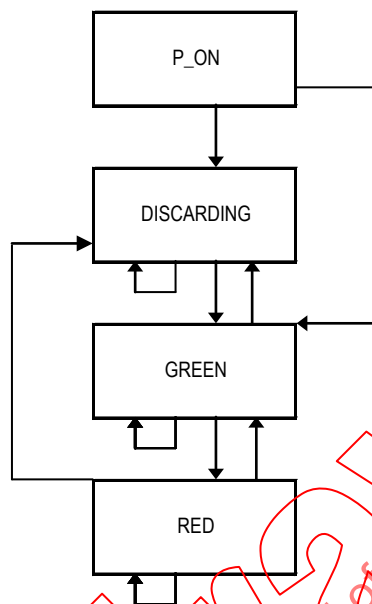


Figure 78 – State transition diagram of DEMUX

States of the DEMUX

P_ON

Power on

DISCARDING

port state discarding

GREEN

green period

RED

red period

4.13.5.2.3 State machine description

The DEMUX is responsible for mapping MAC services according to IEEE 802.3 to LPM, RED_RELAY and MAC_RELAY (IEEE 802.1D).

The generation of the flag ReceivedInRED shall be done by checking the receive TimeStamp made by the SHIM. If the value of the ReceiveTimeStamp is equal or greater the RedOrangePeriodBegin and less or equal the GreenPeriodBegin point in time, then the ReceivedInRED shall be set to TRUE, otherwise to FALSE.

4.13.5.2.4 DEMUX state table

The DEMUX state table is shown in Table 338.

Table 338 – DEMUX state table

#	Current State	Event /Condition =>Action	Next State
1	P_ON	/Port_State != Discarding => ignore	GREEN
2	P_ON	/Port_State == Discarding => ignore	DISCARDING
3	DISCARDING	DMUX_Schedule_req () => DMUX_Schedule_cnf ()	DISCARDING
4	DISCARDING	M_UNITDATA.ind () => ignore	DISCARDING
5	DISCARDING	PortStateChange () /Port_State == Discarding => ignore	DISCARDING
6	DISCARDING	PortStateChange () /Port_State != Discarding => ignore	GREEN
7	GREEN	PortStateChange () /Port_State != Discarding => ignore	GREEN
8	GREEN	PortStateChange () /Port_State == Discarding => ignore	DISCARDING
9	GREEN	DMUX_Schedule_req () /Rx_Period == GREEN => DMUX_Schedule_cnf ()	GREEN
10	GREEN	DMUX_Schedule_req () /Rx_Period == RED => DMUX_Schedule_cnf ()	RED
11	GREEN	M_UNITDATA.ind () /FrameIngressFilter // Checking the ingress filter is always the first check! => ignore	GREEN
12	GREEN	M_UNITDATA.ind () /FRAGMENT => DMUX_DEFRAG.ind ()	GREEN
13	GREEN	M_UNITDATA.ind () /OTHER OR RED OR DFP => DMUX_MAC_RELAY.ind ()	GREEN
14	GREEN	M_UNITDATA.ind () /CTRL => Get_Receive_TimeStamp () DMUX_P_Data.ind ()	GREEN

#	Current State	Event /Condition =>Action	Next State
15	RED	PortStateChange () /Port_State != Discarding => ignore	RED
16	RED	PortStateChange () /Port_State == Discarding => ignore	DISCARDING
17	RED	DMUX_Schedule_req () /Rx_Period == GREEN => DMUX_Schedule_cnf ()	GREEN
18	RED	DMUX_Schedule_req () /Rx_Period == RED => DMUX_Schedule_cnf ()	RED
19	RED	M_UNITDATA.ind () /RED => ReceivedInRED := TRUE DMUX_RED_RELAY.ind ()	RED
20	RED	M_UNITDATA.ind () /DFP => ReceivedInRED := TRUE DMUX_DFP_RELAY.ind ()	RED
21	RED	M_UNITDATA.ind () /FRAGMENT => DMUX_DEFRAG.ind ()	RED
22	RED	M_UNITDATA.ind () /OTHER => DMUX_MAC_RELAY.ind ()	RED
23	RED	M_UNITDATA.ind () /CTRL => Get_Receive_TimeStamp () DMUX_P_Data.ind ()	RED

4.13.5.2.5 Functions, Macros, Timers and Variables

Table 339 contains the functions, macros, timers and variables used by the DEMUX and their arguments and their descriptions.

Table 339 – Functions, Macros, Timers and Variables used by the DEMUX

Name	Type	Function/Meaning
Get_Receive_TimeStamp	Function	Get the receive time stamp frame the Timestamp Shim for this frame
CTRL	Macro	Network control/management e.g. PTCP synchronization frames without "RT_CLASS_3 attribute", LLDP, PTCP Line delay measurement, PTCP announce, media redundancy, ...
DFP	Macro	RT_CLASS_3 frames with "DFP attribute" EtherType == 0x8892 FrameID == 0x0100...0x0FFF
FRAGMENT	Macro	Frame using the fragmentation protocol. EtherType == 0x8892 FrameID == Fragmentation
FrameIngressFilter	Macro	Checks whether this frame is filtered by an ingress rule.

Name	Type	Function/Meaning
OTHER	Macro	Any other frame.
RED	Macro	RT_CLASS_3 frames EtherType == 0x8892 FrameID == 0x0100 ...0x0FFF Special case: PTCP synchronization frames with "RT_CLASS_3 attribute" EtherType == 0x8892 FrameID == 0x0080
PortStateChange	Function	Conveys the info about the port state

4.14 IP suite

4.14.1 Overview

The utilization of the RFC 768 (UDP), RFC 791 (IP), RFC 792 (ICMP), RFC 826 (ARP), RFC 2236 (IP Multicasting), RFC 2474 (IP Extensions) standards are defined for this specification. It includes definitions for UDP ports and IP multicast addresses and utilization of IP header fields for RT_CLASS_UDP.

4.14.2 IP/UDP syntax description

4.14.2.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

4.14.2.2 IP/UDP APDU abstract syntax

Table 340 defines the abstract syntax of the Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 340 shall represent the content of the DLSDU in Table 5.

Table 340 – IP/UDP APDU syntax

APDU name	APDU structure
ICMP-PDU	IPHeader ^a , ICMPHeader, Data*
^a The value of IPHeader.IP_Protocol shall be set to 1 to indicate ICMP.	

Table 341 defines structures for substitutions of elements of the APDU structures shown in Table 340.

Table 341 – IP/UDP substitutions

Substitution name	Structure
IPHeader	IP_VersionIHL(0x45), IP_TypeOfService, IP_TotalLength, IP_Identification, IP_Flags_FragOffset ^a , IP_TTL, IP_Protocol, IP_HeaderChecksum, IP_SrcIPAddress, IP_DstIPAddress, [IP_Options] ^b The encoding of the fields shall be according to RFC 791.
ICMPHeader	ICMP_Type, ICMP_Code, ICMP_HeaderChecksum
UDPHeader	UDP_SrcPort, UDP_DstPort, UDP_DataLength, UDP_Checksum ^c
^a For UDP-RTC-PDU and UDP-RTA-PDU IP fragmentation shall not be used. ^b The field IP_Options shall be omitted for UDP-RTC-PDU and UDP-RTA-PDU. ^c The UDP_Checksum should be set to zero for RT_CLASS_UDP and RTA_CLASS_UDP to improve performance. It is not required for the receiver to check the UDP_Checksum for UDP-RTC-PDU and UDP-RTA-PDU.	

4.14.3 IP/UDP transfer syntax

4.14.3.1 Coding section related to IP, ICMP and UDP

4.14.3.1.1 Coding section related to ICMP

4.14.3.1.1.1 Coding of the field ICMP_Type

The encoding of the fields shall be according to RFC 792.

4.14.3.1.1.2 Coding of the field ICMP_Code

The encoding of the fields shall be according to RFC 792.

4.14.3.1.1.3 Coding of the field ICMP_HeaderChecksum

The encoding of the fields shall be according to RFC 792.

4.14.3.1.2 Coding section related to UDP

4.14.3.1.2.1 Coding of the field UDP_DataLength

The encoding of the fields shall be according to RFC 768.

4.14.3.1.2.2 Coding of the field UDP_Checksum

The encoding of the fields shall be according to RFC 768.

4.14.3.1.2.3 Coding of the field UDP_SrcPort

The encoding of the fields shall be according to RFC 768. Defined values are shown in Table 342.

Table 342 – UDP_SrcPort

Value (hexadecimal)	Meaning	Usage
0x8892	IANA_PNIO_UDP_UNICAST_PORT	UDP-RTC-PDU and UDP-RTA-PDU
0x8893	IANA_PNIO_UDP_MULTICAST_PORT	Reserved
0x8894	IANA_PNIO_EPM_PORT	NDREMapLookupReq or NDREMapLookupFreeReq
0xC000 – 0xFFFF	IANA_FREE_PORT is used as dynamic and/or private port	PROFINETIOServiceReqPDU and PROFINETIOServiceResPDU

4.14.3.1.2.4 Coding of the field UDP_DstPort

The encoding of the fields shall be according to RFC 768. Defined values are shown in Table 343.

Table 343 – UDP_DstPort

Value (hexadecimal)	Meaning	Usage
0x8892	IANA_PNIO_UDP_UNICAST_PORT	UDP-RTC-PDU and UDP-RTA-PDU
0x8893	IANA_PNIO_UDP_MULTICAST_PORT	Reserved
0x8894	IANA_PNIO_EPM_PORT	NDREMapLookupReq or NDREMapLookupFreeReq, also possible for each RPC call

Value (hexadecimal)	Meaning	Usage
0xC000 – 0xFFFF	IANA_FREE_PORT Is used as dynamic and/or private port	PROFINETIOServiceReqPDU and PROFINETIOServiceResPDU

4.14.3.1.3 Coding section related to IP

4.14.3.1.3.1 Coding of the field IP_DstIPAddress

The encoding of the fields shall be according to RFC 791, RFC 2236 and RFC 2365. Defined values are shown in Table 344 and Table 345.

Table 344 – IP_DstIPAddress

IP address	Range	Meaning
224.0.0.34	subnet	SUBNET_MULTICAST_IP_ADDRESS_FOR_DISCOVERY
—	site	SITE_MULTICAST_IP_ADDRESS_FOR_DISCOVERY

Table 345 – IP Multicast DstIPAddress according to RFC 2365

Multicast IP address	Associated Multicast MAC address	Associated FrameID	Usage
239.0.0.0 — 239.192.247.255	—	—	not used
239.192.248.0	01-00-5E-40-F8-00	0xF800	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.192.248.1 — 239.192.251.254	01-00-5E-40-F8-01 — 01-00-5E-40-FB-FE	0xF801 — 0xFBFE	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.192.251.255	01-00-5E-40-FB-FF	0xFBFF	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.193.252.0 — 239.255.255.255	—	—	not used

4.14.3.1.3.2 Coding of the field IP_VersionIHL

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.3 Coding of the field IP_TypeOfService

The encoding of the fields shall be according to RFC 791 and RFC 2474.

4.14.3.1.3.4 Coding of the field IP_TotalLength

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.5 Coding of the field IP_Identification

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.6 Coding of the field IP_Flags_FragOffset

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.7 Coding of the field IP_TTL

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.8 Coding of the field IP_Protocol

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.9 Coding of the field IP_HeaderChecksum

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.10 Coding of the field IP_SrcIPAddress

The encoding of the fields shall be according to RFC 791.

4.14.3.1.3.11 Coding of the field IP_Options

The encoding of the fields shall be according to RFC 791.

4.14.4 ARP cache

The utilization of the RFC 826 (ARP) and the control of the ARP cache may be optimized to reduce the switchover time between two nodes using the same IP address at different times. Thus may be done by deleting the ARP cache entry of an IP address every time the AR is released or aborted.

4.15 Domain name system**4.15.1 General**

The utilization of the RFC 1034 (DNS) standard is defined for this specification. It includes the usage and the content syntax of domain name.

4.15.2 Primitive definitions**4.15.2.1 Primitives exchanged between remote machines**

The remote service primitives including their associated parameters issued or received by DNS are described in the service definition and shown in Table 346.

Table 346 – Remote primitives issued or received by DNS

Primitive name	Source	Destination	Associated parameters	Functions
DNS_QueryName.req	DNS	DNS client	QNAME := StationName.RealStationName or StationName.AliasStationName, QCLASS, QTYPE	Resolve name with IP address
DNS_QueryName.cnf	DNS server	DNS	QNAME := StationName.RealStationName or StationName.AliasStationName, QCLASS, QTYPE, Answer := IPAddress	Resolve name with IP address

NOTE High level interfaces may offer a GetHostByName service instead of the QNAME service.

4.15.2.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by DNS are described in the service definition and shown in Table 347.

Table 347 – Local primitives issued or received by DNS

Primitive name	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.15.3 DNS state transition diagram

See RFC 1034 (DNS).

4.15.4 State machine description

See RFC 1034 (DNS).

4.15.5 DNS state table

See RFC 1034 (DNS).

4.15.6 Functions, Macros, Timers and Variables

Table 348 contains the functions, macros, timers and variables used by the DNS and their arguments and their descriptions.

Table 348 – Functions, Macros, Timers and Variables used by the DNS

Name	Type	Function/Meaning
—	—	—

4.16 Dynamic host configuration

4.16.1 General

The utilization of the RFC 2131 (DHCP) standard is defined for this specification. It includes the usage and the content syntax of Client ID.

NOTE A tight coupling between DNS and DHCP server prevents Name vs. IP address mismatch.

4.16.2 Primitive definitions

4.16.2.1 Primitives exchanged between remote machines

The remote service primitives including their associated parameters issued or received by DHCP are described in the service definition and shown in Table 349.

Table 349 – Remote primitives issued or received by DHCP

Primitive name	Source	Destination	Associated parameters	Functions
DHCP offer	DHCP	DHCP server	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 1, 3, 51, 54, 6, ...)	Second service in a sequence to offer the DHCP client a parameter set.
DHCP acknowledgement	DHCP	DHCP server	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 1, 3, 51, 54, 6, ...)	Fourth service in a sequence to deliver the agreed parameter set.
DHCP discovery	DHCP client	DHCP	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 50, 55, ...)	First service in a sequence to find a DHCP server.

Primitive name	Source	Destination	Associated parameters	Functions
DHCP request	DHCP client	DHCP	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 50, 54, ...)	Third service in a sequence which is used by the client to select a server.

4.16.2.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by DHCP are described in the service definition and shown in Table 350.

Table 350 – Local primitives issued or received by machines

Primitive name	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.16.3 DHCP state transition diagram

See RFC 2131 (DHCP).

4.16.4 State machine description

See RFC 2131 (DHCP).

4.16.5 DHCP state table

See RFC 2131 (DHCP).

4.16.6 Functions, Macros, Timers and Variables

Table 351 contains the functions, macros, timers and variables used by the DHCP and their arguments and their descriptions.

Table 351 – Functions, Macros, Timers and Variables used by the DHCP

Name	Type	Function/Meaning
—	—	—

4.17 Simple network management

4.17.1 Overview

The utilization of the RFC 1213 (MIB 2), RFC 2674 (Bridges with traffic classes), RFC 2863 (IF MIB), RFC 3418 (SNMP), RFC 3621 (Power over Ethernet MIB), RFC 4836 (MAU MIB) and IEEE 802.1AB (LLDP MIB) standards are defined for this specification. Furthermore, the PNIO MIB is specified in 4.17.2 and LLDP EXT MIB is specified in 4.17.4.

4.17.2 Enterprise number for PNIO MIB

The coding of the field enterprise number shall be according to Table 352.

Table 352 – Enterprise number

Value (decimal)	Meaning
24 686	Enterprise number for the PNIO MIB

4.17.3 MIB cross reference

It is recommended that each (LLDP-MIB.)“lldpLocPortDesc” is defined disjunct to make a cross reference to the MIB2 possible.

The link between the MIB2 and the LLDP MIB may be done by the rule (LLDP-MIB.)“lldpLocPortDesc” == (MIB2.)“ifDescr”.

4.17.4 LLDP EXT MIB

See Annex W

4.18 Common DLL Mapping Protocol Machines

4.18.1 Overview

The DLL Mapping Protocol Machines (DMPM) consists of several protocol machines:

- Handling of the services invoked by other Application Layer entities (LMPM)
- Forwarding for RT_CLASS_3 (RED_Relay)
- Forwarding for DFP (DFP_Relay)

NOTE 1 Forwarding of standard frames is according to IEEE 802.1D and therefore not specified here.

- Mapping to Media Access Control interface (MAC) and send/receive functions for synchronization messages

NOTE 2 Invoking of MAC is according to IEEE 802.3 and therefore not specified here.

- Fragmentation and reassembly (FRAG and DEFRAG)
- SYNC_SHIMP to add precise time stamps for synchronization

The Interface between the service handler (LMPM), forwarding machines and Mapping to Media Access Control (MAC) consists of a set of queues and global data.

Figure 79 illustrates the structuring of the protocol machines for the DMPM for a device and Annex H describes the upper protocol layers.

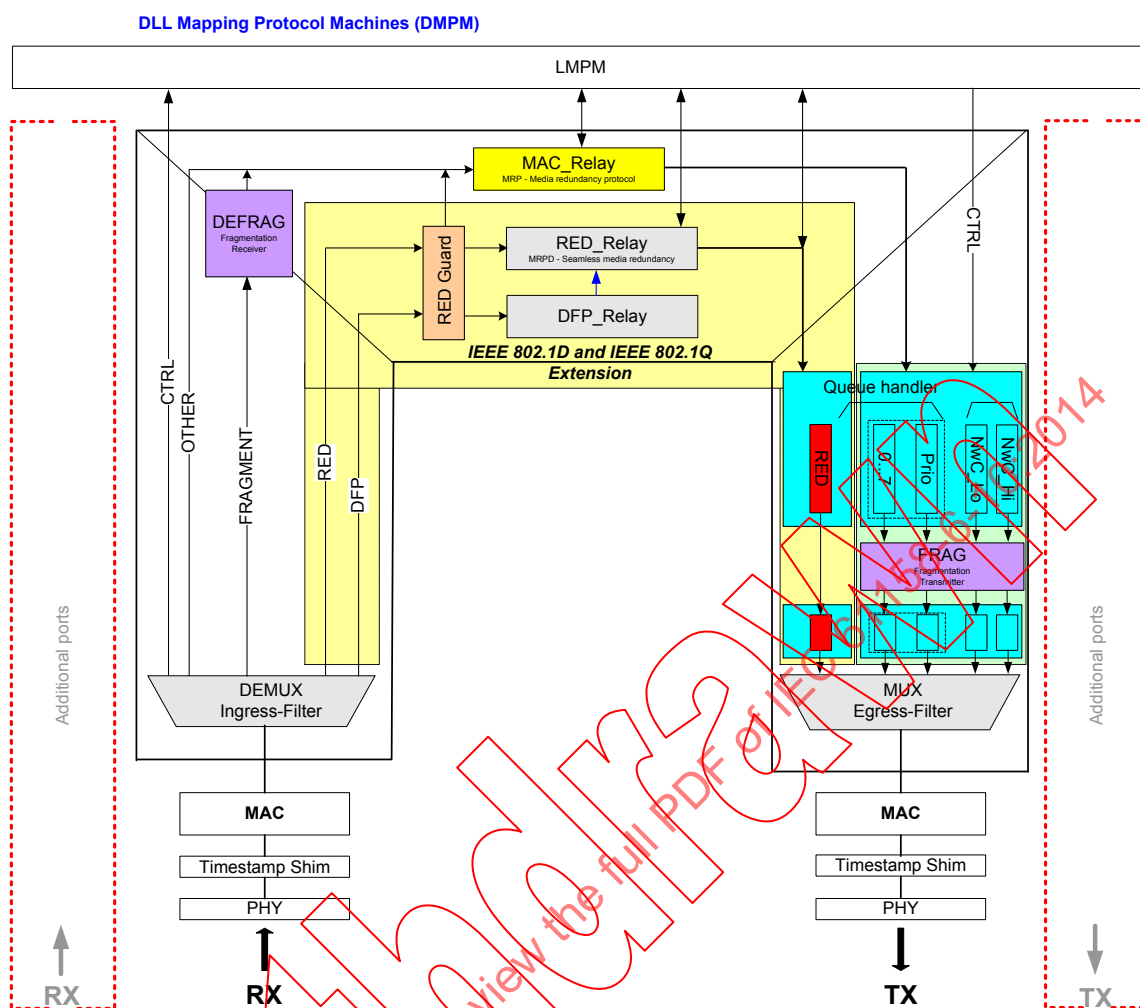


Figure 79 – Structuring of the protocol machines within the DMPM (bridge)

NOTE 3 This picture is often called baggy pants model.

4.18.2 Data Link Layer Mapping Protocol Machine

4.18.2.1 Primitive definitions

4.18.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Data Link Layer Mapping Protocol Machine (LMPM) are described in the service definition and shown in Table 353.

Table 353 – Remote primitives issued or received by LMPM

Primitive	Source	Destination	Associated parameters	Functions
DMUX_P_Data.ind	DEMUX	LMPM	CREP, S_Port, Tstamp, DA, SA, N_SDU	Network management protocols are issued direct to the LMPM.
LMPM_A_Data.cnf	LMPM	APMS APMR DCPUCS DCPUCR	CREP, LMPM_status	—

Primitive	Source	Destination	Associated parameters	Functions
		DCPMCS DCPMCR DCPHMCS DCPHMCR		
LMPM_A_Data.ind	LMPM	APMS APMR DCPUCS DCPUCR DCPMCS DCPMCR DCPHMCS DCPHMCR	CREP, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_A_Data.req	APMS APMR DCPUCS DCPUCR DCPMCS DCPMCR DCPHMCS DCPHMCR	LMPM	CREP, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_C_Data.cnf	LMPM	SCHEDULER	CREP, LMPM_status	—
LMPM_C_Data.ind	LMPM	CPM	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status	—
LMPM_C_Data.req	SCHEDULER PPM	LMPM	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status	—
LMPM_N_Data.cnf	LMPM	APMS APMR IP	CREP, LMPM_status	The placeholder IP is used for other protocol machines.
LMPM_N_Data.ind	LMPM	APMS APMR IP	CREP, DA, SA, VLANPrio, VLANID, N_SDU	The placeholder IP is used for other protocol machines.
LMPM_N_Data.req	APMS APMR IP	LMPM	CREP, DA, SA, VLANPrio, VLANID, N_SDU	The placeholder IP is used for other protocol machines.
LMPM_P_Data.cnf	LMPM	PTCP	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.ind	LMPM	PTCP	CREP, S_Port, Tstamp, DA, SA, N_SDU	—
LMPM_P_Data.req	DelayReq DelayRsp MPSM	LMPM	CREP, D_Port, Tstamp, DA, SA,	—

Primitive	Source	Destination	Associated parameters	Functions
			N_SDU	
MAC_RelayData.cnf	MAC_RELAY	LMPM	CREP, Status	—
MAC_RelayData.ind	MAC_RELAY	LMPM	CREP, DA, SA, LT, VLANPrio, VLANID, C_SDU, APDU_status	—
MAC_RelayData.req	LMPM	MAC_RELAY	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status	This function is provided by the IEEE 802.1D. It searches the filtering database and executes for each found destination port the M_UNITDATA.req.
MUX_Data.ind	MUX	LMPM	CREP, D_Port, Tstamp, Status	Due to the definition of a M_UNITDATA.req a indication is needed the deliver the transmit time stamp if needed.
QueueHandler.req	LMPM	QueueHandler	CREP, D_Port, Tstamp, N_SDU	For network management protocols or protocols which do not use the filtering database the LMPM accesses the MUX via the QueueHandler.req.

4.18.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by LMPM are described in the service definition and shown in Table 354.

Table 354 – Local primitives issued or received by LMPM

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

4.18.2.2 State transition diagram

The state transition diagram of the LMPM is shown in Figure 80.

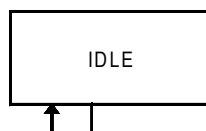


Figure 80 – State transition diagram of LMPM

States of the LMPM

IDLE

The LMPM is the interface between the bridges or the MAC and the local state machines.

4.18.2.3 State machine description

The LMPM forms the interface between MAC and LMPM-User for various data and management services. The services are all handled in the IDLE-State.

The service request will be validated first. A negative validation will result in a negative confirmation. Otherwise the service will be put into the appropriate priority service queue. Services executed by MAC will be moved from the request queues to the confirmation queues. All valid incoming services will be put into the indication queue.

4.18.2.4 LMPM state table

The LMPM state table is shown in Table 355.

Table 355 – LMPM state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	LMPM_P_Data.req () /CheckPDUType () == TimeStamped => QueueHandler.req () //NOTE After the dequeuing and transmitting a MUX_Data.ind () is generated from MUX and issued to the LMPM to convey the transmit Tstamp for the frame	IDLE
2	IDLE	MUX_Data.ind () => ignore LMPM_P_Data.cnf ()	IDLE
3	IDLE	LMPM_N_Data.req () /CheckPDUType () == other //NOTE Other includes RT_CLASS_UDP and RTA_CLASS_UDP => LMPM_N_Data.cnf () MAC_RELAY_Data.req ()	IDLE
4	IDLE	MAC_RELAY_Data.cnf () => ignore	IDLE
5	IDLE	LMPM_A_Data.req () /CheckPDUType () == RTA_CLASS_1 => LMPM_A_Data.cnf () MAC_RELAY_Data.req ()	IDLE
6	IDLE	LMPM_C_Data.req () /CheckPDUType () == RT_CLASS_1 => LMPM_C_Data.cnf () MAC_RELAY_Data.req ()	IDLE
7	IDLE	LMPM_C_Data.req () /CheckPDUType () == RT_CLASS_2 => LMPM_C_Data.cnf () MAC_RELAY_Data.req ()	IDLE
8	IDLE	LMPM_C_Data.req () /CheckPDUType () == RT_CLASS_3 => LMPM_C_Data.cnf () QueueHandler.req ()	IDLE

#	Current State	Event /Condition =>Action	Next State
9	IDLE	DMUX_P_Data.ind () /CheckPDUType () == TimeStamped => LMPM_P_Data.ind ()	IDLE
10	IDLE	RED_RELAY_Data.ind () /CheckPDUType () == RT_CLASS_3 => LMPM_C_Data.ind ()	IDLE
11	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == RT_CLASS_2 => LMPM_C_Data.ind ()	IDLE
12	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == RT_CLASS_1 => LMPM_C_Data.ind ()	IDLE
13	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == RTA_CLASS_1 => LMPM_A_Data.ind ()	IDLE
14	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == other => LMPM_N_Data.ind ()	IDLE

4.18.2.5 Functions, Macros, Timers and Variables

Table 356 contains the functions, macros, timers and variables used by the LMPM and their arguments and their descriptions.

Table 356 – Functions, Macros, Timers and Variables used by the LMPM

Name	Type	Function/Meaning
CheckPDUType	Function	This local function check the PDU type and returns the values - RT_CLASS_1 - RT_CLASS_2 - RT_CLASS_3 - RTA_CLASS_1 - Other - TimeStamped

5 Application layer protocol specification for decentralized periphery

5.1 FAL syntax description

5.1.1 DLPDU abstract syntax reference

The DLPDU abstract syntax from 4.1.1 shall be applied.

5.1.2 APDU abstract syntax

Table 357 defines the abstract syntax of the Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 357 shall represent the content of the DLSDU in Table 5.

Table 357 – IO APDU substitutions

Substitution name	Structure
RTA-SDU	AlarmNotification-PDU ^ AlarmAck-PDU
BlockHeader	BlockType, BlockLength, BlockVersionHigh, BlockVersionLow
AlarmNotification-PDU	BlockHeader, AlarmType, API ^a, SlotNumber ^a, SubslotNumber ^a, ModuleIdentNumber, SubmoduleIdentNumber, AlarmSpecifier, AlarmPayload ^a These fields addresses the submodule which may be source or sink of an alarm. They contains the source if the IO device issues an alarm and the sink if the IO controller issues an alarm. The BlockType field within the BlockHeader shall be set to AlarmNotification high or low according to Table 358.
AlarmAck-PDU	BlockHeader, AlarmType, API, SlotNumber, SubslotNumber, AlarmSpecifier, PNIOStatus The BlockType field within the BlockHeader shall be set to AlarmAck high or low according to Table 358. Special case “No Error”: PNIOStatus(=0x00, 0x00, 0x00, 0x00) Special case “Alarm Type Not Supported” or if a reserved Alarm Type is used: PNIOStatus(=0xDA, 0x81, 0x3C, 0x00) Special case “Wrong Submodule State”: PNIOStatus(=0xDA, 0x81, 0x3C, 0x01) Special case “IOCARSP: Backup – Alarm not executed”: PNIOStatus(=0xDA, 0x81, 0x3C, 0x02)
AlarmPayload	[MaintenanceItem], [DiagnosisItem] ^ [AlarmItem] ^ [Upload&RetrievalItem] ^ [iParameterItem] See Table 361
MaintenanceItem	UserStructureIdentifier, BlockHeader, Padding, Padding, MaintenanceStatus
Upload&RetrievalItem	UserStructureIdentifier, BlockHeader, Padding, Padding, (URRecordIndex, URRecordLength)* Special case “Retrieve all stored records”: UserStructureIdentifier, BlockHeader(=0x0F04), Padding, Padding, URRecordIndex(=0), URRecordLength(=0)
iParameterItem	UserStructureIdentifier, BlockHeader, Padding, Padding, iPar_Req_Header, Max_Segm_Size, Transfer_Index, Total_iPar_Size

Substitution name	Structure
DiagnosisItem	UserStructureIdentifier, ChannelDiagnosisData* ^ ExtChannelDiagnosisData* ^ QualifiedChannelDiagnosisData* ^ DiagnosisData* Special case ChannelDiagnosisData: UserStructureIdentifier(=0x8000), ChannelDiagnosisData* Special case ExtChannelDiagnosisData: UserStructureIdentifier(=0x8002), ExtChannelDiagnosisData* Special case QualifiedChannelDiagnosisData: UserStructureIdentifier(=0x8003), QualifiedChannelDiagnosisData* Special case DiagnosisData with ManufacturerSpecificDiagnosis: UserStructureIdentifier(=0x0000 – 0x7FFF), (BlockHeader, API, ManufacturerSpecific-Diagnosis)*
AlarmItem	UserStructureIdentifier, Data*
PROFINETIO-ServiceReqPDU	IODConnectReq ^ IODWriteReq ^ IODWriteMultipleReq ^ IODReadReq ^ IODControlReq ^ IODReleaseReq ^ IOXControlReq
PROFINETIO-ServiceResPDU	IODConnectRes ^ IODWriteRes ^ IODWriteMultipleRes ^ IODReadRes ^ IODControlRes ^ IODReleaseRes ^ IOXControlRes
IODConnectReq with StartupMode:=1	ARBlockReq, {[IOCRBlockReq*], [AlarmCRBlockReq], [ExpectedSubmoduleBlockReq*]^b, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq], [IRInfoBlock], [SRInfoBlock], [ARVendorBlockReq*], [ARFSUBlock]} Special case: "IOCARSingle" or "IOSAR": ARBlockReq, {[IOCRBlockReq*, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq], [IRInfoBlock], [ARVendorBlockReq*], [ARFSUBlock]} } Special case: "IOSAR with ARProperties.DeviceAccess:=1": ARBlockReq^c Special case: "IOCARSR": ARBlockReq, {[IOCRBlockReq*, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq], [IRInfoBlock], [SRInfoBlock], [ARVendorBlockReq*], [ARFSUBlock]} } ^a The field MCRBlockReq shall only be present if at least one IOCRBlockReq contains the value MULTICAST_CONSUMER_CR as an IO CR Type. ^b The field ExpectedSubmoduleBlockReq shall only contain submodules which are referred in at least one IOCRBlockReq. ^c The field CMInitiatorActivityTimeoutFactor specifies the watchdog time.

Substitution name	Structure
IODConnectRes with StartupMode:=1	ARBlockRes, {[IOCRBlockRes*], [AlarmCRBlockRes], [ModuleDiffBlock], [ARRPCBlockRes]^a, ARServerBlockRes, [ARVendorBlockRes*]^b} Special case: "IOCARSingle" or "IOSAR": ARBlockRes, {[IOCRBlockRes*, AlarmCRBlockRes, [ModuleDiffBlock], [ARRPCBlockRes]^a, ARServerBlockRes, [ARVendorBlockRes*]^b} Special case: "IOSAR with ARProperties.DeviceAccess=1": ARBlockRes, ARServerBlockRes ^a The field ARRPCBlockRes shall only be present if IODConnectReq contains an ARRPCBlockReq. ^b The number of ARVendorBlockRes entries shall be equal to the number of ARVendorBlockReq entries in the IODConnectReq.
IODConnectReq with StartupMode:=0	ARBlockReq, {[IOCRBlockReq*], [AlarmCRBlockReq], [ExpectedSubmoduleBlockReq*]^b, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq]} Special case: "IOCARSingle" or "IOSAR": ARBlockReq, {[IOCRBlockReq*, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq]} Special case: "IOSAR with ARProperties.DeviceAccess:=1": ARBlockReq^c Special case: "IOCARSingle using RT_CLASS_3": ARBlockReq, {[IOCRBlockReq*^d, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq]} ^a The field MCRBlockReq shall only be present if at least one IOCRBlockReq contains the value MULTICAST_CONSUMER_CR as an IO CR Type. ^b The field ExpectedSubmoduleBlockReq shall only contain submodules which are referred in at least one IOCRBlockReq. ^c The field CMInitiatorActivityTimeoutFactor specifies the watchdog time. ^d To establish an "IOCARSingle using RT_CLASS_3" with RT_CLASS_3 input and output communication relations, one input and one output communication relation, shall exist.
IODConnectRes with StartupMode:=0	ARBlockRes, {[IOCRBlockRes*], [AlarmCRBlockRes], [ModuleDiffBlock], [ARRPCBlockRes]^a} Special case: "IOCARSingle" or "IOSAR": ARBlockRes, {[IOCRBlockRes*, AlarmCRBlockRes, [ModuleDiffBlock], [ARRPCBlockRes]^a} Special case: "IOSAR with ARProperties.DeviceAccess=1": ARBlockRes ^a The field ARRPCBlockRes shall only be present if IODConnectReq contains an ARRPCBlockReq.

Substitution name	Structure
IODWriteMultipleReq	IODWriteReqHeader ^a , (IODWriteReqHeader, RecordDataWrite, [Padding*] ^b)* ^a The value of the parameter index shall be 0xE040. ^b The number of padding octets (value=0) shall be 0,1,2,3 to have 32 bit alignment to the next IODWriteReqHeader.
IODWriteReq	IODWriteReqHeader, RecordDataWrite
IODWriteReqHeader	BlockHeader, SeqNumber, ARUUUID, API, SlotNumber, SubslotNumber, Padding* ^a , Index, RecordDataLength, RWPadding* ^b ^a The number of padding octets (value=0) in the write request is 2. ^b The number of padding octets (value=0) in the write request is 24.
IODWriteRes	IODWriteResHeader ^a ^a The value of the parameter PNIOStatus shall be (0,0,0,0) or a duplicate of the PNIOStatus from the NDRDataResponse.
IODWriteMultipleRes	IODWriteResHeader ^a ^b , (IODWriteResHeader)* ^c ^a The value of the parameter index shall be 0xE040. ^b The value of the parameter PNIOStatus shall be (0,0,0,0) or a duplicate of the PNIOStatus from the NDRDataResponse. ^c The value of the parameter PNIOStatus contains the result of the particular write.
IODWriteResHeader	BlockHeader, SeqNumber, ARUUUID, API, SlotNumber, SubslotNumber, Padding* ^a , Index, RecordDataLength, AdditionalValue1, AdditionalValue2, PNIOStatus, RWPadding* ^b ^a The number of padding octets (value=0) in the write response is 2. ^b The number of padding octets (value=0) in the write response is 16.
IODReadReq	IODReadReqHeader, RecordDataReadQuery
IODReadReqHeader	BlockHeader, SeqNumber, ARUUUID, API, SlotNumber, SubslotNumber, Padding* ^a , Index, RecordDataLength, [TargetARUUUID] ^b , RWPadding* ^c ^a The number of padding octets (value=0) in the read request is 2. ^b The optional field TargetARUUUID shall only be used in conjunction with the value 0 of ARUUUID (Implicit AR). ^c The number of padding octets (value=0) in the read request is 24 or 8.
RecordDataReadQuery	UploadBLOBQuery ^a (BlockHeader, Data* ^a) ^a NULL ^a The usage, structure and values shall be defined for a) Manufacturer specific index range by the manufacturer b) Profile specific index ranges by the profiles c) All other index ranges by this standard
IODReadRes	IODReadResHeader, RecordDataRead
IODReadResHeader	BlockHeader, SeqNumber, ARUUUID, API, SlotNumber, SubslotNumber, Padding* ^a , Index, RecordDataLength, AdditionalValue1, AdditionalValue2, RWPadding* ^b ^a The number of padding octets (value=0) in the read res is 2. ^b The number of padding octets (value=0) in the read res is 20.

Substitution name	Structure
IODControlReq	ControlBlockConnect^a ^ (ControlBlockConnect, SubmoduleListBlock)^b ^ ControlBlockPlug^c ^a Shall be used in the context of a connect or a PrmBegin/PrmEnd sequence for ControlCommand.PrmEnd ^b Shall be used in the context of a PrmBegin/PrmEnd sequence for ControlCommand.PrmBegin with SubmoduleListBlock ^c Shall be used in the context of a plug/release for ControlCommand.PrmEnd
IODControlRes	ControlBlockConnect^a ^ (ControlBlockConnect, [ModuleDiffBlock])^b ^ ControlBlockPlug^c ^ NULL^d ^a Shall be used in the context of a connect or a PrmBegin/PrmEnd sequence for ControlCommand.PrmEnd ^b Shall be used in the context of a PrmBegin/PrmEnd sequence for ControlCommand.PrmBegin if the checking of the SubmoduleListBlock leads to a ModuleDiffBlock ^c Shall be used in the context of a plug/release for ControlCommand.PrmEnd ^d In case of a negative response NULL shall be transmitted.
IODReleaseReq	ReleaseBlock
IODReleaseRes	ReleaseBlock ^ NULL^a ^a In case of a negative response NULL shall be transmitted.
IOXControlReq with StartupMode:=1	(ControlBlockConnect, [ModuleDiffBlock])^a ^ (ControlBlockPlug, [ModuleDiffBlock]))^b ^a Shall be used in the context of a connect or a PrmBegin/PrmEnd sequence for ControlCommand.ApplicationReady with ModuleDiffBlock if needed. ^b Shall be used in the context of a plug/release for ControlCommand.ApplicationReady with ModuleDiffBlock if needed. The field ModuleDiffBlock shall only be present if it contains entries. An empty ModuleDiffBlock shall be omitted.
IOXControlRes with StartupMode:=1	ControlBlockConnect^a ^ ControlBlockPlug^b ^ NULL^c ^a Shall be used in the context of a connect or a PrmBegin/PrmEnd sequence for ControlCommand.ApplicationReady ^b Shall be used in the context of a plug/release for ControlCommand.ApplicationReady ^c In case of a negative response NULL shall be transmitted.
IOXControlReq with StartupMode:=0	(ControlBlockConnect, [ModuleDiffBlock])^a ^ (ControlBlockPlug, [ModuleDiffBlock]))^b ^ ControlBlockRFC^c ^ ControlBlockRTC^d ^a Shall be used in the context of a connect for ControlCommand.ApplicationReady with ModuleDiffBlock if needed. ^b Shall be used in the context of a plug/release for ControlCommand.ApplicationReady with ModuleDiffBlock if needed. ^c Shall be used for ControlCommand.ReadyForCompanion ^d Shall be used for ControlCommand.ReadyForRT_CLASS_3 The field ModuleDiffBlock shall only be present if it contains entries. An empty ModuleDiffBlock shall be omitted.

Substitution name	Structure
IOXControlRes with StartupMode:=0	ControlBlockConnect ^a ^ ControlBlockPlug ^b ^ ControlBlockRFC ^c ^ ControlBlockRTC ^d ^ NULL ^e ^a Shall be used in the context of a connect for ControlCommand.ApplicationReady ^b Shall be used in the context of a plug/release for ControlCommand.ApplicationReady ^c Shall be used for ControlCommand.ReadyForCompanion ^d Shall be used for ControlCommand.ReadyForRT_CLASS_3 ^e In case of a negative response NULL shall be transmitted.
ARBlockReq	BlockHeader, ARType, ARUUID, SessionKey, CMInitiatorMacAdd, CMInitiatorObjectUUID, ARProperties, CMInitiatorActivityTimeoutFactor, InitiatorUDPRTPort ^a, StationNameLength, CMInitiatorStationName ^b ^a If no RT_CLASS_UDP CR is used, the value for InitiatorUDPRTPort shall be set to 0x8892. ^b This coding leads to a misalignment if the length of CMInitiatorStationName is uneven. An IO device should support a [Padding*] behind to ensure Unsigned32 alignment. The BlockType field within the BlockHeader shall be set to ARBlockReq according to Table 358.
ARBlockRes	BlockHeader, ARType, ARUUID, SessionKey, CMResponderMacAdd, ResponderUDPRTPort ^a If no RT_CLASS_UDP CR is used, the value for ResponderUDPRTPort shall be set to 0x8892. The BlockType field within the BlockHeader shall be set to ARBlockRes according to Table 358.
IOCRBlockReq	BlockHeader, IOCRTType, IOCRReference, LT, IOCRProperties, DataLength, FrameID, SendClockFactor, ReductionRatio, Phase, Sequence, FrameSendOffset, DataHoldFactor ^a, DataHoldFactor, IOCRTagHeader, IOCRMulticastMACAdd, NumberOfAPIs, (API, NumberOfIODataObjects, (SlotNumber, SubslotNumber, IODataObjectFrameOffset)*, NumberOfIQCS, (SlotNumber, SubslotNumber, IQCSFrameOffset)*)* ^a This field exists due to legacy reasons and was formerly called WatchDogFactor. The BlockType field within the BlockHeader shall be set to IOCRBlockReq according to Table 358. Special Case IOCRTType == OUTPUT_CR not using RT_CLASS_3 The field FrameID shall not be evaluated. Special Case IOCRTType == OUTPUT_CR using RT_CLASS_3 The field FrameID shall contain the FrameID used in the corresponding PDIRData. Special Case IOCRTType == MULTICAST_CONSUMER_CR The field FrameID shall contain a FrameID from the defined multicast range. The configuration tool shall guarantee that this FrameID is unique within the project context.
MCRBlockReq	BlockHeader, IOCRReference, AddressResolutionProperties, MCITimeoutFactor, StationNameLength, ProviderStationName, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.

Substitution name	Structure
IOCRBlockRes	BlockHeader, IOCRTType, IOCReference, FrameID The BlockType field within the BlockHeader shall be set to IOCRBlockRes according to Table 358.
ExpectedSubmodule-BlockReq	BlockHeader, NumberOfAPIs, (API, SlotNumber^a, ModuleIdentNumber, ModuleProperties, NumberOfSubmodules, (SubslotNumber, SubmoduleIdentNumber, SubmoduleProperties^b, (DataDescription, SubmoduleDataLength, LengthIOPS, LengthIOCS)*))* ^a This parameter is constant for one ExpectedSubmoduleBlockReq because each contains the data for only one slot. ^b The field SubmoduleProperties.Type determines the number of subsequent data description blocks. The BlockType field within the BlockHeader shall be set to ExpectedSubmoduleBlockReq according to Table 358.
ModuleDiffBlock	BlockHeader, NumberOfAPIs, (API, NumberOfModules, (SlotNumber, ModuleIdentNumber^a, ModuleState^b, NumberOfSubmodules, [(SubslotNumber, SubmoduleIdentNumber^a, SubmoduleState)*]))* ^a Real identification data ^b If ModuleState=NO_MODULE then NumberOfSubmodules shall be zero. The subsequent part shall be omitted. For all other ModuleState only the SubmoduleState shall be used to decide whether the submodule shall be parameterized. The BlockType field within the BlockHeader shall be set to ModuleDiffBlock according to Table 358.
AlarmCRBlockReq	BlockHeader, AlarmCRTType, LT, AlarmCRProperties, RTATimeoutFactor, RTARetries, LocalAlarmReference, MaxAlarmDataLength, AlarmCRTagHeaderHigh, AlarmCRTagHeaderLow The BlockType field within the BlockHeader shall be set to AlarmCRBlockReq according to Table 358.
AlarmCRBlockRes	BlockHeader, AlarmCRTType, LocalAlarmReference, MaxAlarmDataLength The BlockType field within the BlockHeader shall be set to AlarmCRBlockRes according to Table 358.
PrmServerBlock	BlockHeader, ParameterServerObjectUUID, ParameterServerProperties, CMInitiatorActivityTimeoutFactor, StationNameLength, ParameterServerStationName^a ^a An IO device should support a [Padding*] behind to ensure Unsigned32 alignment. The BlockType field within the BlockHeader shall be set to PrmServerBlock according to Table 358.
ARRPCBlockReq	BlockHeader, InitiatorRPCServerPort The BlockType field within the BlockHeader shall be set to ARRPCBlockReq according to Table 358.
ARRPCBlockRes	BlockHeader, ResponderRPCServerPort The BlockType field within the BlockHeader shall be set to ARRPCBlockRes according to Table 358.

Substitution name	Structure
ARServerBlockRes	BlockHeader, StationNameLength, CMResponderStationName, [Padding* ^a] ^a The number of Padding octets shall ensure Unsigned32 alignment. The BlockType field within the BlockHeader shall be set to ARServerBlockRes according to Table 358.
IRInfoBlock	BlockHeader, Padding* ^a, IRDataUUID, Padding*^a, NumberOfIOCRs ^b, [(IOCRReference, SubframeOffset, SubframeData)*] ^a The number of Padding octets shall be 2. ^b This field shall be coded as zero if no DFP is used. The BlockType field within the BlockHeader shall be set to IRInfoBlock according to Table 358.
SRInfoBlock	BlockHeader, RedundancyDataHoldFactor, SRProperties The BlockType field within the BlockHeader shall be set to SRInfoBlock according to Table 358.
ARFSUBlock	ARFSUDDataAdjust The BlockType field within the BlockHeader shall be set to ARFSUBlock according to Table 358.
ARVendorBlockReq	BlockHeader, APStructureIdentifier, API, [Data*], [Padding* ^a] ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. The BlockType field within the BlockHeader shall be set to ARVendorBlockReq according to Table 358. Special case "Extended identification rules" BlockHeader, APStructureIdentifier (:=0x8000), API (:=0), ExpectedExtendedIdentificationInfo, [Padding* ^a]
ARVendorBlockRes	BlockHeader, APStructureIdentifier, API, [Data*], [Padding* ^a] ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. The BlockType field within the BlockHeader shall be set to ARVendorBlockRes according to Table 358. Special case "Extended identification rules" BlockHeader, APStructureIdentifier (:=0x8000), API (:=0), RealExtendedIdentificationInfo, [Padding* ^a]
ExpectedExtended-IdentificationInfo	ExtendedIdentificationInfo
RealExtended-IdentificationInfo	ExtendedIdentificationInfo
ExtendedIdentification-Info	ExtendedIdentificationUUID, ExtendedIdentificationVersionHigh, ExtendedIdentificationVersionLow

Substitution name	Structure
SubframeBlock	BlockHeader, FrameID^a, SFIOCRProperties, SubframeData*^b ^a For MRPD only the SubframeBlock for the FrameID with the least significant bit set to zero shall be created. ^b The ordering of SubframeData octets shall be used for the non redundant path The BlockType field within the BlockHeader shall be set to SubframeBlock according to Table 358.
PDIRSubframeData	BlockHeader, NumberOfSubframeBlocks, SubframeBlock* The BlockType field within the BlockHeader shall be set to PDIRSubframeBlock according to Table 358.
ControlBlockConnect	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a The number of Padding octets shall be 2. The BlockType field within the BlockHeader shall be set to IOBlockReq, IOBlockRes, IOXBlockReq and IOXBlockRes according to Table 358.
ControlBlockPlug	BlockHeader, Padding*^a, ARUID, SessionKey, AlarmSequenceNumber, ControlCommand, ControlBlockProperties ^a The number of Padding octets shall be 2. The BlockType field within the BlockHeader shall be set to IOXBlockReq, IOXBlockRes according to Table 358. The field AlarmSequenceNumber shall contain the value of the sub-field AlarmSpecifier.SequenceNumber of the corresponding AlarmNotification-PDU.
ControlBlockRFC	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a The number of Padding octets shall be 2. The BlockType field within the BlockHeader shall be set to ReadyForCompanionBlock according to Table 358.
ControlBlockRTC	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a The number of Padding octets shall be 2. The BlockType field within the BlockHeader shall be set to ReadyForRTCLASS3Block according to Table 358.
ReleaseBlock	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a The number of Padding octets shall be 2. The BlockType field within the BlockHeader shall be set to ReleaseBlock according to Table 358.

Substitution name	Structure
SubmoduleListBlock	BlockHeader, NumberOfEntries, (API, SlotNumber, SubslotNumber)* The BlockType field within the BlockHeader shall be set to SubmoduleListBlock according to Table 358.
RecordDataRead	DiagnosisData* ^ ExpectedIdentificationData* ^ RealIdentificationData* ^ SubstituteValue ^ RecordInputDataObjectElement ^ RecordOutputDataObjectElement ^ Data* ^ ARData ^ IMData ^ LogBookData ^ ModuleDiffBlock ^ APIData ^ PDPortDataAdjust* ^ PDPortDataCheck* ^ PDPortDataReal* ^ PDIRData ^ PDSyncData* ^ PdevData ^ IsochronousModeData* ^ PDInterfaceAdjust ^ PDInterfaceMrpDataAdjust* ^ PDInterfaceMrpDataCheck* ^ PDInterfaceMrpDataReal* ^ PDPortMrpDataAdjust ^ PDPortMrpDataReal ^ PDPortFODDataReal ^ PDPortFODDataAdjust ^ PDPortFODDataCheck ^ PDRealData* ^ PDExpectedData* ^ PDNCDataCheck ^ PDPortStatistic ^ I&M0FilterData ^ ARFSUDDataAdjust ^ PDInterfaceFSUDDataAdjust ^ PDInterfaceDataReal ^ AutoConfiguration ^ PDIRSubframeData ^ NestedDiagnosisInfo ^ UploadBLOB ^ NULL ^a ^a NULL shall be used if a requested well-known data record is empty (e.g. Diagnosis, ModuleDiffBlock, ARData, ...).
RecordDataWrite	SubstituteValue ^ Data* ^ IMDataWrite ^ PDPortDataAdjust ^ PDPortDataCheck ^ PDIRData ^ PDSyncData ^ IsochronousModeData ^ PDInterfaceMrpDataAdjust ^ PDInterfaceMrpDataCheck ^ PDPortMrpDataAdjust ^ PDPortFODDataAdjust ^ PDPortFODDataCheck ^ PDNCDataCheck ^ ARFSUDDataAdjust ^ PDInterfaceFSUDDataAdjust ^ PDIRSubframeData ^ PDInterfaceAdjust ^ CombinedObjectContainer
DiagnosisData with BlockVersionLow=0	BlockHeader, ChannelDiagnosis ^ ManufacturerSpecificDiagnosis ^ ExtChannelDiagnosis BlockVersionLow=0 shall be supported by IO controller and IO supervisor. It shall not be generated by IO device.
DiagnosisData with BlockVersionLow=1	BlockHeader, API, ChannelDiagnosis ^ ManufacturerSpecificDiagnosis ^ ExtChannelDiagnosis ^ QualifiedChannelDiagnosis BlockVersionLow=1 shall be generated by IO device and support by IO controller and IO supervisor
ChannelDiagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^a, UserStructureIdentifier(0x8000), ChannelDiagnosisData* ^a The field ChannelProperties.Type, the field ChannelProperties.Direction, the field ChannelProperties.Maintenance shall be set to zero. The field ChannelProperties.Specifier shall be set to appear if at least one ChannelProperties.Specifier in the ChannelDiagnosisData is set to appear in conjunction with ChannelProperties.Maintenance(=diagnosis). Else, the field ChannelProperties.Specifier shall be set to disappear.
ChannelDiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType
Manufacturer-SpecificDiagnosis	SlotNumber, SubslotNumber, ChannelNumber, ChannelProperties, UserStructureIdentifier, Data*
ExtChannelDiagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^a, UserStructureIdentifier(0x8002), ExtChannelDiagnosisData* ^a The field ChannelProperties.Type, the field ChannelProperties.Direction, the field ChannelProperties.Maintenance shall be set to zero. The field ChannelProperties.Specifier shall be set to appear if at least one ChannelProperties.Specifier in the ExtChannelDiagnosisData is set to appear in conjunction with ChannelProperties.Maintenance(=diagnosis). Else, the field ChannelProperties.Specifier shall be set to disappear.
ExtChannel-DiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType, ExtChannelAddValue

Substitution name	Structure
QualifiedChannel-Diagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^a , UserStructureIdentifier(0x8003), QualifiedChannelDiagnosisData [*] ^a The field ChannelProperties.Type, the field ChannelProperties.Direction, the field ChannelProperties.Maintenance shall be set to zero. The field ChannelProperties.Specifier shall be set to appear if at least one ChannelProperties.Specifier in the QualifiedChannelDiagnosisData is set to appear in conjunction with ChannelProperties.Maintenance(=diagnosis). Else, the field ChannelProperties.Specifier shall be set to disappear.
QualifiedChannel-DiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType, ExtChannelAddValue, QualifiedChannelQualifier
ExpectedIdentification-Data with BlockVersionLow = 0 Legacy	BlockHeader, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber)*)*
ExpectedIdentification-Data with BlockVersionLow = 1	BlockHeader, NumberOfAPIs, (API, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber)*))*
RealIdentificationData with BlockVersionLow = 0	BlockHeader, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber)*)*
RealIdentificationData with BlockVersionLow = 1	BlockHeader, NumberOfAPIs, (API, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber)*))*
PDIRData with BlockVersionLow = 1	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, PDIRGlobalData, PDIRFrameData ^{a b} , PDIRBeginEndData ^a This block may contain information about receiving, transmitting and forwarding, or receiving and transmitting only. ^b If this block exists, the DFP frames shall be integrated.
PDSyncData with BlockVersionLow = 2	BlockHeader, Padding, Padding, PTCP_SubdomainUUID, ReservedIntervalBegin, ReservedIntervalEnd, PLLWindow, SyncSendFactor, SendClockFactor, PTCPTimeoutFactor, PTCPTakeoverTimeoutFactor, PTCPMasterStartupTime, SyncProperties, PTCP_MasterPriority1, PTCP_MasterPriority2, PTCPLengthSubdomainName, PTCPSubdomainName, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PDTimeData	BlockHeader, TimeMasterPriority1, TimeMasterPriority2, TimePLLWindow, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PdevData	BlockHeader, Padding, Padding, [PDIRData], [PDSyncData], [PDIRSubframeData], [PDTimeData]
PDRealData	MultipleBlockHeader, { [PDPortDataReal] ^b , [PDInterfaceMrpDataReal], [PDPortMrpDataReal], [PDPortFODDataReal] ^a , [PDInterfaceDataReal], [PDPortStatistic] } ^c ^a There shall be no FiberOpticManufacturerSpecific information ^b The fields SlotNumber and SubslotNumber shall be ignored ^c Each submodule's data (e.g. interface or port) need its own MultipleBlockHeader

Substitution name	Structure
PDExpectedData	MultipleBlockHeader, { [PDPortDataCheck]^a, [PDPortDataAdjust]^a, [PDInterfaceMrpDataAdjust], [PDInterfaceMrpDataCheck], [PDPortMrpDataAdjust], [PDPortFODataAdjust], [PDPortFODataCheck], [PDNCDataCheck], [PDInterfaceFSUDataAdjust], [PDInterfaceAdjust] }^b ^a The fields SlotNumber and SubslotNumber shall be ignored ^b Each submodule's data (e.g. interface or port) need its own MultipleBlockHeader
PDPortDataReal	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, LengthOwnPortName, OwnPortName, NumberOfPeers, [Padding*]^a, [LengthPeerPortName, PeerPortName, LengthPeerStationName, PeerStationName, [Padding*]^a, LineDelay, PeerMACAddress^b, [Padding*]^a]*, MAUType^c, [Padding*]^a, Reserved^d, RTClass3_PortStatus, MulticastBoundary, LinkState, [Padding*]^a, MediaType ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. ^b This field contains the interface MAC address of the peer ^c See Table 552 ^d The number of reserved octets shall be 2.
PDInterfaceDataReal	BlockHeader, LengthOwnStationName, OwnStationName, [Padding*]^a, MACAddressValue^b, [Padding*]^a, IPPParameterValue, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. ^b This field contains the interface MAC address
TxPortGroup	NumberOfTxPortGroups, TxPortGroupArray
SubstituteValue	BlockHeader, SubstitutionMode, SubstituteDataItem
RecordInputData-ObjectElement	BlockHeader, LengthIOCS, IOCS, LengthIOPS, IOPS, LengthData, Data Special case: Response for an output submodule PNIOStatus(=0xDE, 0x80, 0xB0, 0x00)
RecordOutputData-ObjectElement	BlockHeader^c, SubstituteActiveFlag, LengthIOCS, LengthIOPS, LengthData, DataItem^a, SubstituteValue^b Special case: Response for an input submodule PNIOStatus(=0xDE, 0x80, 0xB0, 0x00) ^a For this record DataItem shall be coded as IOCS, Data, IOPS ^b The IOPS of the SubstituteDataItem inherited by the SubstituteValue contains the valid information ^c The BlockLength shall include the SubstituteValue

Substitution name	Structure
ARData with BlockVersionLow = 0 Legacy May only be used with StartupMode:=0	BlockHeader, NumberOfARs, (ARUUID, ARTYPE, ARProperties, CMInitiatorObjectUUID, StationNameLength, CMInitiatorStationName, NumberOfIOCRs, (IOCRType, IOCRProperties, FrameID, APDU_Status^a, InitiatorUDPRTPort, ResponderUDPRTPort)*, AlarmCRTYPE, LocalAlarmReference, RemoteAlarmReference, ParameterServerObjectUUID^b, StationNameLength^c, [ParameterServerStationName], NumberOfAPIs, API*)* Special case: "IOSAR with ARProperties.DeviceAccess=1": NumberOfIOCRs := 0 AlarmCRTYPE := 0 NumberOfAPIs := 0 ^a The APDU_Status.CycleCounter and APDU_Status.TransferStatus can be zero ^b The ParameterServerObjectUUID shall be NIL if not used ^c The StationNameLength shall be 0 if not used. In this case the field ParameterServerStationName shall be omitted
ARData with BlockVersionLow = 1	BlockHeader, NumberOfARs, (ARUUID, CMInitiatorObjectUUID, ParameterServerObjectUUID^b, ARProperties, ARTYPE, AlarmCRTYPE, LocalAlarmReference, RemoteAlarmReference, InitiatorUDPRTPort, ResponderUDPRTPort, StationNameLength, CMInitiatorStationName, [Padding*]^d, StationNameLength^c, [ParameterServerStationName], [Padding*]^d, NumberOfIOCRs, [Padding*]^d, (IOCRProperties, IOCRType, FrameID, APDU_Status^a)*, NumberOfAPIs, [Padding*]^d, API*, ARDataInfo)* Special case: "IOSAR with ARProperties.DeviceAccess=1": NumberOfIOCRs := 0 AlarmCRTYPE := 0 NumberOfAPIs := 0 ^a The APDU_Status.CycleCounter and APDU_Status.TransferStatus can be zero ^b The ParameterServerObjectUUID shall be NIL if not used ^c The StationNameLength shall be 0 if not used. In this case the field ParameterServerStationName shall be omitted ^d The number of padding octets shall be adapted to make the following field Unsigned32 aligned.
ARDataInfo	NumberOfEntries, [Padding*]^a, { [IRInfoBlock], [SRInfoBlock], [ARVendorBlockReq *], [ARVendorBlockRes *], [ARFSUBBlock], [SRLData]^b } ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. ^b The SRLData block shall be used if a redundant network access point exists.
SRLData	BlockHeader, RedundancyInfo, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
APIData	BlockHeader, NumberOfAPIs, API*
LogBookData	BlockHeader, ActualLocalTimeStamp, NumberOfLogEntries, (LocalTimeStamp, ARUUID, PNIOStatus, EntryDetail)*
CMInitiatorObjectUUID	PROFINETIOConstantValue, InstanceHigh, InstanceLow, DeviceIDentNumber
ParameterServer-ObjectUUID	PROFINETIOConstantValue, InstanceHigh, InstanceLow, DeviceIDentNumber
DeviceIDentNumber	DeviceIDHigh, DeviceIDLow, VendorIDHigh, VendorIDLow
IMDataWrite	[I&M1] ^ [I&M2] ^ [I&M3] ^ [I&M4]
IMData	I&M0 ^ [I&M1] ^ [I&M2] ^ [I&M3] ^ [I&M4]

Substitution name	Structure
I&M0	BlockHeader, VendorIDHigh, VendorIDLow, OrderID, IM_Serial_Number, IM_Hardware_Revision, IM_Software_Revision, IM_Revision_Counter, IM_Profile_ID, IM_Profile_Specific_Type, IM_Version, IM_Supported
I&M1	BlockHeader, IM_Tag_Function, IM_Tag_Location
I&M2	BlockHeader, IM_Date
I&M3	BlockHeader, IM_Descriptor
I&M4	BlockHeader, IM_Signature
I&M0FilterData	{I&M0FilterDataSubmodule ^a, [I&M0FilterDataModule] ^b, I&M0FilterDataDevice ^c} ^a Shall contain all submodules with discrete IMData ^b Shall, if exists, contain only the module reference ^c Shall contain the device reference (e.g. the interface submodule) which shall support writing of I&M1, I&M2, I&M3, and I&M4
I&M0FilterDataInfo	NumberOfAPIs, (API, NumberOfModules, (SlotNumber, ModuleIdentNumber, NumberOfSubmodules, (SubslotNumber, SubmoduleIdentNumber)*))*
I&M0FilterData-Submodule	BlockHeader, I&M0FilterDataInfo
I&M0FilterDataModule	BlockHeader, I&M0FilterDataInfo
I&M0FilterDataDevice	BlockHeader, I&M0FilterDataInfo
IM_Version	IM_Version_Major, IM_Version_Minor
IM_Software_Revision	SWRevisionPrefix, IM_SWRevision_Functional_Enhancement, IM_SWRevision_Bug_Fix, IM_SWRevision_Internal_Change
MultipleBlockHeader	BlockHeader, Padding, Padding, API, SlotNumber, SubslotNumber
PDIRGlobalData with BlockVersionLow = 1	BlockHeader, Padding, Padding, IRDataUUID, MaxBridgeDelay, NumberOfPorts, (MaxPortTxDelay, MaxPortRxDelay)*
PDIRGlobalData with BlockVersionLow = 2	BlockHeader, Padding, Padding, IRDataUUID, MaxBridgeDelay, NumberOfPorts, (MaxPortTxDelay, MaxPortRxDelay, MaxLineRxDelay, YellowTime)*
PDIRBeginEndData	BlockHeader, Padding, Padding, RedGuard, NumberOfPorts, (NumberOfAssignments, (TXBeginEndAssignment, RXBeginEndAssignment)*, NumberOfPhases, (TXPhaseAssignment, RXPhaseAssignment))*
RedGuard	StartOfRedFrameID, EndOfRedFrameID
TXBeginEndAssignment	RedOrangePeriodBegin, OrangePeriodBegin ^a, GreenPeriodBegin ^a Shall be set to GreenPeriodBegin
RXBeginEndAssignment	RedOrangePeriodBegin, OrangePeriodBegin ^a, GreenPeriodBegin ^b ^a Shall be set to GreenPeriodBegin ^b GreenPeriodBegin shall be set to the end of the red period.
TXPhaseAssignment	PhaseAssignment
RXPhaseAssignment	PhaseAssignment
PDIRFrameData with BlockVersionLow = 0	BlockHeader, Padding, Padding, (FrameSendOffset, DataLength, ReductionRatio, Phase, FrameID, EtherType, RxPort, FrameDetails, TxPortGroup, [Padding*] ^a)* ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.

Substitution name	Structure
PDIRFrameData with BlockVersionLow = 1	BlockHeader, Padding, Padding, FrameDataProperties, [(FrameSendOffset, DataLength, ReductionRatio, Phase, FrameID^{a b c}, EtherType, RxPort, FrameDetails, TxPortGroup, [Padding*]^d)*] ^a This block shall contain all FrameIDs which are locally received (node is sink) or locally injected (node is source) into the network. It may contain the FrameIDs which are only forwarded by the node. ^b With DFP: This block shall contain two entries for the same FrameID if this node contains a DFP_INBOUND or a DFP_OUTBOUND state machine which is a receiver of this frame. One entry contain the receive path and one the transmit path. If the transmit path doesn't exist (outbound frame is fully consumed by this node) the corresponding entry shall be skipped. ^c With MRPD: A consumer shall exist two times (FrameID and FrameID+1) and a provider shall exist two times (FrameID and FrameID+1) in case of IOCR and may exist more than two times in case of MCR. ^d The number of padding octets shall be adapted to make the block Unsigned32 aligned.
IsochronousModeData	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, ControllerApplicationCycleFactor, TimeDataCycle, TimeIOInput, TimeIOOutput, TimeIOInputValid, TimeIOOutputValid
PDPortDataAdjust	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, { [AdjustDomainBoundary], [AdjustMulticastBoundary], [AdjustMAUType ^ AdjustLinkState], [AdjustPeerToPeerBoundary], [AdjustDCPBoundary], [AdjustPreambleLength] }
PDPortDataCheck	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, { [CheckPeers], [CheckLineDelay], [CheckMAUType], [CheckLinkState], [CheckSyncDifference], [CheckMAUTypeDifference] }
AdjustDomainBoundary with BlockVersionLow = 1	BlockHeader, Padding, Padding, DomainBoundaryIngress, DomainBoundaryEgress, AdjustProperties, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
AdjustMulticast-Boundary	BlockHeader, Padding, Padding, MulticastBoundary, AdjustProperties, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
AdjustMAUType	BlockHeader, Padding, Padding, MAUType, AdjustProperties
AdjustLinkState	BlockHeader, Padding, Padding, LinkState, AdjustProperties
AdjustPeerToPeer-Boundary	BlockHeader, Padding, Padding, PeerToPeerBoundary, AdjustProperties, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
AdjustDCPBoundary	BlockHeader, Padding, Padding, DCPBoundary, AdjustProperties, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
AdjustPreambleLength	BlockHeader, Padding, Padding, PreambleLength, AdjustProperties, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
CheckPeers	BlockHeader, NumberOfPeers, (LengthPeerPortName, PeerPortName, LengthPeerStationName, PeerStationName)*, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
CheckLineDelay	BlockHeader, Padding, Padding, LineDelay
CheckMAUType	BlockHeader, MAUType
CheckLinkState	BlockHeader, LinkState
CheckSyncDifference	BlockHeader, CheckSyncMode

Substitution name	Structure
CheckMAUType-Difference	BlockHeader, MAUTypeMode
PDInterfaceAdjust	BlockHeader, Padding, Padding, MultipleInterfaceMode, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PDInterface-MrpDataAdjust	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Role, [Padding*] ^a , MRP_LengthDomainName, MRP_DomainName, [Padding*] ^a , { [(MrpManagerParams) ^ (MrpClientParams)] } ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PDInterface-MrpDataReal with BlockVersionLow = 0	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Role, MRP_LengthDomainName, MRP_DomainName, [Padding*] ^a , MRP_Version, { [(MrpManagerParams) ^ (MrpClientParams)], [MrpRingStateData] }, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PDInterface-MrpDataReal with BlockVersionLow = 1	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Role, MRP_Version, MRP_LengthDomainName, MRP_DomainName, [Padding*] ^a , { [(MrpManagerParams) ^ (MrpClientParams)], [MrpRingStateData] }, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PDInterface-MrpDataCheck	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Check
PDPortMrpDataAdjust	BlockHeader, Padding, Padding, MRP_DomainUUID
PDPortMrpDataReal	BlockHeader, Padding, Padding, MRP_DomainUUID
MrpManagerParams	BlockHeader, MRP_Prio, MRP_TOPchgT, MRP_TOPNRmax, MRP_TSTshortT, MRP_TSTdefaultT, MRP_TSTNRmax, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
MrpClientParams	BlockHeader, MRP_LNKdownT, MRP_LNKupT, MRP_LNKNRmax
MrpRingStateData	BlockHeader, MRP_RingState
PDPortFODataReal	BlockHeader, Padding, Padding, FiberOpticType, FiberOpticCableType, [FiberOpticManufacturerSpecific*] ^a , [FiberOpticDiagnosisInfo] ^a Only used by legacy nodes.
FiberOpticManufacturerSpecific	BlockHeader, VendorIDHigh, VendorIDLow, VendorBlockType, Data*, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
FiberOpticDiagnosisInfo	BlockHeader, Padding, Padding, FiberOpticPowerBudgetReal, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
FiberOpticPowerBudgetReal	FiberOpticPowerBudgetType ^a ^a FiberOpticPowerBudgetType.CheckEnable shall be set to zero. It has no meaning in this usage.
PDPortFODataAdjust	BlockHeader, Padding, Padding, FiberOpticType, FiberOpticCableType
PDPortFODataCheck	BlockHeader, Padding, Padding, MaintenanceRequiredPowerBudget, MaintenanceDemandedPowerBudget, ErrorPowerBudget
MaintenanceRequired-PowerBudget	FiberOpticPowerBudgetType

Substitution name	Structure
MaintenanceDemanded-PowerBudget	FiberOpticPowerBudgetType
ErrorPowerBudget	FiberOpticPowerBudgetType
PDNCDataCheck	BlockHeader, Padding, Padding, MaintenanceRequiredDropBudget, MaintenanceDemandedDropBudget, ErrorDropBudget
MaintenanceRequired-DropBudget	NCDropBudgetType
MaintenanceDemanded-DropBudget	NCDropBudgetType
ErrorDropBudget	NCDropBudgetType
PDPortStatistic	BlockHeader, Padding, Padding, ifInOctets, ifOutOctets, ifInDiscards, ifOutDiscards, ifInErrors, ifOutErrors, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
PDInterface-FSUDDataAdjust	BlockHeader, [Padding*] ^a , { [FSHelloBlock], [FastStartUpBlock] } ^b ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. ^b At least one optional block shall be existing.
ARFSUDDataAdjust	BlockHeader, [Padding*] ^a , { [FSPParameterBlock], [FastStartUpBlock] } ^b ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. ^b At least one optional block shall be existing.
FSHelloBlock	BlockHeader, [Padding*] ^a , FSHelloMode, FSHelloInterval, FSHelloRetry, FSHelloDelay, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
FSPParameterBlock	BlockHeader, [Padding*] ^a , FSPParameterMode, FSPParameterUUID, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
FastStartUpBlock	BlockHeader, [Padding*] ^a , Data*, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
AutoConfiguration	BlockHeader, [Padding*] ^a , ACCommunication, ACConfiguration, [ACIsynchronous], [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
ACCommunication	BlockHeader, [Padding*] ^a , ACMinDeviceInterval, ACCommunicationProperties, SendClockFactorArray, [Padding*] ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
SubmoduleInputData-Length	SubmoduleDataLength
SubmoduleOutputData-Length	SubmoduleDataLength
ACConfiguration	BlockHeader, [Padding*] ^a , (API, SlotNumber, SubslotNumber, ModuleIdentNumber, SubmoduleIdentNumber, SubmoduleInputDataLength, SubmoduleOutputDataLength)* ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.

Substitution name	Structure
ACIsochronous	BlockHeader, [Padding*] ^a , (API, SlotNumber, SubslotNumber, TimeIOBase ^b , TimeIOInput ^c , TimeIOOutput ^d)* ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned. ^b Minimum value according to the GSDML keyword T_IO_Base ^c Minimum value according to the GSDML keyword T_IO_InputMin ^d Minimum value according to the GSDML keyword T_IO_OutputMin
CombinedObject-Container	{ COContainerContent* }
COContainerContent	BlockHeader, Padding, Padding, API, Slot, Subslot, Padding, Padding, Index, COContainerBlock, [Padding]* ^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
COContainerBlock	Data* Special case "PhysicalDevice" (combined object access point is the interface submodule): [PDPortDataAdjust] ^ [PDPortDataCheck] ^ [PDIRData] ^ [PDSyncData] ^ [PDInterfaceMrpDataAdjust] ^ [PDInterfaceMrpDataCheck] ^ [PDPortMrpDataAdjust] ^ [PDPortFODDataAdjust] ^ [PDPortFODDataCheck] ^ [PDNCDataCheck] ^ [PDInterfaceFSUDDataAdjust] ^ [PDIRSubframeData] ^ [PDInterfaceAdjust] ^ [PDTimeData]
UploadBLOBQuery	BlockHeader, Padding, Padding, FromOffsetData
UploadBLOB	BlockHeader, Padding, Padding, FromOffsetData, NextOffsetData, TotalSize, Data*
NestedDiagnosisInfo	BlockHeader, NumberOfEntries, (IPaddress, StationNameLength, NameOfStationValue, [Padding]* ^a)* ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.

5.2 Transfer syntax

5.2.1 Coding section related to BlockHeader specific fields

5.2.1.1 Coding of the field BlockType

This field shall be coded as data type Unsigned16 with the values according to Table 358. This field identifies the PDU or data block type.

Table 358 – BlockType

Value (hexadecimal)	Used for	Used by
0x0000	Reserved	—
0x0001	AlarmNotification High	RTA-SDU.AlarmNotification-PDU
0x0002	AlarmNotification Low	RTA-SDU.AlarmNotification-PDU
0x0003 – 0x0007	Reserved	—
0x0008	IODWriteReqHeader	PROFINETIO-ServiceReqPDU.IODWriteReq
0x0009	IODReadReqHeader	PROFINETIO-ServiceReqPDU.IODReadReq
0x000A – 0x000F	Reserved	—
0x0010	DiagnosisData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.DiagnosisData RTA-SDU.AlarmNotification-PDU.AlarmPayload.DiagnosisItem.DiagnosisData
0x0011	Reserved	—

Value (hexadecimal)	Used for	Used by
0x0012	ExpectedIdentificationData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.ExpectedIdentificationData
0x0013	RealIdentificationData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.RealIdentificationData
0x0014	SubstituteValue	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.SubstituteValue PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.SubstituteValue
0x0015	RecordInputDataObjectElement	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.RecordInputDataObjectElement
0x0016	RecordOutputDataObjectElement	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.RecordOutputDataObjectElement
0x0017	Reserved	—
0x0018	ARData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.ARData
0x0019	LogBookData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.LogBookData
0x001A	APIData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.APIData
0x001B	SRLData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.ARDataInfo.SRLData
0x001C – 0x001F	Reserved	—
0x0020	I&M0	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.IMData.I&M0
0x0021	I&M1	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.IMData.I&M1 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.IMDataWrite.I&M1
0x0022	I&M2	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.IMData.I&M2 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.IMDataWrite.I&M2
0x0023	I&M3	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.IMData.I&M3 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.IMDataWrite.I&M3

Value (hexadecimal)	Used for	Used by
0x0024	I&M4	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IMData.I&M4 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. IMDataWrite.I&M4
0x0025 – 0x002F	I&M5 – 15 ^a ^a Reserved for additional identification and maintenance data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite
0x0030	I&M0FilterDataSubmodule	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. I&M0FilterData.I&M0FilterDataSubmodule
0x0031	I&M0FilterDataModule	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. I&M0FilterData.I&M0FilterDataModule
0x0032	I&M0FilterDataDevice	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. I&M0FilterData.I&M0FilterDataDevice
0x0033 – 0x0100	Reserved	—
0x0101	ARBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.ARBlockReq
0x0102	IOCRBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.IOCRBlockReq
0x0103	AlarmCRBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.AlarmCRBlockReq
0x0104	ExpectedSubmoduleBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq. ExpectedSubmoduleBlockReq
0x0105	PrmServerBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.PrmServerBlockReq
0x0106	MCRBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.MCRBlockReq
0x0107	ARRPCBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.ARRPCBlockReq
0x0108	ARVendorBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.ARVendorBlockReq
0x0109	IRInfoBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.IRInfoBlock
0x010A	SRInfoBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.SRInfoBlock
0x010B	ARFSUBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.ARFSUBlock
0x010C – 0x010F	Reserved	—
0x0110	IODBlockReq, shall only be used in conjunction with connection establishment phase	PROFINETIO-ServiceReqPDU.IODControlReq.ControlBlockConnect (PrmEnd.req)
0x0111	IODBlockReq, shall only be used in conjunction with a plug alarm event	PROFINETIO-ServiceReqPDU.IODControlReq.ControlBlockPlug (PrmEnd.req)
0x0112	IOXBlockReq, shall be used in conjunction within the connection establishment phase or a PrmBegin/PrmEnd sequence	PROFINETIO-ServiceReqPDU.IOXControlReq.ControlBlockConnect (ApplicationReady.req)

Value (hexadecimal)	Used for	Used by
0x0113	IOXBlockReq, shall only be used in conjunction with a plug alarm event	PROFINETIO- ServiceReqPDU.IOXControlReq.ControlBlockPlug (ApplicationReady.req)
0x0114	ReleaseBlockReq	PROFINETIO- ServiceReqPDU.IODReleaseReq.ReleaseBlock
0x0115	Reserved	—
0x0116	IOXBlockReq, shall only be used in conjunction with connection establishment phase	PROFINETIO- ServiceReqPDU.IOXControlReq.ControlBlockConnect (ReadyForCompanion.req)
0x0117	IOXBlockReq, shall only be used in conjunction with connection establishment phase	PROFINETIO- ServiceReqPDU.IOXControlReq.ControlBlockConnect (ReadyForRT_CLASS_3.req)
0x0118	IODBlockReq	PROFINETIO- ServiceReqPDU.IODControlReq.ControlBlockConnect (PrmBegin.req)
0x0119	SubmoduleListBlock	PROFINETIO- ServiceReqPDU.IODControlReq.SubmoduleListBlock (PrmBegin.req)
0x0118 – 0x01FF	Reserved	—
0x0200	PDPortDataCheck	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataCheck
0x0201	PdevData	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PdevData
0x0202	PDPortDataAdjust	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust
0x0203	PDSyncData	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDSyncData PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDSyncData
0x0204	IsochronousModeData	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. IsochronousModeData PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. IsochronousModeData
0x0205	PDIRData	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDIRData PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRData

Value (hexadecimal)	Used for	Used by
0x0206	PDIRGlobalData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRGlobalData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRGlobalData
0x0207	PDIRFrameData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRFrameData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRFrameData
0x0208	PDIRBeginEndData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRBeginEndData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRBeginEndData
0x0209	AdjustDomainBoundary Sub block for adjusting DomainBoundary	PROFINETIO-ServiceReqPDU.IODReadReq. RecordDataRead. PDPortDataAdjust.AdjustDomainBoundary PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataAdjust.AdjustDomainBoundary
0x020A	Sub block for checking Peers	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckPeers PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataAdjust.CheckPeers
0x020B	Sub block for checking LineDelay	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckLineDelay PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataCheck.CheckLineDelay
0x020C	Sub block for checking MAUType	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckMAUType PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataCheck.CheckMAUType
0x020E	AdjustMAUType Sub block for adjusting MAUType	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustMAUType PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataAdjust.AdjustMAUType
0x020F	PDPortDataReal	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataReal PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PRealData.PDPortDataReal

Value (hexadecimal)	Used for	Used by
0x0210	AdjustMulticastBoundary Sub block for adjusting MulticastBoundary	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustMulticastBoundary PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.AdjustMulticastBoundary
0x0211	PDInterfaceMrpDataAdjust	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataAdjust PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataAdjust
0x0212	PDInterfaceMrpDataReal	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDInterfaceMrpDataReal
0x0213	PDInterfaceMrpDataCheck	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataCheck PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataCheck PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDInterfaceMrpDataCheck
0x0214	PDPortMrpDataAdjust	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPortMrpDataAdjust PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortMrpDataAdjust PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDPortMrpDataAdjust
0x0215	PDPortMrpDataReal	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPortMrpDataReal PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortMrpDataReal
0x0216	MrpManagerParams Sub block for media redundancy manager parameters	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataAdjust.MrpManagerParams PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataAdjust.MrpManagerParams PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal.MrpManagerParams

Value (hexadecimal)	Used for	Used by
0x0217	MrpClientParams Sub block for media redundancy client parameters	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataAdjust.MrpClientParams PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataAdjust.MrpClientParams PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal.MrpClientParams
0x0219	MrpRingStateData Sub block for media redundancy ring state data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal.MrpRingStateData
0x021B	AdjustLinkState Sub block for adjusting LinkState	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustLinkState PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.AdjustLinkState
0x021C	CheckLinkState Sub block for checking LinkState	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckLinkState PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.CheckLinkState
0x021E	CheckSyncDifference Sub block for checking local and remote CableDelay detected by LLDP to discover a sync difference	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckSyncDifference PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataCheck.CheckSyncDifference
0x021F	CheckMAUTypeDifference Sub block for checking local and remote MAUTypes detected by LLDP	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckMAUTypeDifference PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataCheck.CheckMAUTypeDifference
0x0220	PDPorFODataReal	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataReal PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortFODataReal
0x0221	FiberOpticManufacturerSpecific Sub block for reading real fiber optic manufacturerspecific data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataReal.FiberOpticManufacturerSpecific PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortFODataReal.FiberOpticManufacturerSpecific
0x0222	PDPorFODataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortFODataAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDPortFODataAdjust

Value (hexadecimal)	Used for	Used by
0x0223	PDPorFODataCheck	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorFODataCheck PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorFODataCheck PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDPorFODataCheck
0x0224	AdjustPeerToPeerBoundary Sub block for adjusting the peer to peer boundary	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataAdjust.AdjustPeerToPeerBoundary PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataAdjust.AdjustPeerToPeerBoundary
0x0225	AdjustDCPBoundary Sub block for adjusting the DCP boundary	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataAdjust.AdjustDCPBoundary PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataAdjust.AdjustDCPBoundary
0x0226	AdjustPreambleLength Sub block for adjusting the used length of preamble	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataAdjust.AdjustPreambleLength PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataAdjust.AdjustPreambleLength
0x0227	Reserved	—
0x0228	FiberOpticDiagnosisInfo Sub block for reading real fiber optic diagnosis data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorFODataReal.FiberOpticDiagnosisInfo PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.FiberOpticDiagnosisInfo
0x0229	Reserved	—
0x022A	PDIRSubframeData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRSubframeData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRSubframeData PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PdevData.PDIRSubframeData
0x022B	SubframeBlock Sub block for the persistent portion of DFP	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRSubframeData.SubframeBlock PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRSubframeData.SubframeBlock PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PdevData.PDIRSubframeData.SubframeBlock
0x022C	Reserved	—

Value (hexadecimal)	Used for	Used by
0x022D	PDTimeData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. COContainerContent.COContainerBlock.PDTimeData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. COContainerContent.COContainerBlock.PDTimeData
0x022E – 0x022F	Reserved	—
0x0230	PDNCDataCheck	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDNCDataCheck PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDNCDataCheck PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDNCDataCheck
0x0231 – 0x023F	Reserved	—
0x0240	PDInterfaceDataReal	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceDataReal PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDInterfaceDataReal
0x0241 – 0x024F	Reserved	—
0x0250	PDInterfaceAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDInterfaceDataAdjust
0x0251	PDPortStatistic	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortStatistic PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortStatistic
0x0252 – 0x03FF	Reserved	—
0x0400	MultipleBlockHeader	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.MultipleBlockHeader PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.MultipleBlockHeader
0x0401	COContainerContent	PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. CombinedObjectContainer.COContainerContent
0x0402 – 0x04FF	Reserved	—
0x0500	RecordDataReadQuery	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataReadQuery
0x0501 – 0x05FF	Reserved	—

Value (hexadecimal)	Used for	Used by
0x0600	FSHelloBlock	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceFSUDDataAdjust.FSHelloBlock PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceFSUDDataAdjust.FSHelloBlock PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDInterfaceFSUDDataAdjust.FSHelloBlock
0x0601	FSPParameterBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.IOCBlockReq. ARFSUBlock.ARFSUDDataAdjust.FSPParameterBlock
0x0602 – 0x607	Reserved for FastStartUp	—
0x0608	PDInterfaceFSUDDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDInterfaceFSUDDataAdjust
0x0609	ARFSUDDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. ARFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. ARFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODConnectReq.ARFSUBlock. ARFSUDDataAdjust
0x060A – 0x60F	Reserved for FastStartUp	—
0x0610 – 0x06FF	Reserved	—
0x0700	AutoConfiguration Auto configuration data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration
0x0701	AutoConfiguration Auto configuration communication data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration.ACCommunication
0x0702	AutoConfiguration Auto configuration configuration data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration.ACConfiguration
0x0703	AutoConfiguration Auto configuration isochronous data	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration.ACIsynchronous
0x0704 – 0x07FF	Reserved	—
0x0800	Reserved for profiles covering energy saving BlockType for request service	PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite
0x0801	Reserved for profiles covering energy saving BlockType for response service	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead
0x0800 – 0x08FF	Reserved for profiles covering energy saving	—

Value (hexadecimal)	Used for	Used by
0x0900 – 0x09FF	Reserved for profiles covering sequence of events	—
0x0A00	Controller to controller communication Upload BLOB Query	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.UploadBLOBQuery
0x0A01	Controller to controller communication Upload BLOB	PROFINETIO-ServiceResPDU.IODReadRes.RecordDataRead.UploadBLOB
0x0A02	Controller to controller communication Nested diagnosis info	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.NestedDiagnosisInfo
0x0A03 – 0x0AFF	Reserved for profiles covering controller to controller communication	—
0x0B00 – 0x0EFF	Reserved	—
0x0F00	Maintenancelltem	RTA-SDU.Alarmnotification-PDU.AlarmPayload.Maintenancelltem
0x0F01	Upload selected records within Upload&RetrievalItem	RTA-SDU.Alarmnotification-PDU.AlarmPayload.Upload&RetrievalItem
0x0F02	iParameterItem	RTA-SDU.Alarmnotification-PDU.AlarmPayload.iParameterItem
0x0F03	Retrieve selected records within Upload&RetrievalItem	RTA-SDU.Alarmnotification-PDU.AlarmPayload.Upload&RetrievalItem
0x0F04	Retrieve all stored records within Upload&RetrievalItem	RTA-SDU.Alarmnotification-PDU.AlarmPayload.Upload&RetrievalItem
0x0F05 – 0x6FFF	Reserved	—
0x7000 – 0x7FFF	Reserved for user specific data objects	—
0x8000	Reserved	—
0x8001	Alarm Ack High	RTA-SDU.AlarmAck-PDU
0x8002	Alarm Ack Low	RTA-SDU.AlarmAck-PDU
0x8008	IODWriteResHeader	PROFINETIO-ServiceResPDU.IODWriteRes
0x8009	IODReadResHeader	PROFINETIO-ServiceReqPDU.IODReadRes
0x800A – 0x8100	Reserved	—
0x8101	ARBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARBlockRes
0x8102	IOCRBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.IOCRBlockRes
0x8103	AlarmCRBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.AlarmCRBlockRes
0x8104	ModuleDiffBlock	PROFINETIO-ServiceResPDU.IODConnectRes.ModuleDiffBlock PROFINETIO-ServiceResPDU.IODWriteRes.RecordDataRead.ModuleDiffBlock PROFINETIO-ServiceReqPDU.IOXControlReq.ModuleDiffBlock
0x8105	PrmServerBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.PrmServerBlockRes
0x8106	ARServerBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARServerBlockRes
0x8107	ARRPCBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARRPCBlockRes

Value (hexadecimal)	Used for	Used by
0x8108	ARVendorBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARVendorBlockRes
0x8109 – 0x810F	Reserved	—
0x8110	IODBlockRes, shall only be used in conjunction with connection establishment phase	PROFINETIO-ServiceResPDU.IODControlRes.ControlBlockConnect (Prm End.rsp)
0x8111	IODBlockRes, shall only be used in conjunction with a plug alarm event	PROFINETIO-ServiceResPDU.IODControlRes.ControlBlockPlug (Prm End.rsp)
0x8112	IOXBlockRes, shall only be used in conjunction with connection establishment phase	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockConnect (Application Ready.rsp)
0x8113	IOXBlockRes, shall only be used in conjunction with a plug alarm event	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockPlug (Application Ready.rsp)
0x8114	ReleaseBlockRes	PROFINETIO-ServiceResPDU.IODReleaseRes.ReleaseBlock
0x8115	Reserved	—
0x8116	IOXBlockRes, shall only be used in conjunction with connection establishment phase	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockConnect (ReadyForCompanion.rsp)
0x8117	IOXBlockRes, shall only be used in conjunction with connection establishment phase	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockConnect (ReadyForRT_CLASS_3.rsp)
0x8118	IODBlockRes	PROFINETIO-ServiceResPDU.IODControlRes.ControlBlockConnect (PrmBegin.rsp)
Other	Reserved	—

5.2.1.2 Coding of the field BlockLength

This field shall be coded as data type Unsigned16. This field shall contain the number of octets without counting the fields BlockType and BlockLength.

5.2.1.3 Coding of the field BlockVersionHigh

This field shall be coded as data type Unsigned8 with the values according to Table 359.

Table 359 – BlockVersionHigh

Value (hexadecimal)	Meaning	Use
0x00	Reserved	—
0x01	Version 1	Indicates Version 1
0x02 – 0xFF	Reserved	—

5.2.1.4 Coding of the field BlockVersionLow

This field shall be coded as data type Unsigned8 with the values according to Table 360.

Table 360 – BlockVersionLow

Value (hexadecimal)	Meaning	Use
0x00	Version 0	Indicates version 0
0x01	Version 1	Indicates version 1
0x02 – 0xFF	Version 2 to version 255	Indicates version 2 to 255

5.2.2 Coding section related to RTA-SDU specific fields

5.2.2.1 Coding of the field AlarmType

This field shall be coded as data type Unsigned16 with the values according to Table 361.

Table 361 – AlarmType

Value (hexadecimal)	Meaning	Attached to the Diagnosis ASE	USI structured AlarmPayload	AlarmPayload required	Alarm priority ^e
0x0000	Reserved	—	—	—	—
0x0001	Diagnosis	Yes	Yes	Yes ^d	Low
0x0002	Process	No	Yes	Optional	High
0x0003	Pull ^a	No	Yes	Optional	Low
0x0004	Plug	No	Yes	Optional	Low
0x0005	Status	No	Recommended	Optional	Low
0x0006	Update	No	Recommended	Optional	Low
0x0007	Media Redundancy	Yes	Yes	Yes ^d	Low
0x0008	Controlled by supervisor Logical "Pull" of a submodule to withdraw ownership	No	Yes	Optional	Low
0x0009	Released Logical "Plug" of a submodule to return ownership or trigger a reparameterization	No	Yes	Optional	Low
0x000A	Plug Wrong Submodule ^c	No	Yes	Optional	Low
0x000B	Return of Submodule ^f	No	Yes	Optional	Low
0x000C	Diagnosis disappears	Yes	Yes	Optional	Low
0x000D	Multicast communication mismatch notification	Yes	Yes	Yes ^d	Low
0x000E	Port data change notification	Yes	Yes	Yes ^d	Low
0x000F	Sync data changed notification	Yes	Yes	Yes ^d	Low
0x0010	Isochronous mode problem notification	Yes	Yes	Yes ^d	Low
0x0011	Network component problem notification	Yes	Yes	Yes ^d	Low
0x0012	Time data changed notification	Yes	Yes	Yes ^d	Low
0x0013	Dynamic Frame Packing problem notification	Yes	Yes	Yes ^d	Low
0x0014	MRPD problem notification	Yes	Yes	Yes ^d	Low
0x0015	System Redundancy notification	Yes	Yes	Yes ^d	Low

Value (hexadecimal)	Meaning	Attached to the Diagnosis ASE	USI structured AlarmPayload	AlarmPayload required	Alarm priority ^e
0x0016	Multiple interface mismatch notification	Yes	Yes	Yes ^d	Low
0x0017 – 0x001D	Reserved	—	—	—	—
0x001E	Upload and retrieval notification	No	Yes	Yes	Low
0x001F	Pull module ^b	No	Yes	Optional	Low
0x0020 – 0x007F	Manufacturer specific	No	Recommended	Optional	Low
0x0080 – 0x00FF	Reserved for profiles	No	YES	Optional	Low
0x0100 – 0xFFFF	Reserved	—	—	—	—
^a With ARProperties.PullModuleAlarmAllowed(=0) subslot number 0x0001 – 0x8FFF used as “Pull submodule” and subslot number 0 used as “Pull module”. ^b With ARProperties.PullModuleAlarmAllowed(=1) AlarmType(Pull) shall signal pulling of submodule and AlarmType(Pull module) shall signal pulling of module. ^c Only an information for the application of the CMCTL. ^d An AlarmPayload may not exist if a diagnosis disappears. ^e Recommended assignment. ^f Only this AlarmType is used for SharedInput submodules, too.					

5.2.2.2 Coding of the field AlarmSpecifier

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 10: AlarmSpecifier.SequenceNumber

This field shall be coded according to Table 362. By means of the SequenceNumber the receiver detects duplications and reflects the value of the field in the AlarmAck.

Each alarm priority (HIGH or LOW) administrates its own SequenceNumber. It should start with zero and shall be incremented with every AlarmNotification. The Alarm receiver shall accept an arbitrary value as start value.

NOTE The SequenceNumber may start for each AR of an AR set with every switchover with an arbitrary value.

Whenever an Alarm is repeated over different ARs of an ARset the same SequenceNumber shall be used.

For the System Redundancy notification the SequenceNumber is without a meaning and shall be set to zero.

Table 362 – AlarmSpecifier.SequenceNumber

Value (hexadecimal)	Meaning
0x0000 – 0x07FF	Allowed value range

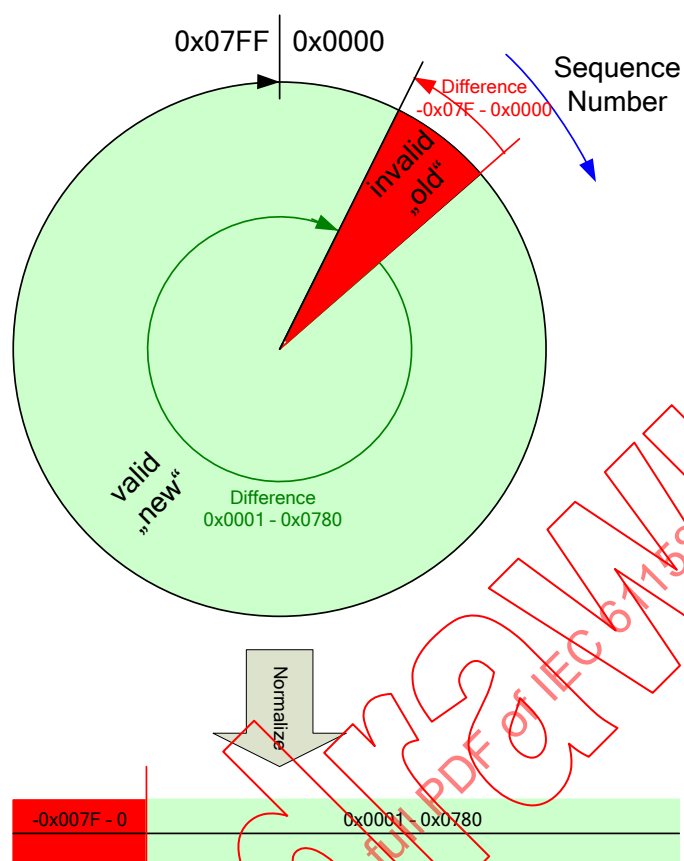


Figure 81 – AlarmSpecifier.SequenceNumber value range

Table 363 defines a maximum SequenceNumber window between an old and a new alarm.

Table 363 – AlarmSpecifier.SequenceNumber Difference

Value (hexadecimal)	Meaning	Use
-0x007F – 0x0000	Red sector according to Figure 81 1/16 of the SequenceNumber range The alarm sink votes the received alarm as older	Alarm is dropped
0x0001 – 0x0780	Green sector according to Figure 81 15/16 of the SequenceNumber range The alarm sink votes the received alarm as newer	Alarm is processed

Bit 11: AlarmSpecifier.ChannelDiagnosis

This field shall be coded with the values according to Table 364. For all other AlarmTypes this field shall be set to zero.

Table 364 – AlarmSpecifier.ChannelDiagnosis

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	Means that the Submodule (alarm source) contains no ChannelDiagnosis, ExtChannelDiagnosis or QualifiedChannelDiagnosis with Channel-Properties.Maintenance (=diagnosis) and Channel-Properties.Specifier (=appear)	All Diagnosis ASE attached AlarmTypes
0x01	Means that the Submodule (alarm source) contains at least one ChannelDiagnosis, ExtChannelDiagnosis or QualifiedChannelDiagnosis with Channel-Properties.Maintenance (=diagnosis) and Channel-Properties.Specifier (=appear)	

Bit 12: AlarmSpecifier.ManufacturerSpecificDiagnosis

This field shall be coded with the values according to Table 365. For all other AlarmTypes this field shall be set to zero.

Table 365 – AlarmSpecifier.ManufacturerSpecificDiagnosis

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	Means that the Submodule (alarm source) contains no ManufacturerSpecificDiagnosis with Channel-Properties.Maintenance (=diagnosis) and Channel-Properties.Specifier (=appear or disappear)	All Diagnosis ASE attached AlarmTypes
0x01	Means that the Submodule (alarm source) contains at least one ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance (=diagnosis) and ChannelProperties.Specifier (=appear or disappear)	

Bit 13: AlarmSpecifier.SubmoduleDiagnosisState

This field shall be coded with the values according to Table 366. For all other AlarmTypes this field shall be set to zero.

Table 366 – AlarmSpecifier.SubmoduleDiagnosisState

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	Error free No DiagnosisData with ChannelProperties.Maintenance (=diagnosis) and ChannelProperties.Specifier (=appear) exists at the submodule. Furthermore, it indicates that all reported diagnosis has been cleared. An individual “disappears” notification can be omitted. However, even in this case a Manufacturer Specific Diagnosis with ChannelProperties.Maintenance (=diagnosis) and ChannelProperties.Specifier (=disappear) may be present.	All Diagnosis ASE attached AlarmTypes
0x01	At least one DiagnosisData with Channel-Properties.Maintenance (=diagnosis) and Channel-Properties.Specifier (=appear) exists at the submodule.	

Bit 14: AlarmSpecifier.reserved

This field shall be set to zero.

Bit 15: AlarmSpecifier.ARDiagnosisState

This field shall be coded with the values according to Table 367. For all other AlarmTypes this field shall be set to zero.

Table 367 – AlarmSpecifier.ARDiagnosisState

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	Error free No DiagnosisData with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=appear) exists at the AR. Furthermore, it indicates that all reported diagnosis has been cleared. An individual “disappears” notification can be omitted. However, even in this case a Manufacturer Specific Diagnosis with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=disappear) may be present.	All Diagnosis ASE attached AlarmTypes
0x01	At least one DiagnosisData with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=appear) exists at the AR.	

5.2.3 Coding section related to common address fields

5.2.3.1 Coding of the field API

This field, defined as an administrative number, shall be coded as data type Unsigned32 according to Table 368.

Table 368 – API

Value (hexadecimal)	Meaning of API
0	Default
0x00000001 – 0x0000FFFF	Used for profile definitions fitting for 5.2.7.8
0x00010000 – 0xFFFFFFFF	Used for profile definitions beyond 5.2.7.8

NOTE API means Application Process Identifier.

5.2.3.2 Coding of the field SlotNumber

This field shall be coded as data type Unsigned16 according to Table 369.

Table 369 – SlotNumber

Value (hexadecimal)	Meaning of SlotNumber
0 – 0x7FFF	The first usable slot for modules is zero. The last usable slot for modules is 0x7FFF. It may contain gaps.
0x8000 – 0xFFFF	Reserved

5.2.3.3 Coding of the field SubslotNumber

This field shall be coded as data type Unsigned16 according to Table 370.

Table 370 – SubslotNumber

Value (hexadecimal)	Meaning of SubslotNumber	
0x0000	ARProperties.PullModuleAlarmAllowed (=0)	Shall be used to code “Pull module” in conjunction with AlarmType (Pull).
	ARProperties.PullModuleAlarmAllowed (=1)	Useable subslot. The first usable subslot for submodules is zero.
0x0001 – 0x7FFF	The last usable subslot for submodules is 0x7FFF. There may be gaps.	
0x8000 – 0x8FFF	Default Used for 16 interface submodules with up to 255 ports. There may be gaps. See Formula (45) If used by system redundancy, the interface submodule mounted left or down into the rack.	
0x9000 – 0x9FFF	Used for 16 interface submodules with up to 255 ports. There may be gaps. See Formula (45) Used by system redundancy with redundant interface for the interface submodule mounted right or up into the rack.	
0xA000 – 0xFFFF	Reserved	

The subslotnumber shall be evaluated with 0xLIPP with I counting interfaces (PP := 00 means interface itself) and PP counting ports as shown in Formula (45).

$$\text{SubslotNumber} = 0xL000 + 0x0I00 + 0x00PP \quad (45)$$

where

SubslotNumber

is the corresponding value

L

is the basic value for this formula
0x8000 or 0x9000

I

is the number of the interface starting with zero

PP

is the number of the port of the interface starting with one. The value zero means no port.

5.2.3.4 Coding of the field Index

5.2.3.4.1 General

This field shall be coded as data type Unsigned16. The range from zero to 0x7FFF shall be used to address user specific record data objects. The range from 0x8000 to 0xFFFF shall be used for protocol specific function or further protocol extensions. The range from 0x8000 to 0xFFFF is divided into four sections. It is defined for each section whether the fields ARUUID, SlotNumber and SubslotNumber shall be ignored or used. The optional field TargetARUUID shall only be present and evaluated in conjunction with the defined indices.

Furthermore, all services with write access shall only use the valid address space of the established AR context. Read services may address record data beyond this context.

5.2.3.4.2 Usage of the address parameters

The following expressions show the evaluating of the address parameters according to the index range.

The rules to evaluate the address parameter shown in Table 371, Table 372, Table 373, Table 374, Table 375 shall be applied.

NOTE Implicit means from outside an existing AR. Explicit means from inside the existing AR.

Table 371 – Expression 1 (subslot specific)

RPCOperationNmb	ARUUID	Target ARUUID	Validity
Implicit Read	NIL (implicit AR)	Evaluated if needed by the index	API, Slot Number, Subslot Number, Index shall be evaluated
Implicit Read	Not NIL (explicit AR)	—	Not allowed
Explicit Read/Write	NIL (implicit AR)	—	Not allowed
Explicit Read/Write	Not NIL (established AR)	Ignore	API, Slot Number, Subslot Number, Index shall be evaluated

Table 372 – Expression 2 (slot specific)

RPCOperationNmb	ARUUID	Target ARUUID	Validity
Implicit Read	NIL (implicit AR)	Evaluated if needed by the index	API, Slot Number, Index shall be evaluated
Implicit Read	Not NIL (explicit AR)	—	Not allowed
Explicit Read/Write	NIL (implicit AR)	—	Not allowed
Explicit Read/Write	Not NIL (established AR)	Ignore	API, Slot Number, Index shall be evaluated

Table 373 – Expression 3 (AR specific)

RPCOperationNmb	ARUUID	Target ARUUID	Validity
Implicit Read	NIL (implicit AR)	Evaluated if needed by the index	Index shall be evaluated
Implicit Read	Not NIL (explicit AR)	—	Not allowed
Explicit Read/Write	NIL (implicit AR)	—	Not allowed
Explicit Read/Write	Not NIL (established AR)	Ignore	Index shall be evaluated

Table 374 – Expression 4 (API specific)

RPCOperationNmb	ARUUID	Target ARUUID	Validity
Implicit Read	NIL (implicit AR)	Ignore	API, Index shall be evaluated
Implicit Read	Not NIL (explicit AR)	—	Not allowed
Explicit Read/Write	NIL (implicit AR)	—	Not allowed
Explicit Read/Write	Not NIL (established AR)	Ignore	API, Index shall be evaluated

Table 375 – Expression 5 (device specific)

RPCOperationNmb	ARUUID	Target ARUUID	Validity
Implicit Read	NIL (implicit AR)	Ignore	Index shall be evaluated
Implicit Read	Not NIL (explicit AR)	—	Not allowed
Explicit Read/Write	NIL (implicit AR)	—	Not allowed
Explicit Read/Write	Not NIL (established AR)	Ignore	Index shall be evaluated

5.2.3.4.3 Grouping of DiagnosisData for the diagnosis records

The diagnosis records contain different data dependent on the index. The grouping shall be according to Table 376. The identifier is used in Table 378, Table 379, Table 380, Table 381 and Table 382.

Table 376 – Grouping of DiagnosisData

Identifier	Meaning
Diagnosis in channel coding	ChannelDiagnosis, ExtChannelDiagnosis and QualifiedChannelDiagnosis with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=appear)
Diagnosis in all codings	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis and ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=appear)
Maintenance required in channel coding	ChannelDiagnosis, ExtChannelDiagnosis and QualifiedChannelDiagnosis with ChannelProperties.Maintenance(=MaintenanceRequired) and ChannelProperties.Specifier(=appear)
Maintenance required in all codings	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis and ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=MaintenanceRequired) and ChannelProperties.Specifier(=appear)
Maintenance demanded in channel coding	ChannelDiagnosis, ExtChannelDiagnosis and QualifiedChannelDiagnosis with ChannelProperties.Maintenance(=MaintenanceDemanded) and ChannelProperties.Specifier(=appear)
Maintenance demanded in all codings	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis and ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=MaintenanceDemanded) and ChannelProperties.Specifier(=appear)
Diagnosis, Maintenance, Qualified and Status	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis, ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=any) and ChannelProperties.Specifier(=appear), and ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=disappear)

5.2.3.4.4 Assigned numbers

The coding for user specific record data shall be according to Table 377.

Table 377 – Index (user specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0 – 0x7FFF	User specific RecordData	Expression 1 applies.

The coding for submodule specific defined record data shall be according to Table 378.

Table 378 – Index (subslot specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUD, API, SlotNumber, SubslotNumber and TargetARUUD
0x8000	ExpectedIdentificationData for one subslot	Expression 1 applies.
0x8001	RealIdentificationData for one subslot	Expression 1 applies.
0x8002 – 0x8009	Reserved	Reserved
0x800A	Diagnosis in channel coding for one subslot	Expression 1 applies.
0x800B	Diagnosis in all codings for one subslot	Expression 1 applies.
0x800C	Diagnosis, Maintenance, Qualified and Status for one subslot	Expression 1 applies.
0x800D – 0x800F	Reserved	Reserved
0x8010	Maintenance required in channel coding for one subslot	Expression 1 applies.
0x8011	Maintenance demanded in channel coding for one subslot	Expression 1 applies.
0x8012	Maintenance required in all codings for one subslot	Expression 1 applies.
0x8013	Maintenance demanded in all codings for one subslot	Expression 1 applies.
0x8014 – 0x801D	Reserved	Reserved
0x801E	SubstituteValues for one subslot	Expression 1 applies.
0x801F	Reserved	Reserved
0x8020	PDIRSubframeData for one subslot	Expression 1 applies.
0x8021 – 0x8027	Reserved	Reserved
0x8028	RecordInputDataObjectElement for one subslot	Expression 1 applies.
0x8029	RecordOutputDataObjectElement for one subslot	Expression 1 applies.
0x802A	PDPortDataReal for one subslot	Expression 1 applies.
0x802B	PDPortDataCheck for one subslot	Expression 1 applies.
0x802C	PDIRData for one subslot	Expression 1 applies.
0x802D	Expected PDSyncData for one subslot with SyncID value 0	Expression 1 applies.
0x802E	Reserved (legacy)	Reserved
0x802F	PDPortDataAdjust for one subslot	Expression 1 applies.
0x8030	IsochronousModeData for one subslot	Expression 1 applies.
0x8031	Expected PDTimeData for one subslot	Expression 1 applies.
0x8032 – 0x804F	Reserved	Reserved
0x8050	PDInterfaceMrpDataReal for one subslot	Expression 1 applies.
0x8051	PDInterfaceMrpDataCheck for one subslot	Expression 1 applies.
0x8052	PDInterfaceMrpDataAdjust for one subslot	Expression 1 applies.
0x8053	PDPortMrpDataAdjust for one subslot	Expression 1 applies.
0x8054	PDPortMrpDataReal for one subslot	Expression 1 applies.
0x8055 – 0x805F	Reserved	Reserved
0x8060	PDPortFODDataReal for one subslot	Expression 1 applies.

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0x8061	PDPortFODataCheck for one subslot	Expression 1 applies.
0x8062	PDPortFODataAdjust for one subslot	Expression 1 applies.
0x8063 – 0x806F	Reserved	Reserved
0x8070	PDNCDataCheck for one subslot	Expression 1 applies.
0x8071	PDInterfaceAdjust for one subslot	Expression 1 applies.
0x8072	PDPortStatistic for one subslot	Expression 1 applies.
0x8073 – 0x807F	Reserved	Reserved
0x8080	PDInterfaceDataReal for one subslot	Expression 1 applies.
0x8081 – 0x808F	Reserved	Reserved
0x8090	Expected PDInterfaceFSUDataAdjust	Expression 1 applies.
0x8091 – 0x809F	Reserved	Reserved
0x80A0	Profiles covering energy saving – Record_0	Expression 1 applies.
0x80A1 – 0x80AF	Reserved for profiles covering energy saving	Expression 1 applies.
0x80B0	CombinedObjectContainer	Expression 1 applies.
0x80B1 – 0x80BF	Reserved	Reserved
0x80C0	Profiles covering sequence of events – Record_0	Expression 1 applies.
0x80C1 – 0x80CF	Reserved for profiles covering sequence of events	Expression 1 applies.
0x80D0 – 0xAFEF	Reserved	Reserved
0xAFF0	I&M0	Expression 1 applies.
0xAFF1	I&M1	Expression 1 applies.
0xAFF2	I&M2	Expression 1 applies.
0xAFF3	I&M3	Expression 1 applies.
0xAFF4	I&M4	Expression 1 applies.
0xAFF5 – 0xAFFF	I&M5 – I&M15 ^a ^a Reserved for additional identification and maintenance data	Expression 1 applies.
0xB000 – 0xBFFF	Reserved for profiles	Expression 1 applies.

The coding for module specific defined record data shall be according to Table 379.

Table 379 – Index (slot specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0xC000	ExpectedIdentificationData for one slot	Expression 2 applies.
0xC001	RealIdentificationData for one slot	Expression 2 applies.
0xC002 – 0xC009	Reserved	Reserved
0xC00A	Diagnosis in channel coding for one slot	Expression 2 applies.
0xC00B	Diagnosis in all codings for one slot	Expression 2 applies.

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0xC00C	Diagnosis, Maintenance, Qualified and Status for one slot	Expression 2 applies.
0xC00D – 0xC00F	Reserved	Reserved
0xC010	Maintenance required in channel coding for one slot	Expression 2 applies.
0xC011	Maintenance demanded in channel coding for one slot	Expression 2 applies.
0xC012	Maintenance required in all codings for one slot	Expression 2 applies.
0xC013	Maintenance demanded in all codings for one slot	Expression 2 applies.
0xC014 – 0xCFFF	Reserved	Reserved
0xD000 – 0xDFFF	Reserved for profiles	Expression 2 applies.

The coding for AR specific defined record data shall be according to Table 380.

Table 380 – Index (AR specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0xE000	ExpectedIdentificationData for one AR	Expression 3 applies.
0xE001	RealIdentificationData for one AR	Expression 3 applies.
0xE002	ModuleDiffBlock for one AR	Expression 3 applies.
0xE003 – 0xE009	Reserved	Reserved
0xE00A	Diagnosis in channel coding for one AR	Expression 3 applies.
0xE00B	Diagnosis in all codings for one AR	Expression 3 applies.
0xE00C	Diagnosis, Maintenance, Qualified and Status for one AR	Expression 3 applies.
0xE00D – 0xE00F	Reserved	Reserved
0xE010	Maintenance required in channel coding for one AR	Expression 3 applies.
0xE011	Maintenance demanded in channel coding for one AR	Expression 3 applies.
0xE012	Maintenance required in all codings for one AR	Expression 3 applies.
0xE013	Maintenance demanded in all codings for one AR	Expression 3 applies.
0xE014 – 0xE02F	Reserved	Reserved
0xE030	Reserved	Reserved
0xE031 – 0xE03F	Reserved	Reserved
0xE040	WriteMultiple	Expression 3 applies.
0xE041 – 0xE04F	Reserved	Reserved

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0xE050	ARFSUDDataAdjust data for one AR Legacy, used only for ARProperties.StartupMode == Legacy	Expression 3 applies.
0xE051 – 0xE05F	Reserved for FastStartUp	Expression 3 applies.
0xE060 – 0xEBFF	Reserved	Reserved
0xEC00 – 0xEFFF	Reserved for profiles	Expression 3 applies.

The coding for API specific defined record data shall be according to Table 381 if no AR has been established.

Table 381 – Index (API specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, API, SlotNumber, SubslotNumber and TargetARUUID
0xF000	RealIdentificationData for one API	Expression 4 applies.
0xF001 – 0xF009	Reserved	Reserved
0xF00A	Diagnosis in channel coding for one API	Expression 4 applies.
0xF00B	Diagnosis in all codings for one API	Expression 4 applies.
0xF00C	Diagnosis, Maintenance, Qualified and Status for one API	Expression 4 applies.
0xF00D – 0xF00F	Reserved	Reserved
0xF010	Maintenance required in channel coding for one API	Expression 4 applies.
0xF011	Maintenance demanded in channel coding for one API	Expression 4 applies.
0xF012	Maintenance required in all codings for one API	Expression 4 applies.
0xF013	Maintenance demanded in all codings for one API	Expression 4 applies.
0xF014 – 0xF01F	Reserved	Reserved
0xF020	ARData for one API	Expression 4 applies.
0xF021 – 0xF3FF	Reserved	Reserved
0xF400 – 0xF7FF	Reserved for profiles	Expression 4 applies.

The coding for device specific defined record data shall be according to Table 382.

Table 382 – Index (device specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUID, API, SlotNumber, SubslotNumber and TargetARUID
0xF800 – 0xF80B	Reserved	Reserved
0xF80C	Diagnosis, Maintenance, Qualified and Status for one device	Expression 5 applies.
0xF80D – 0xF81F	Reserved	Reserved
0xF820	ARData	Expression 5 applies.
0xF821	APIData	Expression 5 applies.
0xF822 – 0xF82F	Reserved	Reserved
0xF830	LogBookData	Expression 5 applies.
0xF831	PdevData	Expression 5 applies.
0xF832 – 0xF83F	Reserved	Reserved
0xF840	I&M0FilterData	Expression 5 applies.
0xF841	PDRealData	Expression 5 applies.
0xF842	PDExpectedData	Expression 5 applies.
0xF843 – 0xF84F	Reserved	Reserved
0xF850	AutoConfiguration	Expression 5 applies.
0xF851 – F85F	Reserved	Reserved
0xF860	Controller to controller communication GSD upload using UploadBLOBQuery and UploadBLOB	Expression 5 applies.
0xF861	Controller to controller communication Nested diagnosis info	Expression 5 applies.
0xF862 – 0xF86F	Reserved for controller to controller communication	Reserved
0xF870 – 0xFBFE	Reserved	Reserved
0xFBFF	Trigger index for the RPC connection monitoring inside the CMSM	Expression 5 applies.
0xFC00 – 0xFFFF	Reserved for profiles ^a	Expression 5 applies.
^a The index is defined as an administrative number.		

5.2.4 Coding section related to AL services

5.2.4.1 Coding of the field RecordDataLength

This field shall be coded as data type Unsigned32. The field RecordDataLength shall only contain the number of user octets.

5.2.4.2 Coding of the field SeqNumber

This field shall be coded as data type Unsigned16 and shall be incremented for each IODWriteReq and IODReadReq by the requestor and mirrored by the responder; the protocol layer shall not check the SeqNumber.

The check for the correct SeqNumber should be done by the application.

The SeqNumber is valid in the context of a SessionKey and starts with zero.

NOTE The SeqNumber and the SessionKey allows detection of sequence errors.

This field shall contain the value 0 for an implicit AR.

5.2.4.3 Coding of the field ARType

This field shall be coded as data type Unsigned16 with the values according to Table 383.

Table 383 – ARType

Value (hexadecimal)	Meaning	Use
0x0000	Reserved	—
0x0001	IOCARSsingle	—
0x0002 – 0x0005	Reserved	—
0x0006	IOSAR	The supervisor AR is a special form of the IOCARSsingle allowing takeover of the ownership of a submodule
0x0007 – 0x000F	Reserved	—
0x0010	IOCARSsingle using RT_CLASS_3	This is a special form of the IOCARSsingle indicating RT_CLASS_3 communication
0x0011 – 0x001F	Reserved	—
0x0020	IOCARSR	The SR AR is a special form of the IOCARSsingle indicating system redundancy or configure in run usage
0x0021 – 0xFFFF	Reserved	—

5.2.4.4 Coding of the field SessionKey

This field shall be coded as data type Unsigned16 and shall be increased by one for each connect by the CMinitiator. The CMResponder shall use this value for each subsequent PROFINETIOServiceReq- and PROFINETIOServiceResPDU within this session.

NOTE The SessionKey allows the CMinitiator to detect sequence errors during the establishment and release phase of an AR.

This field shall contain the value zero for the implicit AR.

5.2.4.5 Coding of the field CMinitiatorMacAdd

This field shall be coded as data type OctetString[6]. The value of the field CMinitiatorMacAdd shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

5.2.4.6 Coding of the field IOCRMulticastMACAdd

This field shall be coded as data type OctetString[6]. The value of the field IOCRMulticastMACAdd shall be set according to the 48-bit universal LAN MAC addresses of IEEE 802, to Table 384 and Table 385.

The IP Multicast address used for multicast communication relation using RT_CLASS_UDP shall be set according RFC 2365.

Table 384 – IOCRMulticastMACAdd using RT_CLASS_UDP

IP multicast address	Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Associated FrameID	Usage
239.192.248.0	01-00-5E	40-F8-00	0xF800	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.192.248.1 – 239.192.251.254	01-00-5E	40-F8-01 to 40-FB-FE	0xF801 – 0xFBFE	Used for multicast communication relations in conjunction with RT_CLASS_UDP
239.192.251.255	01-00-5E	40-FB-FF	0xFBFF	Used for multicast communication relations in conjunction with RT_CLASS_UDP

Table 385 – IOCRMulticastMACAdd using RT_CLASS_2 or RT_CLASS_3

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-00-00 to 00-00-FF	Reserved for other applications
01-0E-CF	00-01-00	Reserved for further multicast addresses within the Type 10 context
01-0E-CF	00-01-01	RT_CLASS_3 destination multicast address
01-0E-CF	00-01-02	RT_CLASS_3 invalid frame multicast address
01-0E-CF	00-01-03 to 00-01-FF	Reserved for further multicast addresses within the Type 10 context
01-0E-CF	00-02-00 to 00-02-FF	RT_CLASS_2 multicast communication address
01-0E-CF	00-03-00 to 00-03-FF	Reserved for further multicast addresses within the Type 10 context
01-0E-CF	00-04-00 to FF-FF-FF	Reserved for other applications
03-00-00 to FF-FF-00	00-00-00	Fast Forwarding local administered multicast address range

NOTE Octet 1 contains the Individual/Group Address Bit (Isb).

The Organizationally Unique Identifier (OUI) as defined by IEEE 802 for Type 10 is 00-0E-CF and shall be set according to Table 386.

Table 386 – Type 10 OUI

Value (hexadecimal)	Meaning
00-0E-CF	Global administered individual unicast
01-0E-CF	Global administered group (multicast) address
02-0E-CF	Local administered individual unicast
03-0E-CF	Local administered group (multicast) address

5.2.4.7 Coding of the field CMResponderMacAdd

This field shall be coded as data type OctetString[6]. The value of the field CMResponderMacAdd shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

NOTE Octet 1 contains the Individual/Group Address Bit (Isb).

5.2.4.8 Coding of the field ARProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 2: ARProperties.State

This field shall be set according to Table 387.

Table 387 – ARProperties.State

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Active
0x02 – 0x07	Reserved

Bit 3: ARProperties.SupervisorTakeoverAllowed

This field shall be set according to Table 388.

Table 388 – ARProperties.SupervisorTakeoverAllowed

Value (hexadecimal)	Meaning
0x00	Not allowed
0x01	Allowed

Bit 4: ARProperties.ParameterizationServer

This field shall be set according to Table 389.

Table 389 – ARProperties.ParameterizationServer

Value (hexadecimal)	Meaning
0x00	External PrmServer
0x01	CM Initiator

Bit 5 – 7: ARProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 8: ARProperties.DeviceAccess

This field shall be set according to Table 390.

Table 390 – ARProperties.DeviceAccess

Value (hexadecimal)	Meaning
0x00	Only the submodules from the ExpectedSubmoduleBlock are accessible
0x01	Submodule access is controlled by IO device application

Bit 9-10: ARProperties.CompanionAR

This field shall be set according to Table 391.

Table 391 – ARProperties.CompanionAR

Value (hexadecimal)	Meaning	Use
0x00	Single AR	RT_CLASS_1, RT_CLASS_2, RT_CLASS_3, or RT_CLASS_UDP connection establishment
0x01	First AR of a companion pair and a companion AR shall follow	For future use
0x02	Companion AR	For future use
0x03	Reserved	—

Bit 11: ARProperties.AcknowledgeCompanionAR

This field shall be set according to Table 392.

Table 392 – ARProperties.AcknowledgeCompanionAR

Value (hexadecimal)	Meaning	Use
0x00	No companion AR or no acknowledge for the companion AR required	RT_CLASS_1, RT_CLASS_2, RT_CLASS_3, or RT_CLASS_UDP connection establishment
0x01	Companion AR with acknowledge	For future use

Bit 12 – 23: ARProperties.reserved_2

This field shall be set according to 3.4.2.2.

Bit 24 – 29: ARProperties.reserved_3

This field shall be set to zero.

Bit 30: ARProperties.StartupMode

This field shall be set according to Table 393.

Table 393 – ARProperties.StartupMode

Value (hexadecimal)	Meaning	Use
0x00	Legacy	Recommended Compatibility mode for legacy nodes. The two provider mode for the RT_CLASS_3 startup is required
0x01	Advanced	Mandatory for this standard.

Bit 31: ARProperties.PullModuleAlarmAllowed

This field shall be set according to Table 394.

Table 394 – ARProperties.PullModuleAlarmAllowed

Value (hexadecimal)	Meaning	Use
0x00	The AlarmType(=Pull) shall signal pulling of submodule and module. The subplot number zero shall code pulling of module in conjunction with AlarmType(=Pull).	Mandatory
0x01	The AlarmType(=Pull) shall signal pulling of submodule. The AlarmType(=Pull module) shall signal pulling of module. The subplot number 0 – 0x8FFF shall code pulling of submodule in conjunction with AlarmType(=Pull).	Optional

5.2.4.9 Coding of the field IOCRProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 3: IOCRProperties.RTClass

This field shall be set according to Table 395.

Table 395 – IOCRProperties.RTClass

Value (hexadecimal)	Meaning	Use
0x00	Reserved	—
0x01	Use FrameID range 7 for the IOCRs See Table 30	Data-RTC-PDU IOCRProperties.RTClass == 0x02 shall be used as substitution by the engineering tool. Shall be supported for legacy IO devices by IO controller and IO supervisor. It should not be generated by an IO device.
0x02	Use FrameID range 6 for the IOCRs See Table 29	Data-RTC-PDU
0x03	Use FrameID range 3 for the IOCRs See Table 26	Data-RTC-PDU
0x04	Use FrameID range 7 for the IOCRs See Table 30	UDP-RTC-PDU
0x05 – 0x07	Reserved	—

Bit 4 – 12: IOCRProperties.reserved_1

This field shall be set to zero.

Bit 13 – 23: IOCRProperties.reserved_2

This field shall be set according to 3.4.2.2.

Bit 24 – 31: IOCRProperties.reserved_3

This field shall be set to zero.

5.2.4.10 Coding of the field NumberOfIODataObjects

This field shall be coded as data type Unsigned16.

5.2.4.11 Coding of the field NumberOfIOCS

This field shall be coded as data type Unsigned16.

5.2.4.12 Coding of the field IOCRReference

This field shall be coded as data type Unsigned16. It is an identification tag for the CR and is used within the IOCRBlockReq and IOCRBlockRes to reference the DataItem. Furthermore, it is used in PDIRData of the Physical Device ASE.

5.2.4.13 Coding of the field IOCRTagHeader

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0-11: IOCRTagHeader.IOCRVLANID

This field shall be set according to Table 396.

Table 396 – IOCRTagHeader.IOCRVLANID

Value (hexadecimal)	Meaning
0x000	No VLAN
0x001	Default VLAN
0x002 – 0xFFFF	According IEEE 802.1Q

Bit 12: IOCRTagHeader.reserved

This field shall be set to zero.

Bit 13 – 15: IOCRTagHeader.IOUserPriority

This field shall be set according to Table 397.

Table 397 – IOCRTagHeader.IOUserPriority

Value (hexadecimal)	Meaning
0x00 – 0x05	Reserved
0x06	IO CR Priority
0x07	Reserved

5.2.4.14 Coding of the field IOCRType

This field shall be coded as data type Unsigned16 with the values according to Table 398.

Table 398 – IOCRType

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001	Input CR
0x0002	Output CR
0x0003	Multicast Provider CR
0x0004	Multicast Consumer CR
0x0005 – 0xFFFF	Reserved

5.2.4.15 Coding of the field CMInitiatorActivityTimeoutFactor

This field shall be coded as data type Unsigned16 with the values according to Table 399 and Table 400.

Table 399 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=0

Value (decimal)	Meaning	Use
0	Reserved	—
1 – 1 000	With a time base of 100 ms	The IO device monitors the time between Connect response and the subsequent first activity of the IO controller
1 001 – 65 535	Reserved	—

Table 400 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=1 or ARProperties.StartupMode:=1

Value (decimal)	Meaning	Use
0 – 99	Reserved	—
100 – 1 000	With a time base of 100 ms	Default value: 200 The IO device monitors the time between Connect response and the subsequent Read and Write record activity of the IO controller
1 001 – 65 535	Reserved	—

This field shall be used from the CMDEV for the calculation of the CMInitiatorActivityTimeout according to Formula (46).

$$\text{CMInitiatorActivityTimeout} = \text{CMInitiatorActivityTimeoutFactor} \times 100 \text{ ms} \quad (46)$$

where

CMInitiatorActivityTimeout is the CM initiator activity timeout

CMInitiatorActivityTimeoutFactor is the CM initiator activity timeout factor

This field shall be used from the CMCTL for the calculation of the CMInitiatorTriggerTimeoutFactor which is used for the AR establishing with the values according to Formula (47) and Table 401.

$$\text{CMInitiatorTriggerTimeoutFactor} = \text{CMInitiatorActivityTimeoutFactor} \text{ DIV } 3 \quad (47)$$

where

CMInitiatorActivityTimeoutFactor is the CM initiator activity timeout factor

CMInitiatorTriggerTimeoutFactor is the CM initiator trigger timeout factor

Table 401 – CMInitiatorTriggerTimeoutFactor

Value (decimal)	Meaning	Use
0 – 32	Reserved	—
33 – 333	With a time base of 100 ms	Used for the connection monitoring during startup until the PPM / CPM connection monitoring takes place.
334 – 65 535	Reserved	—

The value of the CMInitiatorTriggerTimeout shall be calculated according to Formula (48).

$$\text{CMInitiatorTriggerTimeout} = \text{CMInitiatorTriggerTimeoutFactor} \times 100 \text{ ms} \quad (48)$$

where

CMInitiatorTriggerTimeout is the CM initiator trigger timeout
 CMInitiatorTriggerTimeoutFactor is the CM initiator trigger timeout factor

5.2.4.16 Coding of the field StationNameLength

This field shall be coded as data type Unsigned16.

5.2.4.17 Coding of the field CMInitiatorStationName

This field shall be coded according to the rules of 4.3.1.4.15.1.

5.2.4.18 Coding of the field CMResponderStationName

This field shall be coded according to the rules of 4.3.1.4.15.1.

5.2.4.19 Coding of the field ParameterServerStationName

This field shall be coded according to the rules of 4.3.1.4.15.1.

5.2.4.20 Coding of the field ProviderStationName

This field shall be coded according to the rules of 4.3.1.4.15.1.

5.2.4.21 Coding of the field IODataObjectFrameOffset

This field shall be coded as data type Unsigned16. It is used within the IOCRBlockReq to reference the offset of the DataItem. It shall be in a range off 0 to 1 439. The maximum number of used octets shall not exceed the maximum C_SDU size.

5.2.4.22 Coding of the field IOCSFrameOffset

This field shall be coded as data type Unsigned16. It is used within the IOCRBlockReq to reference the offset of the IOCS. It shall be in a range off 0 to 1 439. The maximum number of used octets shall not exceed the maximum C_SDU size.

5.2.4.23 Coding of the field LengthIOCS

This field shall be coded as data type Unsigned8 and shall be set according to Table 402.

Table 402 – LengthIOCS

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	One octet for IOCS
0x02 – 0xFF	Reserved

5.2.4.24 Coding of the field LengthIOPS

This field shall be coded as data type Unsigned8 and shall be set according to Table 403.

Table 403 – LengthIOPS

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	One octet for IOPS
0x02 – 0xFF	Reserved

5.2.4.25 Coding of the field LengthData

This field shall be coded as data type Unsigned16. The values shall be in the range from 0 to 1 439.

5.2.4.26 Coding of the field NumberOfAPIs

This field shall be coded as data type Unsigned16.

NOTE The value 0 is only used in conjunction with ARProperties.DeviceAccess = 1.

5.2.4.27 Coding of the field AlarmCRProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0: AlarmCRProperties.Priority

This field shall be set according to Table 404.

Table 404 – AlarmCRProperties.Priority

Value (hexadecimal)	Meaning
0x00	User priority (default) The priority given by the user is used and two alarm resources are available
0x01	Fixed priority Use only low priority, user priority is ignored and only one alarm resource is available

An IO device as alarm sink shall always support AlarmCRProperties.Priority:="User priority" and the IO controller as alarm source may choose whether it limits itself to one resource or not.

Bit 1: AlarmCRProperties.Transport

This field shall be set according to Table 405.

Table 405 – AlarmCRProperties.Transport

Value (hexadecimal)	Meaning	Usage
0x00	RTA_CLASS_1	Alarm CR uses DATA-RTA-PDU
0x01	RTA_CLASS_UDP	Alarm CR uses UDP-RTA-PDU

Bit 2 – 23: AlarmCRProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 24 – 31: AlarmCRProperties.reserved_2

This field shall be set to zero.

5.2.4.28 Coding of the field AlarmCRTagHeaderHigh

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0-11: AlarmCRTagHeaderHigh.AlarmCRVLANID

This field shall be set according to Table 406.

Table 406 – AlarmCRTagHeaderHigh.AlarmCRVLANID

Value (hexadecimal)	Meaning
0x000	No VLAN
0x001	Default VLAN
0x002 – 0xFFFF	According IEEE 802.1Q

Bit 12: AlarmCRTagHeaderHigh.reserved

This field shall be set to zero.

Bit 13 – 15: AlarmCRTagHeaderHigh.AlarmUserPriority

This field shall be set according to Table 407.

Table 407 – AlarmCRTagHeaderHigh.AlarmUserPriority

Value (hexadecimal)	Meaning
0x00 – 0x05	Reserved
0x06	Alarm CR Priority High
0x07	Reserved

5.2.4.29 Coding of the field AlarmCRTagHeaderLow

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0-11: AlarmCRTagHeaderLow.AlarmCRVLANID

This field shall be set according to Table 408.

Table 408 – AlarmCRTagHeaderLow.AlarmCRVLANID

Value (hexadecimal)	Meaning
0x000	No VLAN
0x001	Default VLAN
0x002 – 0xFFFF	According IEEE 802.1Q

Bit 12: AlarmCRTagHeaderLow.reserved

This field shall be set to zero.

Bit 13 – 15: AlarmCRTagHeaderLow.AlarmUserPriority

This field shall be set according to Table 409.

Table 409 – AlarmCRTagHeaderLow.AlarmUserPriority

Value (hexadecimal)	Meaning
0x00 – 0x04	Reserved
0x05	Alarm CR Priority Low
0x06 – 0x07	Reserved

5.2.4.30 Coding of the field AlarmSequenceNumber

This field shall be coded as data type Unsigned16 with the values according to Table 410.

Table 410 – AlarmSequenceNumber

Value (hexadecimal)	Meaning	Use
0x000 – 0x7FF	Mirrors the sequence number of the plug alarm notification	It provides the relation between Plug Alarm notification and the ControlBlockPlug within Application Ready
0x800 – 0xFFFF	Reserved	—

5.2.4.31 Coding of the field AlarmCRType

This field shall be coded as data type Unsigned16 with the values according to Table 411.

Table 411 – AlarmCRType

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001	Alarm CR
0x0002 – 0xFFFF	Reserved

5.2.4.32 Coding of the field RTATimeoutFactor

This field shall be coded as data type Unsigned16 with the values according to Table 412.
The time base is 100 ms.

The RTATimeout shall be calculated according to Formula (49).

$$\text{RTATimeout} = \text{RTATimeoutFactor} \times 100 \text{ ms} \quad (49)$$

where

RTATimeout is the RTA timeout

RTATimeoutFactor is the RTA timeout factor

Table 412 – RTATimeoutFactor

Value (hexadecimal)	Meaning	Use
0x0000	Reserved	—
0x0001	Mandatory	Default DATA-RTA-PDU or UDP-RTA-PDU acknowledge monitoring
0x0002 – 0x0064	Mandatory	DATA-RTA-PDU or UDP-RTA-PDU acknowledge monitoring
0x0065 – 0xFFFF	Optional	DATA-RTA-PDU or UDP-RTA-PDU acknowledge monitoring

NOTE The value of the RTATimeoutFactor depends on the conveyance time of the DATA-RTA-PDU or UDP-RTA-PDU between two peers.

5.2.4.33 Coding of the field RTARetries

This field shall be coded as data type Unsigned16 with the values according to Table 413.

Table 413 – RTARetries

Value (hexadecimal)	Meaning	Meaning
0x0000 – 0x0002	Reserved	Reserved
0x0003	Mandatory	Default
0x0004 – 0x000F	Mandatory	—
0x0010 – 0xFFFF	Reserved	Reserved

5.2.4.34 Coding of the field PROFINETIOConstantValue

This field shall be coded as data type structure containing the following elements:

- Data1 as Unsigned32 (0xDEA00000)
- Data2 as Unsigned16 (0x6C97)
- Data3 as Unsigned16 (0x11D1)
- Data4 as array of two Unsigned8, Octet 1 (0x82) to Octet 2 (0x71)

NOTE The ordering of transmission of the Unsigned32 or Unsigned16 values is according to the RPC Flag Drep (Little Endian or Big Endian) if it is part of the RPCHeader or NDREMapPDU. If it is part of the NDRDataxxx PDUs then only Big Endian Format is used.

The value of the field PROFINETIOConstantValue shall be DEA00000-6C97-11D1-8271.

5.2.4.35 Coding of the field AddressResolutionProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 2: AddressResolutionProperties.Protocol

This field shall be set according to Table 414.

Table 414 – AddressResolutionProperties.Protocol

Value (hexadecimal)	Meaning	Use
0x00	Reserved	—
0x01	DNS	Multicast consumer uses DNS to resolve the multicast provider source MAC address.
0x02	DCP	Multicast consumer uses DCP to resolve the multicast provider source MAC address.
0x03 – 0x07	Reserved	—

Bit 3 – 7: AddressResolutionProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 8 – 15: AddressResolutionProperties.reserved_2

This field shall be set to zero.

Bit 16 – 31: AddressResolutionProperties.Factor

This field shall be coded with the values according to Table 415. The time base is one second. The AddressResolutionInterval shall be calculated according to Formula (50).

Table 415 – AddressResolutionProperties.Factor

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001 – 0x0064	Mandatory
0x0065 – 0xFFFF	Optional

$$\text{AddressResolutionInterval} = \text{AddressResolutionProperties.Factor} \times 1 \text{ s} \quad (50)$$

where

AddressResolutionInterval is the address resolution interval

AddressResolutionProperties.Factor is the factor for the calculation of the address resolution interval

5.2.4.36 Coding of the field MCITimeoutFactor

This field shall be coded as data type Unsigned16 with the values according to Table 416. The time base shall be 100 ms. The MCIMonitoringInterval shall be calculated according to Formula (51).

Table 416 – MCITimeoutFactor

Value (hexadecimal)	Meaning
0x0000 – 0x0064	Mandatory
0x0065 – 0xFFFF	Reserved

$$\text{MCIMonitoringInterval} = \text{MCITimeoutFactor} \times 100 \text{ ms} \quad (51)$$

where

MCIMonitoringInterval is the MCI monitoring interval

MCITimeoutFactor is the factor for the calculation of the MCI monitoring interval

5.2.4.37 Coding of fields related to Instance, DeviceID, VendorID

5.2.4.37.1 General

The Unsigned16 fields Instance, DeviceID and VendorID are divided into two Unsigned8 fields to define the order of the octets inside the ObjectUUID OctetString.

5.2.4.37.2 Coding of the field InstanceLow

This field shall be coded as data type Unsigned8.

5.2.4.37.3 Coding of the field InstanceHigh

This field shall be coded as data type Unsigned8.

The value zero for both InstanceHigh and InstanceLow is recommended for single instance devices.

5.2.4.37.4 Coding of the field DeviceIDLow

This field, defined as an administrative number, shall be coded as data type Unsigned8 by the manufacturer of the device.

The value zero for both DeviceIDHigh and DeviceIDLow is reserved.

5.2.4.37.5 Coding of the field DeviceIDHigh

This field, defined as an administrative number, shall be coded as data type Unsigned8 by the manufacturer of the device.

The value zero for both DeviceIDHigh and DeviceIDLow is reserved.

5.2.4.37.6 Coding of the field VendorIDLow

This field, defined as an administrative number, shall be coded as data type Unsigned8 with the values according to Table 417.

Table 417 – VendorIDLow

Value (hexadecimal)	Meaning
0x00	The value zero for both VendorIDHigh and VendorIDLow is reserved for IEC 61158-6-3 devices.
0x01 – 0xFF	Administrative number

5.2.4.37.7 Coding of the field VendorIDHigh

This field, defined as an administrative number, shall be coded as data type Unsigned8 with the values according to Table 418.

Table 418 – VendorIDHigh

Value (hexadecimal)	Meaning
0x00	The value 0 for both VendorIDHigh and VendorIDLow is reserved for IEC 61158-6-3 devices.
0x01 – 0xEF	Administrative number
0xFF	The value 0xFF is reserved for profiles.

5.2.4.38 Coding of the field ModuleIdentNumber

This field shall be coded as data type Unsigned32. This field shall be coded with the values according to Table 419.

Table 419 – ModuleIdentNumber

Value (hexadecimal)	Meaning
0x00000000	Reserved
0x00000001 – 0xFFFFFFFF	Manufacturer specific

5.2.4.39 Coding of the field SubmoduleIdentNumber

This field shall be coded as data type Unsigned32. This field shall be coded with the values according to Table 420.

Table 420 – SubmoduleIdentNumber

Value (hexadecimal)	Meaning ARProperties.PullModuleAlarm- Allowed := 0	Meaning ARProperties.PullModuleAlarm- Allowed := 1
0x00000000	Assigned to the subslot zero only in conjunction with Pull alarm and SubslotNumber := 0 as identification for the function pull module. Manufacturer specific for subslots > 0	Manufacturer specific for all other subslots
0x00000001 – 0xFFFFFFFF	Manufacturer specific for all other subslots	

The numbering for device identification is hierarchical. The toplevel is the DeviceIdentNumber which contains an administrative uniquely assigned number. The second level is the ModuleIdentNumber that is unique in the scope of the DeviceIdentNumber. The third level is the SubmoduleIdentNumber that is unique in the scope of the ModuleIdentNumber.

NOTE 1 Unique means, that the associated submodule properties, e.g. data structure, diagnosis and parameterization, defined in this standard are identified.

NOTE 2 The hardware of a submodule is identified by the VendorID and the OrderID.

5.2.4.40 Coding of the field NumberOfARs

This field shall be coded as data type Unsigned16.

5.2.4.41 Coding of the field NumberOfIOCRs

This field shall be coded as data type Unsigned16.

5.2.4.42 Coding of the field SubmoduleDataLength

This field shall be coded as data type Unsigned16. The range shall be between 0 and 1 439.

5.2.4.43 Coding of the field NumberOfModules

This field shall be coded as data type Unsigned16.

5.2.4.44 Coding of the field NumberOfSlots

This field shall be coded as data type Unsigned16.

5.2.4.45 Coding of the field NumberOfSubmodules

This field shall be coded as data type Unsigned16.

5.2.4.46 Coding of the field NumberOfSubslots

This field shall be coded as data type Unsigned16.

5.2.4.47 Coding of the field ARUUID

This field shall be coded as data type UUID according to Table 421 and Table 422.

Table 421 – ARUUID

Value (UUID)	Meaning	Use
00000000-0000-0000-0000-000000000000	Reserved	The value NIL indicates the usage of the implicit AR.
Other	Valid for an AR	Unambiguous values which identifies an ARs. See Table 422 for IOCARSR

Table 422 – ARUUID in conjunction with ARTYPE:= IOCARSR

Value	Description		
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeezzzz	Identifies an application relation where		
	aaaaaaa bbbb (6 octets)	Is similar to the standard ARUUID	
	cccc dddd eeeeeeee (8 octets)	Is used as a configuration info covering the content of the Connect.req.	
	zzzz (2 octets)	Bit 0-2:	01: 1 st AR of an AR-set 02: 2 nd AR of an AR-set 03: 3 rd AR of an AR-set 04: 4 th AR of an AR-set Others: Reserved
		Bit 3-4:	02: Communication endpoint shall allocate two ARs for the AR-set Others: Reserved
		Bit 5-15:	Reserved
The ARUUID of an IOCARSR differs only in Bit 0-2.			

5.2.4.48 Coding of the field TargetARUUID

This field shall be coded as data type UUID according to Table 423.

Table 423 – TargetARUUID

Value (UUID)	Meaning	Use
00000000-0000-0000-0000-000000000000	Reserved	The value NIL indicates the usage of the implicit AR.
other	Valid for an AR	All other values identify established Ars.

5.2.4.49 Coding of the field ActualLocalTimeStamp

This field shall be coded as data type Unsigned64 according to Table 424.

Table 424 – ActualLocalTimeStamp

Value (hexadecimal)	Meaning	Use
0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	Contains the current cycle count value when reading the log book.	—

5.2.4.50 Coding of the field LocalTimeStamp

This field shall be coded as data type Unsigned64 according to Table 425.

Table 425 – LocalTimeStamp

Value (hexadecimal)	Meaning	Use
0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	Contains the current cycle count when storing the entry to the log book.	—

5.2.4.51 Coding of the field NumberOfLogEntries

This field shall be coded as data type Unsigned16 according to Table 426.

Table 426 – NumberOfLogEntries

Value	Meaning	Use
0	Reserved	—
16	Minimum number shall be 16 for an IO device	—
128	Minimum number shall be 128 for an IO controller	—
other	Number of log entries	—

5.2.4.52 Coding of the field EntryDetail

This field shall be coded as data type Unsigned32 according to protocol machine behavior and Table 427.

Table 427 – EntryDetail

Value (hexadecimal)	Meaning	Use
0x00000000	No detail	—
other	Value derived from the signaling protocol machine	—

5.2.4.53 Coding of the field AdditionalValue1 and AdditionalValue2

This field shall be coded as data type Unsigned16 according to Table 428.

Table 428 – AdditionalValue1 and AdditionalValue2

Value (hexadecimal)	Meaning	Use
0x0000	No further information or positive response	IODReadRes, IODWriteRes, IODWriteMultipleRes
other	Additional user information within negative response	IODReadRes, IODWriteRes, IODWriteMultipleRes

5.2.4.54 Coding of the field ControlBlockProperties

The coding of this field shall be according to 3.4.2.4, Table 429, Table 430 and Table 431.

Table 429 – ControlBlockProperties in conjunction with ControlCommand.ApplicationReady with ARProperties.StartupMode:=1

Bitposition	Value (hexadecimal)	Meaning
Bit 0 – 15	—	Shall be set according to 3.4.2.2.

Table 430 – ControlBlockProperties in conjunction with ControlCommand.ApplicationReady with ARProperties.StartupMode:=0

Bitposition	Value (hexadecimal)	Meaning
Bit 0	0x00	Wait for explicit ControlCommand.ReadyForCompanion
	0x01	Implicit ControlCommand.ReadyForCompanion
Bit 1	0x00	Wait for explicit ControlCommand.ReadyForRT_CLASS_3
	0x01	Implicit ControlCommand.ReadyForRT_CLASS_3
Bit 2 – 15	—	Shall be set according to 3.4.2.2.

Table 431 – ControlBlockProperties in conjunction with the other values of the field ControlCommand

Bitposition	Value (hexadecimal)	Meaning
Bit 0 – 15	—	Shall be set according to 3.4.2.2.

5.2.4.55 Coding of the field ControlCommand

The coding of this field shall be according to 3.4.2.4. Only one of the following bits shall be set. The individual bits shall have the following meaning:

Bit 0: ControlCommand.PrmEnd

This field shall be set according to Table 432.

Table 432 – ControlCommand.PrmEnd

Value (hexadecimal)	Meaning	Use
0x00	No PrmEnd	—
0x01	The IO controller has finished the transmission of the stored start-up parameter.	IODControlReq

Bit 1: ControlCommand.ApplicationReady

This field shall be set according to Table 433.

Table 433 – ControlCommand.ApplicationReady

Value (hexadecimal)	Meaning	Use
0x00	No Application Ready	—
0x01	The IO device has finished the start-up of its application or a new module or submodule was plugged and is ready to operate.	IOXControlReq

Bit 2: ControlCommand.Release

This field shall be set according to Table 434.

Table 434 – ControlCommand.Release

Value (hexadecimal)	Meaning	Use
0x00	No Release	—
0x01	The IO controller terminates the AR.	IODReleaseReq

Bit 3: ControlCommand.Done

This field shall be set according to Table 435.

Table 435 – ControlCommand.Done

Value (hexadecimal)	Meaning	Use
0x00	No Done	—
0x01	Acknowledge is sent.	IODReleaseRes, IOXControlRes, IODControlRes

Bit 4: ControlCommand.ReadyForCompanion

This field shall be set according to Table 436.

Table 436 – ControlCommand.ReadyForCompanion

Value (hexadecimal)	Meaning	Use
0x00	Not ready for companion	—
0x01	The IO device has finished the start-up of its application and is ready for the companion AR.	IOXControlReq

Bit 5: ControlCommand.ReadyForRT_CLASS_3

This field shall be set according to Table 437.

Table 437 – ControlCommand.ReadyForRT_CLASS_3

Value (hexadecimal)	Meaning	Use
0x00	Not ready for RT_CLASS_3	—
0x01	The IO device has finished the start-up of its application and is ready for RT_CLASS_3 communication.	IOXControlReq

Bit 6: ControlCommand.PrmBegin

This field shall be set according to Table 438.

Table 438 – ControlCommand.PrmBegin

Value (hexadecimal)	Meaning	Use
0x00	No PrmBegin	—
0x01	The IO controller starts the transmission of the stored start-up parameter.	IODControlReq

Bit 7 to 15: ControlCommand.reserved

This field shall be set to zero.

5.2.4.56 Coding of the field DataDescription

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: DataDescription.Type

This field shall be set according to Table 439.

Table 439 – DataDescription.Type

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Input
0x02	Output
0x03	Reserved

Bit 2-15: DataDescription.reserved

This field shall be set to zero.

5.2.4.57 Coding of the field DataLength

This field shall be coded as data type Unsigned16. The values shall contain the length of the C_SDU or the CSF_SDU according to Table 440.

NOTE The field DataLength is in minimum the sum of all DataItems. It can be larger.

Table 440 – Values of DataLength

Value (decimal)	Domain	Meaning
40 – 1 440	RT_CLASS_1, RT_CLASS_2 and RT_CLASS_3	Valid range
12 – 1 440	RT_CLASS_UDP	Valid range
other	—	Reserved

5.2.4.58 Coding of the field GAP

This field shall be coded as data type Unsigned8. The values shall be set to zero.

5.2.4.59 Coding of the field Padding

This field shall be coded as data type Unsigned8. The values shall be set to zero.

5.2.4.60 Coding of the field RTCPadding

This field shall be coded as data type Unsigned8.

5.2.4.61 Coding of the field RWPadding

This field shall be coded as data type Unsigned8. The values shall be set to zero.

5.2.4.62 Coding of the field SendClockFactor

These fields shall be coded as data type Unsigned16 according to Table 441. The time base is 31,25 µs.

Table 441 – Values of SendClockFactor

Value (decimal)	LengthOfPeriod	Meaning
0	—	Reserved
1	31,25 µs	Optional
2	62,5 µs	Optional
3	93,75 µs	Optional
4	125 µs	Optional
5 – 7	—	Optional
8	250 µs	Optional
9 – 15	—	Optional
16	500 µs	Optional
17 – 31	—	Optional
32	1 ms	Mandatory
33 – 63	—	Optional
64	2 ms	Optional
65 – 127	—	Optional
128	4 ms	Optional
129 – 65 535	—	Reserved

Valid combinations of the SendClockFactor with ReductionRatio are shown in Table 442.

The LengthOfPeriod corresponds with a value range from 31,25 µs to 4 000 µs. A maximum Ethernet frame contains 1 518 or 1 522 octets. To transfer this kind of frame at a data rate of 100 Mbit/s the LengthOfPeriod should be greater than 122,24 µs and at a data rate of 1 Gbit/s the LengthOfPeriod should be greater than 12,224 µs.

The value of the LengthOfPeriod shall be calculated according to Formula (52).

$$\text{LengthOfPeriod} = \text{SendClockFactor} \times 31,25 \mu\text{s} \quad (52)$$

where

LengthOfPeriod is the length of the period

SendClockFactor factor of the send clock, the length of one phase

Used within the PDSyncData block, this field shall only be valid if the field SyncProperties.SyncID contains the value 0.

5.2.4.63 Coding of the field ReductionRatio

These fields shall be coded as data type Unsigned16 with the values according to Table 442, Table 443 and Table 444.

The allowed value shall be calculated in conjunction with Table 441 according Formula (53).

Table 442 – Values of ReductionRatio for RT_CLASS_1 and RT_CLASS_2

Value (decimal)	Meaning	Use
1	Mandatory	Transmit and receive every send clock
2	Mandatory	Transmit and receive every second send clock
4	Mandatory	Transmit and receive every forth send clock
8	Mandatory	Transmit and receive every eighth send clock
16	Mandatory	Transmit and receive every sixteenth send clock
32	Mandatory	Transmit and receive every 32 send clock
64	Mandatory	Transmit and receive every 64 send clock
128	Mandatory	Transmit and receive every 128 send clock
256	Mandatory	Transmit and receive every 256 send clock
512	Mandatory	Transmit and receive every 512 send clock
16 385 – 65 535	Reserved	—
other	Optional	—

Table 443 – Values of ReductionRatio for RT_CLASS_3

Value (decimal)	Meaning	Use
1	Mandatory	Transmit and receive every send clock
2	Mandatory	Transmit and receive every second send clock
4	Mandatory	Transmit and receive every forth send clock
8	Mandatory	Transmit and receive every eighth send clock
16	Mandatory	Transmit and receive every sixteenth send clock
16 385 – 65 535	Reserved	—
other	Optional	—

Table 444 – Values of ReductionRatio for RT_CLASS_UDP

Value (decimal)	Meaning	Use
1	Optional	Transmit and receive every send clock
2	Optional	Transmit and receive every second send clock
4	Optional	Transmit and receive every forth send clock
8	Optional	Transmit and receive every eighth send clock
16	Optional	Transmit and receive every sixteenth send clock
32	Optional	Transmit and receive every 32 send clock
64	Optional	Transmit and receive every 64 send clock
128	Mandatory	Transmit and receive every 128 send clock
256	Mandatory	Transmit and receive every 256 send clock
512	Mandatory	Transmit and receive every 512 send clock
1 024	Mandatory	Transmit and receive every 1 024 send clock
2 048	Mandatory	Transmit and receive every 2 048 send clock
4 096	Mandatory	Transmit and receive every 4 096 send clock

Value (decimal)	Meaning	Use
8 192	Mandatory	Transmit and receive every 8 192 send clock
16 384	Mandatory	Transmit and receive every 16 384 send clock
16 385 – 65 535	Reserved	—
other	Optional	—

5.2.4.64 Coding of the field Phase

These fields shall be coded as data type Unsigned16 according to Table 445. It shall be less or equal to the related ReductionRatio.

Table 445 – Values of Phase

Value (decimal)	Meaning	Use
0	Reserved	—
1 – 16 384	Mandatory	But shall not exceed the value of ReductionRatio
16 385 – 65 535	Reserved	—

5.2.4.65 Coding of the field Sequence

These fields shall be coded as data type Unsigned16 according to Table 446.

Table 446 – Values of Sequence

Value (hexadecimal)	Meaning	Use
0	Mandatory, sequence undefined	The PPMs uses a random sequence for each phase
0x0001 – 0xFFFF	Optional, sequence defined	The PPMs uses the given sequence for each phase

NOTE For RT_CLASS_3 is a given sequence used, even if the parameter Sequence is set to zero.

5.2.4.66 Coding of the field DataHoldFactor

These fields shall be coded as data type Unsigned16. The time base is SendClockFactor multiplied with ReductionRatio of the monitored consumer.

The DataHoldTime shall be calculated according to Formula (53).

$$\text{DataHoldTime} = \text{DataHoldFactor} \times \text{SendClockFactor} \times \text{ReductionRatio} \times 31,25 \mu\text{s} \quad (53)$$

where

DataHoldTime	is the data hold time
DataHoldFactor	is the data hold factor
SendClockFactor	is the send clock factor
ReductionRatio	is the reduction ratio

The value range for Data-RTC-PDUs is from 0x0003 to 0x1E00. The DataHoldTime shall equal or less than 1,92 s.

The value range for UDP-RTC-PDUs is from 0x0003 to 0xF000. The DataHoldTime shall equal or less than 61,44 s.

For a frame, the usage of the DataHoldFactor is defined in Table 447.

Table 447 – DataHoldFactor of a frame

Value (hexadecimal)	Meaning	Description
0x0000	Reserved	—
0x0001 – 0x0002	Optional	A value “1” leads to an AR termination for one missed frame, a value “2” leads to an AR termination for two missed frames.
0x0003 – 0x00FF	Mandatory	An expiration of the time leads to an AR termination.
0x0100 – 0x1E00	Optional	An expiration of the time leads to an AR termination.
0x1E01 – 0xFFFF	Reserved	—

For a Subframe, the usage of the DataHoldFactor is defined in Table 448.

Table 448 – DataHoldFactor of a Subframe

Value (hexadecimal)	Meaning	Description
0x0000	Reserved	—
0x0001 – 0x0002	Optional	A value “1” leads to an AR termination for one missed frame, a value “2” leads to an AR termination for two missed frames.
0x0003 – 0x001F	Mandatory	An expiration of the time leads to an AR termination.
0x0020 – 0x1E00	Optional	An expiration of the time leads to an AR termination.
0x1E01 – 0xFFFF	Reserved	—

5.2.4.67 Coding of the field FrameSendOffset

This field shall be coded as data type Unsigned32 and set according to Table 449, Figure 82 and Formula (54). The time base is 1 ns.

NOTE The minimal planned gap between two RT_CLASS_3 frames is 1 120 ns.

Table 449 – Values of FrameSendOffset

Value (decimal)	Meaning	Use
0 – 0x00001388	Relative send offset to the start of the related cycle	Optional
0x00001389 – 0x003B28FF	Relative send offset to the start of the related cycle	Mandatory for RT_CLASS_3 Optional for RT_CLASS_1, RT_CLASS_2 and RT_CLASS_UDP
0x003B2900 – 0x003D08FF	Relative send offset to the start of the related cycle Only useable if the frame size fulfill Formula (54)	Mandatory for RT_CLASS_3 Optional for RT_CLASS_1, RT_CLASS_2 and RT_CLASS_UDP

Value (decimal)	Meaning	Use
0x003D0900 – 0xFFFFFFFFE	Reserved	Not used
0xFFFFFFFF	Best effort Provider protocol machines sends the frame as soon as possible	Mandatory Not used for PDIRFrameData

$$\text{FrameSendOffset}[\text{Frame}] + \text{Duration}[\text{Frame}] < \text{SendClockFactor} \times 31,25 \mu\text{s} \quad (54)$$

where

FrameSendOffset	is the frame send offset
Frame	is the current frame
Duration	is the duration
SendClockFactor	is the factor of the send clock, the length of one phase

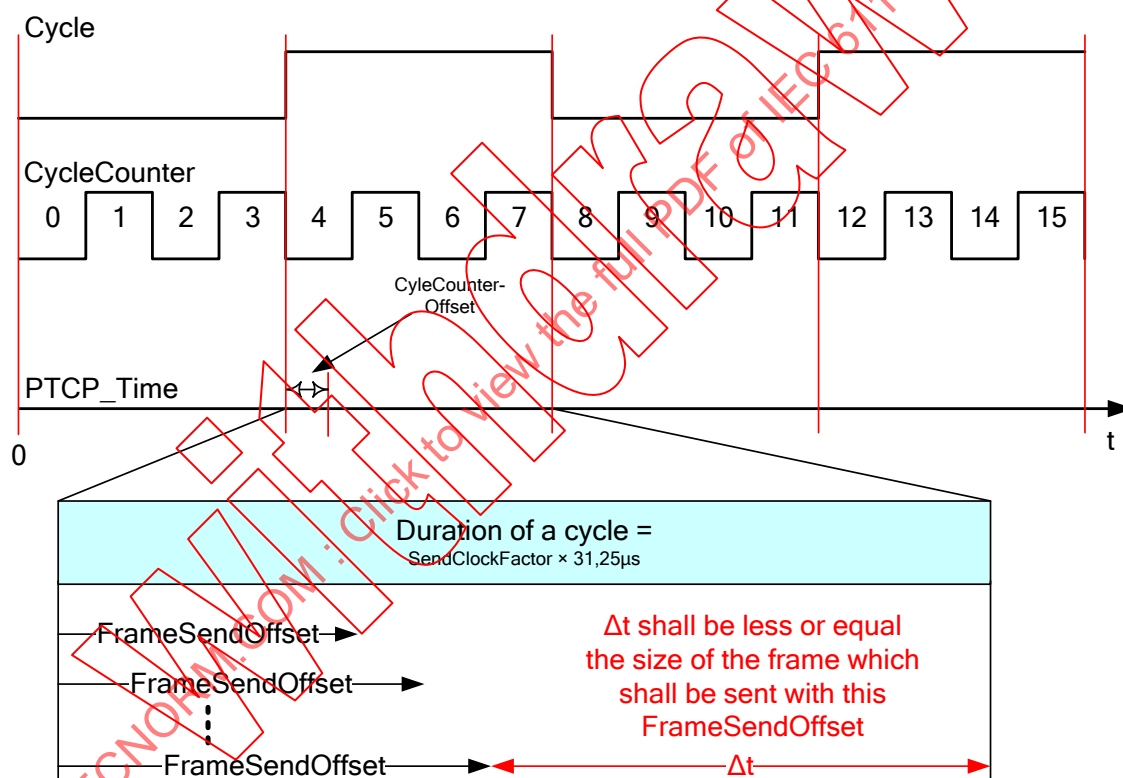


Figure 82 – FrameSendOffset vs. duration of a cycle

5.2.4.68 Coding of the field ModuleState

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 450.

Table 450 – ModuleState

Value (hexadecimal)	Meaning	Use
0x0000	No module	E.g. module not plugged
0x0001	Wrong module ^a	E.g. ModuleIdentNumber wrong
0x0002	Proper module	Module is okay but at least one submodule is locked, wrong or missing
0x0003	Substitute ^b	Module is not the same as requested – but the IO device was able to adapt by its own knowledge
0x0004 – 0xFFFF	Reserved	—
^a This coding should be avoided if possible. The substitution may be “Substitute” for the module and “Wrong (WR)” for each submodule. See Table 457 for valid combinations.		
^b This coding may be used in combination with “SubmoduleState.IdentInfo = Wrong (WR)”.		

5.2.4.69 Coding of the field SubmoduleState

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

5.2.4.69.1 Coding if SubmoduleState.FormatIndicator == 1

Bit 0 – 2: SubmoduleState.AddInfo

This field shall be set according to Table 451.

Table 451 – SubmoduleState.AddInfo

Value (hexadecimal)	Meaning	Description
0x00	None	—
0x01	Takeover is not allowed	This Submodule is not available for takeover by IOSAR.
0x02 – 0x07	Reserved	—

Bit 3: SubmoduleState.QualifiedInfo

This field shall be set according to Table 452.

Table 452 – SubmoduleState.QualifiedInfo

Value (hexadecimal)	Meaning	Description
0x00	No QualifiedInfo available	No Channel of the Submodule contains QualifiedChannelDiagnosis
0x01	QualifiedInfo available	At least one Channel of the Submodule contains QualifiedChannelDiagnosis

Bit 4: SubmoduleState.MaintenanceRequired

This field shall be set according to Table 453.

Table 453 – SubmoduleState.MaintenanceRequired

Value (hexadecimal)	Meaning	Description
0x00	No MaintenanceRequired available	No Channel of the Submodule requires maintenance
0x01	MaintenanceRequired available	At least one Channel of the Submodule requires maintenance

Bit 5: SubmoduleState.MaintenanceDemanded

This field shall be set according to Table 454.

Table 454 – SubmoduleState.MaintenanceDemanded

Value (hexadecimal)	Meaning	Description
0x00	No MaintenanceDemanded available	No Channel of the Submodule demandes maintenance
0x01	MaintenanceDemanded available	At least one Channel of the Submodule demandes maintenance

Bit 6: SubmoduleState.DiagInfo

This field shall be set according to Table 455.

Table 455 – SubmoduleState.DiagInfo

Value (hexadecimal)	Meaning	Description
0x00	No DiagnosisData available	There is no DiagnosisData available/stored for this submodule.
0x01	DiagnosisData available	There is DiagnosisData (with ChannelProperties.Specifier := Appears) available for this submodule: It can be read with the corresponding records.

Bit 7 – 10: SubmoduleState.ARInfo

This field shall be set according to Table 456.

Table 456 – SubmoduleState.ARInfo

Value (hexadecimal)	Meaning	Description
0x00	Own	This AR is owner of the submodule
0x01	ApplicationReadyPending (ARP)	This AR is owner of the submodule but it is blocked. E.g. parameter checking pending
0x02	Superordinated Locked (SO)	This AR is not owner of the submodule. It is blocked by superordinated means
0x03	Locked By IO Controller (IOC)	This AR is not owner of the submodule. It is owned by another IOAR
0x04	Locked By IO Supervisor (IOS)	This AR is not owner of the submodule. It is owned by another IOSAR
0x05 – 0x0F	Reserved	—

Bit 11 – 14: SubmoduleState.IdentInfo

This field shall be set according to Table 457.

Table 457 – SubmoduleState.IdentInfo

Value (hexadecimal)	Meaning	Usable in conjunction with ModuleState “Wrong module”
0x00	OK	No
0x01	Substitute (SU)	No
0x02	Wrong (WR)	Yes
0x03	NoSubmodule (NO)	Yes
0x04 – 0x0F	Reserved	—

Bit 15: SubmoduleState.FormatIndicator

This field shall be set according to Table 458.

Table 458 – SubmoduleState.FormatIndicator

Value (hexadecimal)	Meaning	Use
0x00	Coding uses SubmoduleState.Detail	Shall be supported by an IO controller and IO supervisor.
0x01	Coding uses SubmoduleState.IdentInfo, .ARInfo and .AddInfo	Shall be used by an IO device, IO controller and IO supervisor.

5.2.4.69.2 Coding if SubmoduleState.FormatIndicator == 0**Bit 0 – 14: SubmoduleState.Detail**

This field shall be set according to Table 459.

Table 459 – SubmoduleState.Detail

Value (hexadecimal)	Meaning	Use
0x0000	No submodule	—
0x0001	Wrong submodule	—
0x0002	Locked by IO controller	—
0x0003	Reserved	—
0x0004	Application ready pending	Shall only be used in conjunction with IOXControl request (application ready)
0x0005	Reserved	—
0x0006	Reserved	—
0x0007	Substitute	Submodule is not the same as requested – but the IO device was able to adapt In this case the IO device should adapt (e.g. to new input or output length)
0x0008 – 0x7FFF	Reserved	—

NOTE The field SubmoduleState is only responded if there is a difference between expected and real configuration data. SubmoduleState “GOOD” is not defined because for such submodules no status is reported within the ModuleDiffBlock.

Bit 15: SubmoduleState.FormatIndicator

This field shall be set according to Table 458.

5.2.4.70 Coding of the field SubmoduleProperties

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: SubmoduleProperties.Type

This field shall be set according to Table 460.

Table 460 – SubmoduleProperties.Type

Value (hexadecimal)	Meaning	Use
0x00	No input and no output data	Submodule without IO data, treated as input without data (for IOCS/IOPS), one Input DataDescription Block follows
0x01	Input data	Submodule with input data, one Input DataDescription Block follows
0x02	Output data	Submodule with output data, one Output DataDescription Block follows
0x03	Input and output data	Submodule with input and output data, one Input and one Output DataDescription Block follows

Bit 2: SubmoduleProperties.SharedInput

This field shall be set according to Table 461.

Table 461 – SubmoduleProperties.SharedInput

Value (hexadecimal)	Meaning	Use
0x00	IO controller	IO controller Can be used together with all possible values of SubmoduleProperties.Type.
0x01	IO controller shared	Only the shared IO controller shall use this value. The shared IO controller does not receive alarms and is restricted to read access. The value shall not be used together with SubmoduleProperties.Type=0x02.

Bit 3: SubmoduleProperties.ReduceInputSubmoduleDataLength

This field shall be set according to Table 462.

Table 462 – SubmoduleProperties.ReduceInputSubmoduleDataLength

Value (hexadecimal)	Meaning	Use
0x00	Expected	Use expected input SubmoduleDataLength for I-CR
0x01	Zero	Reduce input SubmoduleDataLength to zero for I-CR

Bit 4: SubmoduleProperties.ReduceOutputSubmoduleDataLength

This field shall be set according to Table 463.

Table 463 – SubmoduleProperties.ReduceOutputSubmoduleDataLength

Value (hexadecimal)	Meaning	Use
0x00	Expected	Use expected output SubmoduleDataLength for O-CR
0x01	Zero	Reduce output SubmoduleDataLength to zero for O-CR

Bit 5: SubmoduleProperties.DiscardIOXS

This field shall be set according to Table 464.

Table 464 – SubmoduleProperties.DiscardIOXS

Value (hexadecimal)	Meaning	Use
0x00	Expected	Merge all expected IOXS for this submodule in the frames of the corresponding CRs
0x01	Zero	Discard all expected IOXS for this submodule in the frames of the corresponding CRs

Bit 6 – 7: SubmoduleProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 8 – 15: SubmoduleProperties.reserved_2

This field shall be set to zero.

5.2.4.71 Coding of the field ModuleProperties

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 7: ModuleProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 8 – 15: ModuleProperties.reserved_2

This field shall be set to zero.

5.2.4.72 Coding of the field SubstitutionMode

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 465.

Table 465 – SubstitutionMode

Value (hexadecimal)	Meaning	Use
0x0000	ZERO	The outputs are set to zero or inactive
0x0001	Last value	Hold the last valid application value
0x0002	Replacement value	Set the output to the configured replacement value
0x0003 – 0x00FF	Reserved	—
0x0100 – 0x01FF	Reserved for profiles	Shall be used for profiles
0x0200 – 0xFFFF	Reserved	—

5.2.4.73 Coding of the field SubstituteActiveFlag

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 466.

Table 466 – SubstituteActiveFlag

Value (hexadecimal)	Meaning	Use
0x0000	Operation	The outputs are set according to normal control process (substitute inactive)
0x0001	Substitute	The outputs are set according to substitute function (substitute active)
0x0002 – 0xFFFF	Reserved	—

5.2.4.74 Coding of the field InitiatorUDPRTPort

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 467.

Table 467 – InitiatorUDPRTPort

Value (hexadecimal)	Meaning	Use
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Optional Port may be blocked by other protocols
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Usable
0x8892	Default for unicast IANA_PNIO_UDP_UNICAST_PORT	Recommended
0x8893	Default for multicast IANA_PNIO_UDP_MULTICAST_PORT	Recommended
0xC000 – 0xFFFF	IANA_FREE_PORT is used as dynamic and/or private port	Usable

5.2.4.75 Coding of the field ResponderUDPRTPort

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 468.

Table 468 – ResponderUDPRTPort

Value (hexadecimal)	Meaning	Use
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Optional Port may be blocked by other protocols
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Usable
0x8892	Default for unicast IANA_PNIO_UDP_UNICAST_PORT	Recommended
0x8893	Default for multicast IANA_PNIO_UDP_MULTICAST_PORT	Recommended
0xC000 – 0xFFFF	IANA_FREE_PORT is used as dynamic and/or private port	Usable

5.2.4.76 Coding of the field InitiatorRPCServerPort

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 469.

Table 469 – InitiatorRPCServerPort

Value (hexadecimal)	Meaning	Use
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Reserved
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Useable
0xC000 – 0xFFFF	IANA_FREE_PORT is used as dynamic and/or private port	Recommended

5.2.4.77 Coding of the field ResponderRPCServerPort

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 470.

Table 470 – ResponderRPCServerPort

Value (hexadecimal)	Meaning	Use
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Reserved
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Useable
0xC000 – 0xFFFF	IANA_FREE_PORT is used as dynamic and/or private port	Recommended

5.2.4.78 Coding of the field LocalAlarmReference

This field shall be coded as data type Unsigned16. This field shall contain the value of the reference of the IO controller within AlarmCRBlockReq and shall contain the value of the reference of the IO device within AlarmCRBlockRes.

In the ARData, this field contains the reference of the IO controller received by the AlarmCRBlockReq.

5.2.4.79 Coding of the field RemoteAlarmReference

This field shall be coded as data type Unsigned16. This field shall contain the value of the reference of the IO device, transmitted within AlarmCRBlockRes to the IO controller, within ARData.

5.2.4.80 Coding of the field MaxAlarmDataLength

This field shall be coded as data type Unsigned16 with values according to Table 471.

Table 471 – MaxAlarmDataLength

Value (hexadecimal)	Meaning
0x00C8	Mandatory Maximal size of the AlarmNotification-PDU accepted by the AlarmEndpoint.
0x00C9 – 0x0598	Optional Maximal size of the AlarmNotification-PDU accepted by the AlarmEndpoint.
other	Reserved

An AlarmEndpoint shall not create larger AlarmNotification-PDUs than negotiated during AR establishment.

5.2.4.81 Coding of the field ParameterServerProperties

This field shall be coded as data type Unsigned32. This field is reserved for future use.

5.2.5 Coding section related to ARVendorBlock

5.2.5.1 Coding of the field APStructureIdentifier

This field, defined as an administrative number, shall be coded as data type Unsigned16 according to Table 472 and Table 473.

Table 472 – APStructureIdentifier with API := 0

Value (hexadecimal)	Meaning
0 – 0x7FFF	Vendor specific This key defines in combination with the key API the structure of Data in ARVendorBlockReq and ARVendorBlockRes.
0x8000	Extended identification rules Used for common profile “Controller to controller communication”
0x8001 – 0xFFFF	Administrative number This key defines in combination with the key API the structure of Data in ARVendorBlockReq and ARVendorBlockRes.

Table 473 – APStructureIdentifier with API != 0

Value (hexadecimal)	Meaning
0 – 0xFFFF	Administrative number for application profiles This key defines in combination with the key API the structure of Data in ARVendorBlockReq and ARVendorBlockRes.

5.2.5.2 Coding of the field ExtendedIdentificationUUID

This field shall be coded as data type UUID.

5.2.5.3 Coding of the field ExtendedIdentificationVersionHigh

This field shall be coded as data type Unsigned8 with the values according to Table 474.

Table 474 – ExtendedIdentificationVersionHigh

Value (hexadecimal)	Meaning	Use
0x00	Reserved	—
0x01	Version 1	Indicates Version 1
0x02 – 0xFF	Version 2 to version 255	Indicates version 2 to 255

5.2.5.4 Coding of the field ExtendedIdentificationVersionLow

This field shall be coded as data type Unsigned8 with the values according to Table 475.

Table 475 – ExtendedIdentificationVersionLow

Value (hexadecimal)	Meaning	Use
0x00	Version 0	Indicates version 0
0x01	Version 1	Indicates version 1
0x02 – 0xFF	Version 2 to version 255	Indicates version 2 to 255

5.2.6 Coding section related to PNIOStatus

5.2.6.1 General

In general, the value ErrorCode=0, ErrorDecode=0, ErrorCode1=0 and ErrorCode2=0 shall be used to indicate “okay”.

Furthermore, in case of an illegal combination of address parameters within an IODReadReq the value ErrorCode=“IODReadRes”, ErrorDecode=“PNIORW”, ErrorCode1=“access-invalid area” and ErrorCode2 may be used to indicate the faulty parameter.

NOTE An illegal address combination is for example TargetARUID==zero and ARUID==zero in ReadExpectedIdentification service.

5.2.6.2 Coding of the field ErrorCode

This field shall be coded as data type Unsigned8. The usage is defined in Table 476.

Table 476 – Values of ErrorCode for negative responses

Value (hexadecimal)	Meaning	Use
0x00	Reserved	Special case “No Error”: ErrorCode = 0, ErrorDecode = 0, ErrorCode1 = 0, ErrorCode2 = 0
0x01 – 0x1F	Reserved	—
0x20 – 0x3F	Manufacturer specific	Within the LogBookData
0x40 – 0x80	Reserved	—
0x81	PNIO	Used for all errors not covered elsewhere.
0x82 – 0xCE	Reserved	—
0xCF	RTA error	Within the ERR-RTA-PDU and UDP-RTA-PDU
0xD0 – 0xD9	Reserved	—

Value (hexadecimal)	Meaning	Use
0xDA	AlarmAck	Within the DATA-RTA-PDU and UDP-RTA-PDU
0xDB	IODConnectRes	Within the CL-RPC-PDU
0xDC	IODReleaseRes	Within the CL-RPC-PDU
0xDD	IODControlRes, IOXControlRes	Within the CL-RPC-PDU
0xDE	IODReadRes	Within the CL-RPC-PDU only used with ErrorDecode=PNIORW
0xDF	IODWriteRes	Within the CL-RPC-PDU only used with ErrorDecode=PNIORW
0xE0 – 0xEF	Reserved	—
0xF0 – 0xFF	Reserved	—

5.2.6.3 Coding of the field ErrorDecode

This field shall be coded as data type Unsigned8. The usage is defined in Table 477.

Table 477 – Values of ErrorDecode

Value (hexadecimal)	Meaning	Use
0x00	Reserved	See Table 476
0x01 – 0x7F	Reserved	—
0x80	PNIORW ^a	Used in context with user error codes of the services Read and Write
0x81	PNIO	Used in context with other services or internal e.g. RPC errors
0x82	Manufacturer Specific	Used only in context of LogBookData
0x83 – 0xFF	Reserved	—
^a Equivalent to IEC 61158-6-3 (DPV1)		

5.2.6.4 Coding of the field ErrorCode1 and ErrorCode2

5.2.6.4.1 General

These fields shall be coded as data type Unsigned8.

The field ErrorDecode defines the coding and meaning of ErrorCode1 and ErrorCode2.

5.2.6.4.2 ErrorDecode value PNIORW

The ErrorDecode value PNIORW indicate that the parameters ErrorCode1 shall be set according to Table 478. The ErrorCode1 consists of ErrorClass and ErrorDecode.

Table 478 – Coding of ErrorCode1 with ErrorDecode PNORW

ErrorCode1		
ErrorClass (decimal) Bit7 – 4	Meaning	ErrorCode (decimal) Bit3 – 0
0 to 9	Not specified	Reserved
10	Application	0 = read error 1 = write error 2 = module failure 3 = not specified 4 = not specified 5 = not specified 6 = not specified 7 = busy 8 = version conflict 9 = feature not supported 10 = User specific 1 11 = User specific 2 12 = User specific 3 13 = User specific 4 14 = User specific 5 15 = User specific 6
11	Access	0 = invalid index 1 = write length error 2 = invalid slot / subslot 3 = type conflict 4 = invalid area / API 5 = state conflict 6 = access denied 7 = invalid range 8 = invalid parameter 9 = invalid type 10 = backup 11 = User specific 7 12 = User specific 8 13 = User specific 9 14 = User specific 10 15 = User specific 11
12	Resource	0 = read constrain conflict 1 = write constrain conflict 2 = resource busy 3 = resource unavailable 4 = not specified 5 = not specified 6 = not specified 7 = not specified 8 = User specific 12 9 = User specific 13 10 = User specific 14 11 = User specific 15 12 = User specific 16 13 = User specific 17 14 = User specific 18 15 = User specific 19
13 to 15	User specific	User specific
NOTE Not specified values are used to serve legacy codes and are intended to be passed unchanged to the application.		

The ErrorDecode value PNORW indicates that the parameter ErrorCode2 shall be set according to Table 479.

Table 479 – Coding of ErrorCode2 with ErrorDecode PNIO

ErrorCode2 (hexadecimal)	Meaning	Usage
0x00 – 0xFF	User specific	—

5.2.6.4.3 ErrorDecode value PNIO

The ErrorDecode value PNIO indicates that the parameter ErrorCode1 shall be set according to Table 480, Table 481, Table 482, Table 483, Table 484, Table 485, Table 486 and the definitions of 4.10.2.3.6.

Table 480 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 1)

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
0	Reserved	0 – 255	Reserved
1	Connect Parameter Error Faulty ARBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		13	Error in Parameter NameOfStation
		14 – 255	Reserved
2	Connect Parameter Error Faulty IOCRBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		28	Error in Parameter IOCSFrameOffset production
		29 – 255	Reserved
3	Connect Parameter Error Faulty ExpectedSubmoduleBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		16	Error in Parameter LengthIOCS production
		17 – 255	Reserved
4	Connect Parameter Error Faulty AlarmCRBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		15	Error in Parameter AlarmCRTagHeaderLow
		16 – 255	Reserved
5	Connect Parameter Error Faulty PrmServerBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		8	Error in Parameter ParameterServerStationName
		9 – 255	Reserved
6	Connect Parameter Error Faulty MCRBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
		...	...
		8	Error in Parameter ProviderStationName
		9 – 255	Reserved
7	Connect Parameter Error Faulty ARRPCBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		4	Error in Parameter InitiatorRPCServerPort
		5 – 255	Reserved
8	Read Write Record Parameter Error Faulty Record	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		12	Error in Parameter TargetARUUID
		13 – 255	Reserved
9	Connect Parameter Error Faulty IRInfoBlock	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		5	Error in Parameter IRDataUUID
		6 – 255	Reserved
10	Connect Parameter Error Faulty SRInfoBlock	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		5	Error in Parameter SRProperties
		6 – 255	Reserved
11	Connect Parameter Error Faulty ARFSUBlock	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		5	FastStartUpBlock
		6 – 255	Reserved
12	Connect Parameter Error Faulty ARVendorBlockReq	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		5	Error in Parameter API
		6	Error in Parameter Data*
		7 – 255	Reserved
13 – 19	Reserved	0 – 255	Reserved
20	IODControl Parameter Error Faulty ControlBlockConnect	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		9	Error in Parameter ControlBlockProperties
		10 – 255	Reserved

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
21	IODControl Parameter Error Faulty ControlBlockPlug	0	Error in Parameter BlockType
		...	...
		9	Error in Parameter ControlBlockProperties
		10 – 255	Reserved
22	IOXControl Parameter Error Faulty ControlBlockConnect after a connection establishment	0	Error in Parameter BlockType
		...	...
		7	Error in Parameter ControlBlockProperties
23	IOXControl Parameter Error Faulty ControlBlockPlug after a plug alarm	8 – 255	Reserved
		0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		7	Error in Parameter ControlBlockProperties
24	IODControl Parameter Error Faulty ControlBlockPrmBegin	8 – 255	Reserved
		0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		9	Error in Parameter ControlBlockProperties
25	IODControl Parameter Error Faulty SubmoduleListBlock	10 – 255	Reserved
		0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		7	Error in Parameter SubslotNumber
26 – 39	Reserved	8 – 255	Reserved
		0 – 255	Reserved
		0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
40	Release Parameter Error Faulty ReleaseBlock	7	Error in Parameter ControlBlockProperties
		8 – 255	Reserved
		0 – 255	Reserved
		0	Error in Parameter BlockType
41 – 49	Reserved	1	Error in Parameter Length
		...	...
		7	Error in Parameter ControlBlockProperties
		8 – 255	Reserved
		0 – 255	Reserved
50	Response Parameter Error Faulty ARBlockRes	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		8	Error in Parameter ResponderUDPRTPort
		9 – 255	Reserved
51	Response Parameter Error Faulty IOCRBlockRes	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		6	Error in Parameter FrameID

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
		7 – 255	Reserved
52	Response Parameter Error Faulty AlarmCRBlockRes	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		6	Error in Parameter MaxAlarmDataLength
		7 – 255	Reserved
53	Response Parameter Error Faulty ModuleDiffBlock	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		13	Error in Parameter SubmoduleState
		14 – 255	Reserved
54	Response Parameter Error Faulty ARRPCBlockRes	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		4	Error in Parameter ResponderRPCServerPort
		5 – 255	Reserved
55	Response Parameter Error Faulty ARServerBlockRes	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		4	Error in Parameter CMResponderStationName
		5 – 255	Reserved
56	Response Parameter Error Faulty ARVendorBlockRes	0	Error in Parameter BlockType
		1	Error in Parameter Length
		...	...
		5	Error in Parameter API
		6	Error in Parameter Data*
		7 – 255	Reserved
57 – 59	Reserved	0 – 255	Reserved

**Table 481 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO
(part 2 – alarm acknowledge)**

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
60	AlarmAck Error Codes	0	Alarm Type Not Supported
		1	Wrong Submodule State
		2	IOCARSR Backup - Alarm not executed
		3 – 255	Reserved

**Table 482 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO
(part 3 – machines)**

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
65	ALPMI	0	Invalid state
		1	Wrong ACK-PDU
		2	Invalid
		3	Wrong state
		4 – 255	Reserved
66	ALPMR	0	Invalid state
		1	Wrong Notification PDU
		2	Invalid
		3	Wrong state
		4 – 255	Reserved
67	LMPM	0 – 255	Usage see protocol machines and stored as LogBook entries
68	MAC	0 – 255	Usage see protocol machines and stored as LogBook entries
69	RPC	0 – 255	see Table 486
70	APMR	0	Invalid state
		1	LMPM signaled an error
		2 – 255	Reserved
71	APMS	0	Invalid state
		1	LMPM signaled an error
		2	Timeout
		3 – 255	Reserved
72	CPM	0	Invalid state
		1 – 255	Reserved
73	PPM	0	Invalid state
		1 – 255	Reserved
74	Used by DCPUCS	0	Invalid state
		1	LMPM signaled an error
		2	Timeout
		3 – 255	Reserved
75	Used by DCPUCR	0	Invalid state
		1	LMPM signaled an error
		2 – 255	Reserved
76	Used by DCPMCS	0	Invalid state
		1	LMPM signaled an error
		2 – 255	Reserved
77	Used by DCPMCR	0	Invalid state
		1	LMPM signaled an error
		2 – 255	Reserved
78	FSPM	0 – 255	Usage see protocol machines and stored as LogBook entries

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
79 – 99	Reserved	0 – 255	Reserved
121 – 199	Reserved	0 – 255	Reserved
221 – 252	Reserved	0 – 255	Reserved

**Table 483 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO
(part 4 – IO controller)**

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
62	CMCTL	0	State conflict
		1	Timeout
		2	No data send
		3	Out of resource
		4 – 255	Reserved
63	CTLDINA (successor of the NRPM)	0	No DCP active / No Link
		1	DNS unknown RealStationName
		2	DCP no RealStationName
		3	DCP multiple RealStationName
		4	DCP no StationName
		5	No IP address
		6	DCP set error
		7 – 255	Reserved
100	CTLSTM	0	Invalid state
		1	CTLSTM signaled an error
		2 – 255	Reserved
101	CTLRDI	0	Invalid state
		1	CTLRDI signaled an error
		2 – 255	Reserved
102	CTLRDR	0	Invalid state
		1	CTLRDR signaled an error
		2 – 255	Reserved
103	CTLWRI	0	Invalid state
		1	CTLWRI signaled an error
		2 – 255	Reserved
104	CTLWRR	0	Invalid state
		1	CTLWRR signaled an error
		2 – 255	Reserved
105	CTLIO	0	Invalid state
		1	CTLIO signaled an error
		2 – 255	Reserved
106	CTLSU	0	Invalid state
		1	AR add provider or consumer failed

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
		2	AR alarm-open failed
		3	AR alarm-ack-send
		4	AR alarm-send
		5	AR alarm-ind
		6 – 255	Reserved
107	CTLRPC	0	Invalid state
		1	CTLRPC signaled an error
		2 – 255	Reserved
108	CTLPBE	0	Invalid state
		1	CTLPBE signaled an error
		2 – 255	Reserved
109 – 120	Reserved	0 – 255	Reserved

**Table 484 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO
(part 5 – IO device)**

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
61	CMDEV	0	State conflict
		1	Resource
		2 – 255	Usage see protocol machines and stored as LogBook entries
64	CMRPC (successor of the RMPM)	0	ArgsLength invalid
		1	Unknown blocks
		2	IOCR missing
		3	Wrong block count (e.g. wrong AlarmCRBlock count)
		4	Out of AR resources
		5	AR UUID unknown
		6	State conflict
		7	Out of Provider, Consumer, or Alarm Resources
		8	Out of memory
		9	Pdev already owned
		10	ARset State conflict during connection establishment
		11	ARset Parameter conflict during connection establishment
		12 – 255	Reserved
200	CMSM	0	Invalid state
		1	CMSM signaled an error
		2 – 255	Reserved
201	Reserved	0 – 255	Reserved

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
202	CMRDR	0	Invalid state
		1	CMRDR signaled an error
		2 – 255	Reserved
203	Reserved	0 – 255	Reserved
204	CMWRR	0	Invalid state
		1	AR is not in state Primary. In this case the write of a record is not allowed.
		2	CMWRR signaled an error
		3 – 255	Reserved
205	CMIO	0	Invalid state
		1	CMIO signaled an error
		2 – 255	Reserved
206	CMSU	0	Invalid state
		1	AR add provider or consumer failed
		2	AR alarm-open failed
		3	AR alarm-send
		4	AR alarm-ack-send
		5	AR alarm-ind
		6 – 255	Reserved
207	Reserved	0 – 255	Reserved
208	CMINA	0	Invalid state
		1	CMINA signaled an error
		2 – 255	Reserved
209	CMPBE	0	Invalid state
		1	CMPBE signaled an error
		2 – 255	Reserved
210	CMDMC	0	Invalid state
		1	CMDMC signaled an error
		2 – 255	Reserved
211 – 220	Reserved	0 – 255	Reserved

Table 485 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO (part 6 – abort reasons)

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
253	Used by RTA for protocol error (RTA_ERR_CLS_PROTOCOL)	0	Reserved
		1	Error within the coordination of sequence numbers (RTA_ERR_CODE_SEQ) error
		2	Instance closed (RTA_ERR_ABORT)

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
		3	AR out of memory (RTA_ERR_ABORT)
		4	AR add provider or consumer failed (RTA_ERR_ABORT)
		5	AR consumer DHT expired (RTA_ERR_ABORT)
		6	AR cmi timeout (RTA_ERR_ABORT)
		7	AR alarm-open failed (RTA_ERR_ABORT)
		8	AR alarm-send.cnf(-) (RTA_ERR_ABORT)
		9	AR alarm-ack-send.cnf(-) (RTA_ERR_ABORT)
		10	AR alarm data too long (RTA_ERR_ABORT)
		11	AR alarm.ind(err) (RTA_ERR_ABORT)
		12	AR rpc-client call.cnf(-) (RTA_ERR_ABORT)
		13	AR abort.req (RTA_ERR_ABORT)
		14	AR re-run aborts existing (RTA_ERR_ABORT)
		15	AR release.ind received (RTA_ERR_ABORT)
		16	AR device deactivated (RTA_ERR_ABORT)
		17	AR removed (RTA_ERR_ABORT)
		18	AR protocol violation (RTA_ERR_ABORT)
		19	AR name resolution error (RTA_ERR_ABORT)
		20	AR RPC-Bind error (RTA_ERR_ABORT)
		21	AR RPC-Connect error (RTA_ERR_ABORT)
		22	AR RPC-Read error (RTA_ERR_ABORT)
		23	AR RPC-Write error (RTA_ERR_ABORT)

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
		24	AR RPC-Control error (RTA_ERR_ABORT)
		25	AR forbidden pull or plug after check.rsp and before in-data.ind (RTA_ERR_ABORT)
		26	AR AP removed (RTA_ERR_ABORT)
		27	AR link down (RTA_ERR_ABORT)
		28	AR could not register multicast-mac address (RTA_ERR_ABORT)
		29	Not synchronized (cannot start companion-ar) (RTA_ERR_ABORT)
		30	Wrong topology (cannot start companion-ar) (RTA_ERR_ABORT)
		31	DCP, station-name changed (RTA_ERR_ABORT)
		32	DCP, reset to factory or factory reset (RTA_ERR_ABORT)
		33	Can't start companion-AR because a 0xLIPP submodule in the first AR... (RTA_ERR_ABORT)
		34	No IRDATA record yet (RTA_ERR_ABORT)
		35	Pdev, owner is leaving (RTA_ERROR_ABORT)
		36	Pdev, no port offers required speed/duplexity (RTA_ERROR_ABORT)
		37	IP-Suite [of the IOC] changed by means of DCP_set(IPParameter) or local engineering (RTA_ERROR_ABORT)
		38	IOCARSR RDHT expired (RTA_ERROR_ABORT)
		39	IOCARSR Pdev, parametrization impossible (RTA_ERROR_ABORT)
		40	Remote application ready timeout expired (RTA_ERROR_ABORT)

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
		41	IOCARSR Redundant interface lost or access to the peripherals impossible (RTA_ERROR_ABORT)
		42 – 200	Reserved
		201 – 255	Manufacturer specific
254	Reserved	0 – 255	Reserved
255	User specific	0 – 254	User specific
		255	Recommended for “User abort” without further detail

Table 486 – Values of ErrorCode2 for ErrorCode1 = RPC

Value (decimal)	Definition	Meaning
0	Reserved	—
1	CLRPC_ERR_REJECTED	Endpoint mapper or server did reject the call. For further details see Table 255 and Table 256.
2	CLRPC_ERR_FAULTED	Server had a fault while executing the call. For further details see Table 255 and Table 256.
3	CLRPC_ERR_TIMEOUT	Endpoint mapper or server did not respond
4	CLRPC_ERR_IN_ARGS	Broadcast or maybe “ndr_data” too large
5	CLRPC_ERR_OUT_ARGS	Server sent back more than “alloc_len”
6	CLRPC_ERR_DECODE	Result of endpoint mapper “lookup” could not be decoded
7	CLRPC_ERR_PNIO_OUT_ARGS	Out-args not “PN IO signature”, too short or inconsistent
8	CLRPC_ERR_PNIO_APP_TIMEOUT	RPC call was terminated after RPC application timeout
9 to 255	Reserved	—

5.2.6.4.4 ErrorDecode value ManufacturerSpecific

The ErrorDecode value ManufacturerSpecific indicates that the parameter ErrorCode1 and ErrorCode2 shall be set according to Table 487 and Table 488.

Table 487 – Coding of ErrorCode1 for ErrorDecode with the value ManufacturerSpecific

ErrorCode1 (hexadecimal)	Meaning	Usage
0x00 – 0xFF	Manufacturer specific	—

Table 488 – Coding of ErrorCode2 for ErrorDecode with the value ManufacturerSpecific

ErrorCode2 (hexadecimal)	Meaning	Usage
0x00 – 0xFF	Manufacturer specific	—

5.2.7 Coding section related to I&M Records

5.2.7.1 General

All data with type VisibleString shall be left justified. If the text is shorter than the defined string length, the gap shall be filled with blanks.

5.2.7.2 Coding of the field IM_Serial_Number

This field shall be coded as data type VisibleString[16]. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 16 octets.

It uniquely identifies one produced entity and is thus an extension of VendorID, DeviceID, ModuleIdentNumber, and SubmoduleIdentNumber.

5.2.7.3 Coding of the field IM_Hardware_Revision

This field shall be coded as data type Unsigned16 with the values according to Table 489.

Table 489 – IM_Hardware_Revision

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Hardware revision

5.2.7.4 Coding of the field IM_SWRevision_Functional_Enhancement

This field shall be coded as data type Unsigned8 with the values according to Table 490.

Table 490 – IM_SWRevision_Functional_Enhancement

Value (hexadecimal)	Meaning
0x00 – 0xFF	Functional Enhancement

5.2.7.5 Coding of the field IM_SWRevision_Bug_Fix

This field shall be coded as data type Unsigned8 with the values according to Table 491.

Table 491 – IM_SWRevision_Bug_Fix

Value (hexadecimal)	Meaning
0x00 – 0xFF	Bug Fix

5.2.7.6 Coding of the field IM_SWRevision_Internal_Change

This field shall be coded as data type Unsigned8 with the values according to Table 492.

Table 492 – IM_SWRevision_Internal_Change

Value (hexadecimal)	Meaning
0x00 – 0xFF	Internal Change

5.2.7.7 Coding of the field IM_Revision_Counter

This field shall be coded as data type Unsigned16 with the values according to Table 493.

Table 493 – IM_Revision_Counter

Value (hexadecimal)	Meaning	Usage
0x0000	Revision Counter	Mandatory
0x0001 – 0xFFFF	Revision Counter	Optional

5.2.7.8 Coding of the field IM_Profile_ID

This field, defined as an administrative number, shall be coded as data type Unsigned16 with the values according to Table 494.

Table 494 – IM_Profile_ID

Value (hexadecimal)	Meaning
0x0000	Non profile device
0x0001 – 0xF6FF	Administrative number
0xF700 – 0xFFFE	Reserved for profiles
0xFFFF	Reserved

NOTE 1 Depends on 5.2.3.1.

NOTE 2 In the future this standard may extend this field to Unsigned32.

5.2.7.9 Coding of the field IM_Profile_Specific_Type

This field, defined as an administrative number, shall be coded as data type Unsigned16 with the values according to Table 495 and Table 496.

Table 495 – IM_Profile_Specific_Type in conjunction with IM_Profile_ID := 0x0000

Value (hexadecimal)	Meaning
0x0000	Unspecified
0x0001	Standard Controller, e.g. Programmable Logic Controller
0x0002	PC-based Station
0x0003	IO-Module
0x0004	Communication Module
0x0005	Interface Module
0x0006	Active Network Infrastructure Component
other	—

Table 496 – IM_Profile_Specific_Type in conjunction with IM_Profile_ID range 0x0001 – 0xF6FF

Value (hexadecimal)	Meaning
0x0000	Unspecified, if not defined otherwise by the profile
0x0001 – 0xFFFF	Using shall be defined by profiles and will be treated as a subversion of the profile

5.2.7.10 Coding of the field IM_Version_Major

This field shall be coded as data type Unsigned8 with the values according to Table 497.

Table 497 – IM_Version_Major

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Shall set in this version
0x02 – 0xFF	Reserved

5.2.7.11 Coding of the field IM_Version_Minor

This field shall be coded as data type Unsigned8 with the values according to Table 498.

Table 498 – IM_Version_Minor

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	Shall set in this version
0x02 – 0xFF	Reserved

5.2.7.12 Coding of the field IM_Supported

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0: IM_Supported.Profile_specific

This field should be set to zero.

NOTE An application profile can define a different behavior.

Bit 1: IM_Supported.I&M1

This field shall be set according to Table 499.

Table 499 – IM_Supported.I&M1

Value (hexadecimal)	Meaning
0x00	Not supported
0x01	The record allows read and write access.

A write access may be denied by the application.

Bit 2: IM_Supported.I&M2

This field shall be set according to Table 499.

Bit 3: IM_Supported.I&M3

This field shall be set according to Table 499.

Bit 4: IM_Supported.I&M4

This field shall be set according to Table 499.

Bit 5: IM_Supported.I&M5

This field shall be set according to Table 499.

Bit 6: IM_Supported.I&M6

This field shall be set according to Table 499.

Bit 7: IM_Supported.I&M7

This field shall be set according to Table 499.

Bit 8: IM_Supported.I&M8

This field shall be set according to Table 499.

Bit 9: IM_Supported.I&M9

This field shall be set according to Table 499.

Bit 10: IM_Supported.I&M10

This field shall be set according to Table 499.

Bit 11: IM_Supported.I&M11

This field shall be set according to Table 499.

Bit 12: IM_Supported.I&M12

This field shall be set according to Table 499.

Bit 13: IM_Supported.I&M13

This field shall be set according to Table 499.

Bit 14: IM_Supported.I&M14

This field shall be set according to Table 499.

Bit 15: IM_Supported.I&M15

This field shall be set according to Table 499.

5.2.7.13 Coding of the field IM_Tag_Function

This field shall be coded as data type VisibleString[32]. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 32 octets.

NOTE The VisibleString[32] filled with the character blank ' ' is the reset to factory value.

5.2.7.14 Coding of the field IM_Tag_Location

This field shall be coded as data type VisibleString[22]. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 22 octets.

NOTE The VisibleString[22] filled with the character blank ' ' is the reset to factory value.

5.2.7.15 Coding of the field IM_Date

This field shall be coded as data type VisibleString[16] with the values according to Table 500.

The string format "YYYY-MM-DD'T'HH:MM'Z'" or "YYYY-MM-DD'T'HH:MM'±HH:MM'" is in accordance with ISO 8601.

The letter 'T' indicates, that a time value follows after the date and the letter 'Z' indicates, that the time value is a UTC time.

For legacy reasons the 'T' is replaced by " " and the time zone information 'Z' or '±HH:MM' is discarded.

Table 500 – IM_Date

Octet	Meaning	Usage
0 – 3	YYYY	Year with four digits
4	"_"	Character dash '-' as separator
5 – 6	MM	Month with two digits
7	"_"	Character dash '-' as separator
8 – 9	DD	Day with two digits
10	" "	Character blank ' ' as separator
11 – 12	HH	Hour with two digits
13	":"	Character colon ':' as separator
14 – 15	MM	Minute with two digits

NOTE The VisibleString[16] filled with the character blank ' ' is the reset to factory value.

5.2.7.16 Coding of the field IM_Descriptor

This field shall be coded as data type VisibleString[54]. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 54 octets.

NOTE The VisibleString[54] filled with the character blank ' ' is the reset to factory value.

5.2.7.17 Coding of the field IM_Signature

This field shall be coded as data type OctetString[54]. The value shall be set manufacturer specific and shall be filled with zero if it is shorter than 54 octets.

NOTE The OctetString[54] filled with zero or an application default is the reset to factory value.

5.2.8 Coding section related to Alarm and Diagnosis PDUs

5.2.8.1 Coding of the field UserStructureIdentifier

This field shall be coded as data type Unsigned16 with the values according to Table 501. This field identifies the structure of the field Data of the AlarmNotification and structure of the field Data of the alarm data.

Table 501 – UserStructureIdentifier

Value (hexadecimal)	Meaning	Usage
0x0000 – 0x7FFF	ManufacturerSpecific	ManufacturerSpecific Diagnosis shall be used in conjunction with following AlarmTypes: All Diagnosis ASE attached AlarmTypes ManufacturerSpecific Diagnosis shall be used in conjunction with following records: All records containing Diagnosis Data In conjunction with other alarm types the usage is manufacturer specific.
0x8000	ChannelDiagnosis	ChannelDiagnosis shall be used in conjunction with following AlarmTypes: All Diagnosis ASE attached AlarmTypes ChannelDiagnosis shall be used in conjunction with following Records: All records containing Diagnosis Data
0x8001	Multiple	Reserved
0x8002	ExtChannelDiagnosis	ExtChannelDiagnosis shall be used in conjunction with following AlarmTypes: All Diagnosis ASE attached AlarmTypes ExtChannelDiagnosis shall be used in conjunction with following Records: All records containing Diagnosis Data
0x8003	QualifiedChannel-Diagnosis	QualifiedChannelDiagnosis shall be used in conjunction with following AlarmTypes: All Diagnosis ASE attached AlarmTypes QualifiedChannelDiagnosis shall be used in conjunction with following Records: All records containing Diagnosis Data
0x8004 – 0x80FF	Reserved	—
0x8100	Maintenance	Maintenance shall be used in conjunction with following AlarmTypes: All Diagnosis ASE attached AlarmTypes Furthermore, the AlarmNotification shall only convey this block if the alarm source contains at least one Maintenance Required entry, or one Maintenance Demanded entry, or one Qualifier_x entry. Otherwise the block shall be omitted.
0x8101 – 0x81FF	Reserved	—
0x8200	Upload&Retrieval	Upload&Retrieval shall be used in conjunction with upload and retrieval notification.
0x8201	iParameter	iParameter shall be used in conjunction with upload and retrieval notification.
0x8202 – 0x8FFF	Reserved	—

Value (hexadecimal)	Meaning	Usage
0x8300	Sequence of events	Sequence of events shall be used in conjunction with following AlarmTypes: Process alarm
0x8301 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	—
0xA000 – 0xFFFF	Reserved	—

5.2.8.2 Coding of the field ChannelErrorType

This field shall be coded as data type Unsigned16 with the values according to Table 502.

Table 502 – ChannelErrorType

Value (hexadecimal)	Meaning	Assigned text
0x0000	Reserved	Unknown error
0x0001	Short circuit	Short circuit
0x0002	Undervoltage	Undervoltage
0x0003	Overvoltage	Overvoltage
0x0004	Overload	Overload
0x0005	Overtemperature	Overtemperature
0x0006	Line break	Line break
0x0007	Upper limit value exceeded	Upper limit value exceeded
0x0008	Lower limit value exceeded	Lower limit value exceeded
0x0009	Error	Error
0x000A	Simulation active	Simulation active
0x000B	Reserved	Unknown error
0x000C	Reserved	Unknown error
0x000D	Reserved	Unknown error
0x000E	Reserved	Unknown error
0x000F	Default for “parameter missing” ^b	The channel needs (additional) parameters. No or too less parameters are written
0x0010	Default for “parameterization fault” ^b	Parameterization fault. Wrong or too many parameters are written
0x0011	Default for “power supply fault” ^b	Power supply fault
0x0012	Default for “fuse blown / open” ^b	Fuse blown / open
0x0013	Default for “communication fault” ^b	Communication fault. Sequence number wrong / sequence wrong
0x0014	Default for “ground fault” ^b	Ground fault
0x0015	Default for “reference point lost” ^b	Reference point lost
0x0016	Default for “process event lost / sampling error” ^b	Process event lost / sampling error
0x0017	Default for “threshold warning” ^b	Threshold warning
0x0018	Default for “output disabled” ^b	Output disabled
0x0019	Default for “safety event” ^b	Safety event
0x001A	Default for “external fault” ^b	External fault

Value (hexadecimal)	Meaning	Assigned text
0x001B	Manufacturer specific	Manufacturer specific
0x001C	Manufacturer specific	Manufacturer specific
0x001D	Manufacturer specific	Manufacturer specific
0x001E	Manufacturer specific	Manufacturer specific
0x001F	Default for “temporary fault” ^b	Temporary fault
0x0020 – 0x00FF	Reserved for common profiles ^a	Central administrative number to unambiguously distinguish between common profiles
0x0100 – 0x7FFF	Manufacturer specific	Manufacturer specific
0x8000	Data transmission impossible	Data Transmission Impossible
0x8001	Remote mismatch	Remote Mismatch
0x8002	Media redundancy mismatch	Media Redundancy Mismatch
0x8003	Sync mismatch	Sync Mismatch
0x8004	IsochronousMode mismatch	IsochronousMode Mismatch
0x8005	Multicast CR mismatch	Multicast CR Mismatch
0x8006	Reserved	Reserved
0x8007	Fiber optic mismatch	Information for fiber optic links
0x8008	Network component function mismatch	Network functionality problems occur
0x8009	Time mismatch	Time master not existent or precision problems
0x800A	Dynamic frame packing function mismatch	DFP problems occur
0x800B	Media redundancy with planned duplication mismatch	MRPD problems occur
0x800C	System redundancy mismatch	System Redundancy Mismatch
0x800D	Multiple interface mismatch	Information about multiple interface problems
0x800E	Nested diagnosis indication	Controller to controller communication Used for propagation of diagnosis in hierarchical structures
0x800F	Reserved	Unknown error
0x8010 – 0x8FFF	Reserved	Unknown error
0x9000 – 0x9FFF	Reserved for profiles ^a	Profile specific
0xA000 – 0xFFFF	Reserved	Unknown error
^a Central administrative number to unambiguously distinguish between profiles		
^b For legacy devices “Manufacturer specific”		

5.2.8.3 Coding of the field ChannelNumber

This field shall be coded as data type Unsigned16 with the values according to Table 503.

Table 503 – ChannelNumber

Value (hexadecimal)	Meaning
0x0000 – 0x7FFF	Manufacturer specific

Value (hexadecimal)	Meaning
0x8000	Submodule Only one coding for ChannelProperties.Direction shall be used.
0x8001 – 0xFFFF	Reserved

NOTE The ChannelNumber 0x8000 references the submodule itself.

A distinct channel is addressed by the ChannelNumber in addition with the ChannelProperties.Direction.

In dependence with the field ChannelProperties.Accumulative the following meaning shall be applied:

- ChannelProperties.Accumulative=1 then the field ChannelNumber shall contain the number of the smallest channel of the effected group
- ChannelProperties.Accumulative=0 then the field ChannelNumber shall contain the number of the effected channel

5.2.8.4 Coding of the field ChannelProperties

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 7: ChannelProperties.Type

This field shall be set according to Table 504.

Table 504 – ChannelProperties.Type

Value (hexadecimal)	Meaning	Usage
0x00	—	Shall be used if the field ChannelNumber contains the value 0x8000 (submodule) Furthermore, it shall be used if none of the below defined types are appropriate.
0x01	1 Bit	The data length of this channel is 1 Bit.
0x02	2 Bit	The data length of this channel is 2 Bit.
0x03	4 Bit	The data length of this channel is 4 Bit.
0x04	8 Bit	The data length of this channel is 8 Bit.
0x05	16 Bit	The data length of this channel is 16 Bit.
0x06	32 Bit	The data length of this channel is 32 Bit.
0x07	64 Bit	The data length of this channel is 64 Bit.
0x08 – 0xFF	Reserved	—

Bit 8: ChannelProperties.Accumulative

This field shall be set to one if it is an accumulative diagnosis from more than one channel. Otherwise it shall be set to zero.

Bit 9 – 10: ChannelProperties.Maintenance

This field shall be set according to Table 505, Table 506 and Table 507.

The dependencies with elements of the field ChannelProperties are specified in Table 506 and the dependencies with Alarmnotification and RecordDataRead(DiagnosisData) in Table 507.

Table 505 – ChannelProperties.Maintenance

MaintenanceRequired Bit 9	MaintenanceDemanded Bit 10:	Usage
0x00	0x00	Diagnosis
0x01	0x00	Maintenance required
0x00	0x01	Maintenance demanded
0x01	0x01	QualifiedChannelQualifier ^a

^a This indicates that the severity is coded in the field QualifiedChannelQualifier (see Figure 83). All values of the ChannelProperties.Maintenance could be used in combination with Qualified Diagnosis.

Table 506 – Valid combinations within ChannelProperties

Maintenance	Specifier	Meaning	Valid with
Diagnosis	0x00	All subsequent ^a Diagnosis, MaintenanceRequired, MaintenanceDemanded and QualifiedDiagnosis disappear	ChannelDiagnosis
	0x01	Diagnosis appears	
	0x02	Diagnosis disappears	ChannelDiagnosis, ManufacturerSpecificDiagnosis, ExtChannelDiagnosis
	0x03	Diagnosis disappears but other remain	
Maintenance required	0x00	Reserved	
	0x01	MaintenanceRequired appears	
	0x02	MaintenanceRequired disappears	ChannelDiagnosis, ManufacturerSpecificDiagnosis, ExtChannelDiagnosis
	0x03	MaintenanceRequired disappears but other remain	
Maintenance demanded	0x00	Reserved	
	0x01	MaintenanceDemanded appears	
	0x02	MaintenanceDemanded disappears	ChannelDiagnosis, ManufacturerSpecificDiagnosis, ExtChannelDiagnosis
	0x03	MaintenanceDemanded disappears but other remain	
Qualified-Channel-Qualifier	0x00	Reserved	
	0x01	QualifiedDiagnosisQualifier appears	
	0x02	QualifiedDiagnosisQualifier disappears	QualifiedChannelDiagnosis
	0x03	QualifiedDiagnosisQualifier disappears but other remain	

^a Subsequent means, all ExtChannelErrorTypes for the delivered ChannelErrorType and also the ChannelErrorType itself disappears. Shall be used in AlarmNotification only.

Table 507 – Valid combinations for Alarmnotification and Record-DataRead(DiagnosisData)

Maintenance	Specifier	Meaning	Valid with
Diagnosis	0x00	All subsequent ^a Diagnosis, MaintenanceRequired, MaintenanceDemanded and QualifiedDiagnosis disappear	Alarmnotification
	0x01	Diagnosis appears	Alarmnotification and Record-DataRead(DiagnosisData)
	0x02	Diagnosis disappears	Alarmnotification RecordDataRead(DiagnosisData) shall only be used in conjunction with ManufacturerSpecific-Diagnosis
	0x03	Diagnosis disappears but other remain	Alarmnotification
Maintenance required	0x00	Reserved	—
	0x01	MaintenanceRequired appears	Alarmnotification and Record-DataRead(DiagnosisData)
	0x02	MaintenanceRequired disappears	Alarmnotification
	0x03	MaintenanceRequired disappears but other remain	Alarmnotification
Maintenance demanded	0x00	Reserved	—
	0x01	MaintenanceDemanded appears	Alarmnotification and Record-DataRead(DiagnosisData)
	0x02	MaintenanceDemanded disappears	Alarmnotification
	0x03	MaintenanceDemanded disappears but other remain	Alarmnotification
Qualified diagnosis	0x00	Reserved	—
	0x01	QualifiedDiagnosis appears	Alarmnotification and Record-DataRead(DiagnosisData)
	0x02	QualifiedDiagnosis disappears	Alarmnotification
	0x03	QualifiedDiagnosis disappears but other remain	Alarmnotification
^a Subsequent means, all ExtChannelErrorTypes for the delivered ChannelErrorType and also the ChannelErrorType itself disappears.			

Bit 11 – 12: ChannelProperties.Specifier

This field shall be coded with the values according to Table 508. The field ChannelProperties.Specifier is related to the addressed channel and severity (diagnosis, maintenance or qualified). The dependencies with elements of the field ChannelProperties are specified in Table 506.

Table 508 – ChannelProperties.Specifier

Value (hexadecimal)	Meaning	Usage
0x00	All subsequent disappears	See Table 506 The Diagnosis ASE contains no longer any entries (of any severity) for this channel
0x01	Appears	An event appears and/or exists further The Diagnosis ASE contains this and possible other entries for this channel.
0x02	Disappears	An event disappears and/or exists no longer The Diagnosis ASE contains no longer any entries of the same severity for this channel
0x03	Disappears but other remain	An event disappears The Diagnosis ASE still contains other entries of the same severity for this channel

NOTE The conveyed data of the AlarmNotification depends on the creation time and can differ from the actual content of the Diagnosis ASE. The AlarmNotification is used to convey the changes and the Diagnosis ASE to contain the actual status.

Bit 13 – 15: ChannelProperties.Direction

This field shall be set according to Table 509.

Table 509 – ChannelProperties.Direction

Value (hexadecimal)	Meaning
0x00	Manufacturer specific
0x01	Input
0x02	Output
0x03	Input/Output
0x04 – 0x07	Reserved

5.2.8.5 Coding of the field ExtChannelErrorType

This field shall be coded as data type Unsigned16. The value of this field depends on the field ChannelErrorType. The values shall be set according to Table 510.

Table 510 – ExtChannelErrorType

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Definition is depending on the ChannelErrorType (see tables below).

In conjunction with defined ChannelErrorTypes the ExtChannelErrorType shall be set according to Table 512, Table 513, Table 514, Table 515, Table 516, Table 517, Table 518, Table 519, Table 520, Table 521, Table 522, Table 523, Table 524, Table 525, and Table 526.

Additional an ExtChannelErrorType value range is bound to the type of the ChannelErrorType according to Table 511.

Table 511 – Allowed combinations of ChannelErrorType, ExtChannelErrorType, and ExtChannelAddValue

ChannelErrorType	ExtChannelErrorType	ExtChannelAddValue	Usage
Normative	Normative	Normative	Allowed
Normative	Vendor specific	Vendor specific	Allowed
Normative	Profile specific	Profile specific	Allowed
Profile specific	Profile specific	Profile specific	Allowed
Profile specific	Vendor specific	Vendor specific	Allowed
Profile specific	Normative	Normative	Prohibited
Vendor specific	Vendor specific	Vendor specific	Allowed
Vendor specific	Normative	Normative	Prohibited
Vendor specific	Profile specific	Profile specific	Prohibited

Table 512 – ExtChannelErrorType for ChannelErrorType 0 – 0xFF

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Accumulative Info	Alarm/diagnosis
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 513 – ExtChannelErrorType for ChannelErrorType 0x0100 – 0x7FFF

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Accumulative Info	Alarm/diagnosis
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 514 – ExtChannelErrorType for ChannelErrorType “Data transmission impossible”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Link State mismatch – Link down	Alarm/diagnosis
0x8001	MAUType mismatch	Alarm/diagnosis
0x8002	Line Delay mismatch	Alarm/diagnosis
0x8003 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis

Value (hexadecimal)	Meaning	Usage
0xA000 – 0xFFFF	Reserved	—

Table 515 – ExtChannelErrorType for ChannelErrorType “Remote mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Peer name of station mismatch	Alarm/diagnosis
0x8001	Peer name of port mismatch	Alarm/diagnosis
0x8002	Peer RT_CLASS_3 mismatch ^a	Alarm/diagnosis
0x8003	Peer MAUType mismatch	Alarm/diagnosis
0x8004	Peer MRP domain mismatch	Alarm/diagnosis
0x8005	No peer detected	Alarm/diagnosis
0x8006	Reserved	—
0x8007	Peer Line Delay mismatch	Alarm/diagnosis
0x8008	Peer PTCP mismatch ^b	Alarm/diagnosis
0x8009	Peer Preamble Length mismatch	Alarm/diagnosis
0x800A	Peer Fragmentation mismatch	Alarm/diagnosis
0x800B – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—
^a This events occurs, if the remote RTClass3_PortStatus.State is not RTCLASS3_RUN. This could also happen, if the remote IRDATAUUID is not equal to the local IRDATAUUID. ^b This events occurs, if the remote PTCP_SubdomainUUID is equal to the local PTCP_SubdomainUUID and the remote PTCP_MasterSourceAddress is not equal to the local PTCP_MasterSourceAddress.		

Table 516 – ExtChannelErrorType for ChannelErrorType “Media redundancy mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Manager role fail	Alarm/diagnosis
0x8001	MRP ring open	Alarm/diagnosis
0x8002	Reserved	—
0x8003	Multiple manager	Alarm/diagnosis
0x8004 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

The source of a “Media redundancy mismatch” is always a port. If two ports are affected, the one with the lesser number shall be used.

Table 517 – ExtChannelErrorType for ChannelErrorType “Sync mismatch” and for ChannelErrorType “Time mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	No sync message received	Alarm/diagnosis
0x8001 – 0x8002	Reserved	—
0x8003	Jitter out of boundary	Alarm/diagnosis
0x8004 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 518 – ExtChannelErrorType for ChannelErrorType “Isochronous mode mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Output Time Failure – Output update missing or out of order	Alarm/diagnosis
0x8001	Input Time Failure	Alarm/diagnosis
0x8002	Master Life Sign Failure – Error in MLS update detected	Alarm/diagnosis
0x8003 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 519 – ExtChannelErrorType for ChannelErrorType “Multicast CR mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Multicast Consumer CR timed out	Alarm/diagnosis
0x8001	Address resolution failed	Alarm/diagnosis
0x8002 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 520 – ExtChannelErrorType for ChannelErrorType “Fiber optic mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Power Budget	Alarm/diagnosis

Value (hexadecimal)	Meaning	Usage
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 521 – ExtChannelErrorType for ChannelErrorType “Network component function mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Frame dropped – no resource	Alarm/diagnosis
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 522 – ExtChannelErrorType for ChannelErrorType “Dynamic Frame Packing function mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000 – 0x80FF	Reserved	—
0x8100	Frame late error for FrameID “0x0100”	Alarm/diagnosis
0x8101 + 0x8FFE	See Formula (55)	Alarm/diagnosis
0x8FFF	Frame late error for FrameID “0x0FFF”	Alarm/diagnosis
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

The source of a “Dynamic Frame Packing function mismatch” is the interface submodule using the DFP packing function.

The used ExtChannelErrorType shall be calculated according Formula (55).

$$\text{ExtChannelErrorType} = 0x8000 + \text{FrameID} \quad (55)$$

where

ExtChannelErrorType	is the ExtChannelErrorType used for the signaling of an event
0x8000	is the base value of the ChannelNumber
FrameID	is the FrameID of the frame issuing the event

Table 523 – ExtChannelErrorType for ChannelErrorType “Media redundancy with planned duplication mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000 – 0x86FF	Reserved	—
0x8700	MRPD duplication void for FrameID “0x0700”	Alarm/diagnosis
0x8701 + 0x8FFE	See Formula (56)	Alarm/diagnosis
0x8FFF	MRPD duplication void for FrameID “0x0FFF”	Alarm/diagnosis
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

The source of a “Media redundancy with planned duplication mismatch” is the interface submodule using the MRPD function.

NOTE Searching for the smallest SlotNumber and then for the smallest SubslotNumber of this slot identifies the least SlotNumber/SubslotNumber combination.

The used ExtChannelErrorType shall be calculated according Formula (56).

$$\text{ExtChannelErrorType} = 0x8000 + \text{FrameID} \quad (56)$$

where

ExtChannelErrorType is the ExtChannelErrorType used for the signaling of an event
 0x8000 is the base value of the ChannelNumber
 FrameID is the FrameID of the frame issuing the event

Table 524 – ExtChannelErrorType for ChannelErrorType “System redundancy mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	System redundancy event	Alarm/diagnosis
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

The source of a “System redundancy mismatch” is the submodule with the lowest API/SlotNumber/SubslotNumber combination, which is part of the AR.

Table 525 – ExtChannelErrorType for ChannelErrorType “Multiple interface mismatch”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	StandardGateway mismatch	Alarm/diagnosis
0x8001	NameOfStation of the interface is not unambiguous	Alarm/diagnosis
0x8002	IPAddress range of the interface is not unambiguous	Alarm/diagnosis
0x8003	Conflicting MultipleInterfaceMode.NameOfDevice mode	Alarm/diagnosis
0x8004 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

Table 526 – ExtChannelErrorType for ChannelErrorType “Nested diagnosis indication”

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved	—
0x0001 – 0x7FFF	Manufacturer specific	Alarm/diagnosis
0x8000	Subordinated entity information	Alarm/diagnosis
0x8001 – 0x8FFF	Reserved	—
0x9000 – 0x9FFF	Reserved for profiles	Alarm/diagnosis
0xA000 – 0xFFFF	Reserved	—

5.2.8.6 Coding of the field ExtChannelAddValue

This field shall be coded as data type Unsigned32 with the values according to Table 527.

Table 527 – Values for ExtChannelAddValue

Value (hexadecimal)	Meaning
0x00000000	No additional information
0x00000001 – 0xFFFFFFFF	Definition depending on ChannelError and ExtChannelErrorType

5.2.8.6.1 Coding of the field ExtChannelAddValue for ChannelErrorType = 0 – 0x7FFF

This field shall be coded with the values according to Table 528 if the field ExtChannelErrorType contains the value 0x8000 and the field ChannelErrorType contains the value 0 – 0x7FFF.

Table 528 – Values for Accumulative Info

Bitposition	Value (hexadecimal)	Meaning
Bit 0	0x00	ChannelNumber is not effected
	0x01	ChannelNumber is effected
Bit 1	0x00	ChannelNumber + 1 is not effected
	0x01	ChannelNumber + 1 is effected
Bit 2	0x00	ChannelNumber + 2 is not effected
	0x01	ChannelNumber + 2 is effected
...	...	...
Bit30	0x00	ChannelNumber + 30 is not effected
	0x01	ChannelNumber + 30 is effected
Bit31	0x00	ChannelNumber + 31 is not effected
	0x01	ChannelNumber + 31 is effected

5.2.8.6.2 Coding of the field ExtChannelAddValue for ChannelErrorType “Fiber optic mismatch”

This field shall be coded with the values according to Table 529 if the field ExtChannelErrorType contains the value 0x8000 and the field ChannelErrorType contains the value 0x8007.

Table 529 – Values for “Fiber optic mismatch” – “Power Budget”

Value (hexadecimal)	Meaning
0x00 – 0x3E7	PowerBudget in 0,1 dB steps [0..99,9 dB]
0x03E8 – 0xFFFFFFFF	Reserved
0xFFFFFFFF	Unknown

5.2.8.6.3 Coding of the field ExtChannelAddValue for ChannelErrorType “Network component function mismatch”

This field shall be coded with the values according to Table 530 if the field ExtChannelErrorType contains the value 0x8000 and the field ChannelErrorType contains the value 0x8008.

Table 530 – Values for “Network component function mismatch” – “Frame dropped”

Value (hexadecimal)	Meaning
0x00000000 – 0x00FFFFFF	Number of dropped frames in case of no resource
0x01000000 – 0xFFFFFFFF	Reserved

5.2.8.6.4 Coding of the field ExtChannelAddValue for ChannelErrorType “Remote mismatch”

This field shall be coded with the values according to Table 531 if the field ExtChannelErrorType contains the value 0x8007 and the field ChannelErrorType contains the value 0x8001.

Table 531 – Values for “Remote mismatch” – “Peer CableDelay mismatch”

Value (hexadecimal)	Meaning	Usage
0x00 – 0x32	Error of measurement	No signaling
0x33 – 0x3B9ACA00	CableDelay difference between peers	Signal deviation
0x3B9ACA01 – 0xFFFFFFFF	Reserved	—

5.2.8.6.5 Coding of the field ExtChannelAddValue for ChannelErrorType “Multiple interface mismatch”

This field shall be coded with the values according to Table 532 if the field ExtChannelErrorType contains the value 0x8003 and the field ChannelErrorType contains the value 0x800D.

Table 532 – Values for “Multiple interface mismatch” – “Conflicting MultipleInterfaceMode.NameOfDevice mode”

Value (hexadecimal)	Meaning	Usage
0x00	Reserved	—
0x01	MultipleInterfaceMode.NameOfDevice for this interface is set to zero but one other interface is set to one.	Inconsistency
0x02	MultipleInterfaceMode.NameOfDevice for this interface is set to one but one other interface is set to zero.	Inconsistency
other	Reserved	—

5.2.8.7 Coding of the field QualifiedChannelQualifier

This field shall be coded as data type Unsigned32. The values shall be set according to Table 533. There shall only one bit be set in one QualifiedChannelDiagnosis.

Table 533 – Values for QualifiedChannelQualifier

Bitposition	Value (hexadecimal)	Meaning	Usage
Bit 0	Reserved	—	—
Bit 1	Reserved	—	—
Bit 2	0x00	Qualifier_2 not set	Profile specific
	0x01	Qualifier_2 is set	
Bit 3	0x00	Qualifier_3 not set	
	0x01	Qualifier_3 is set	
...	...	...	
Bit 30	0x00	Qualifier_30 not set	
	0x01	Qualifier_30 is set	
Bit 31	0x00	Qualifier_31 not set	
	0x01	Qualifier_31 is set	

5.2.8.8 Coding of the field MaintenanceStatus

This field shall be coded as data type Unsigned32. At least one of the bits below shall be set to one in order to convey this alarm block according to Table 534. Otherwise this block shall be omitted.

Table 534 – Values for MaintenanceStatus

Bitposition	Bit name	Value (hexadecimal)	Meaning
Bit 0	MaintenanceRequired	0x00	No maintenance required information available
		0x01	Maintenance required information available
Bit 1	MaintenanceDemanded	0x00	No maintenance demanded information available
		0x01	Maintenance demanded information available
Bit 2	Qualifier_2	0x00	No information available
		0x01	Information available
Bit 3	Qualifier_3	0x00	No information available
		0x01	Information available
...	—	—	...
Bit 30	Qualifier_30	0x00	No information available
		0x01	Information available
Bit 31	Qualifier_31	0x00	No information available
		0x01	Information available

The classification of diagnosis, maintenance and qualified according to their severity is shown in Figure 83.

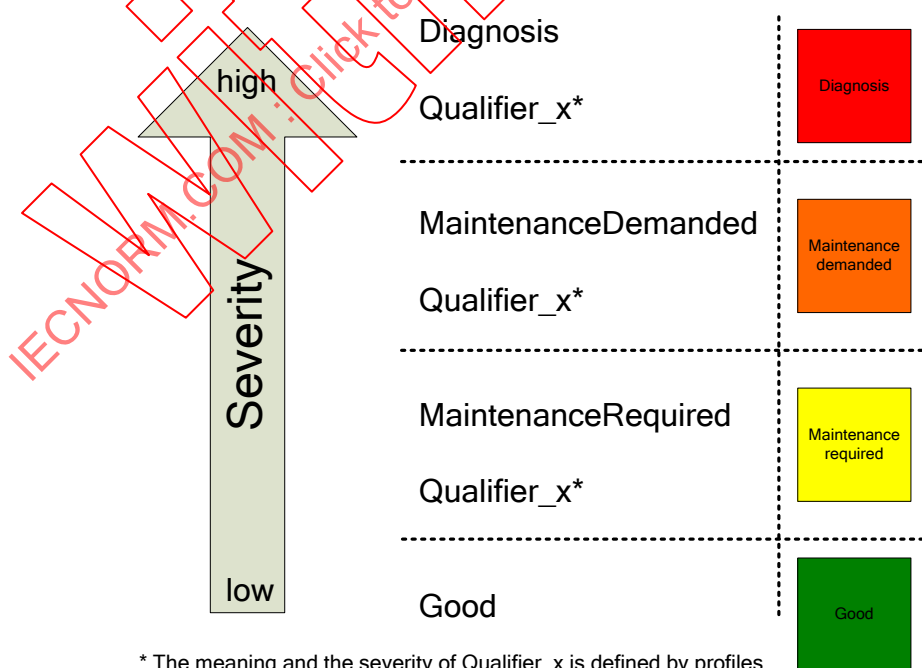


Figure 83 – Severity classification of diagnosis, maintenance and qualified

The meaning and the severity of Qualifier_x shall be defined by a profile.

5.2.9 Coding section related to upload and retrieval

5.2.9.1 Coding of the field URRecordIndex

This field shall be coded as data type Unsigned32 with the values according to Table 535.

Table 535 – URRecordIndex

Value (hexadecimal)	Meaning
0x00000000 – 0x0000FFFF	Index of the used record
0x00010000 – 0xFFFFFFFF	Reserved

5.2.9.2 Coding of the field URRecordLength

This field shall be coded as data type Unsigned32 with the values according to Table 536.

Table 536 – URRecordLength

Value (hexadecimal)	Meaning
0x00000000	Write stored records to the IO device.
0x00000001 – 0xFFFFFFFF	Write stored records to the IO device, if the length of each selected record is less or equal URRecordLength.

5.2.10 Coding section related to iParameter

5.2.10.1 Coding of the field iPar_Req_Header

This field shall be coded as data type Unsigned32 with the values according to IEC 61784-3-3.

5.2.10.2 Coding of the field Max_Segm_Size

This field shall be coded as data type Unsigned32 with the values according to IEC 61784-3-3.

5.2.10.3 Coding of the field Transfer_Index

This field shall be coded as data type Unsigned32 with the values according to IEC 61784-3-3.

5.2.10.4 Coding of the field Total_iPar_Size

This field shall be coded as data type Unsigned32 with the values according to IEC 61784-3-3.

5.2.11 Coding section related to Physical Device Interface Data

5.2.11.1 Coding of the field MultipleInterfaceMode

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0: MultipleInterfaceMode.NameOfDevice

This field shall be set according to Table 537.

Table 537 – MultipleInterfaceMode.NameOfDevice

Value (hexadecimal)	Meaning
0x00	Default The field LLDPChassis contains the NameOfStation and the field PortID of LLDP contains the name of the port.
0x01	The field LLDPChassis contains the NameOfDevice and is defined by local means. The field PortID of LLDP contains the name of the port and the NameOfStation.

Bit 1 – 15: MultipleInterfaceMode.reserved_1

This field shall be set to zero.

Bit 16 – 31: MultipleInterfaceMode.reserved_2

This field shall be set according to 3.4.2.2.

5.2.12 Coding section related to Physical Device Port Data**5.2.12.1 Coding of the field OwnPortName**

This field shall be coded as data type OctetString[] according to 4.3.1.4.16.

5.2.12.2 Coding of the field LengthOwnPortName

This field shall be coded as data type Unsigned8 according to 4.3.1.4.16.

5.2.12.3 Coding of the field NumberOfPeers

This field shall be coded as data type Unsigned8.

5.2.12.4 Coding of the field LengthPeerPortName

This field shall be coded as data type Unsigned8.

5.2.12.5 Coding of the field PeerPortName**5.2.12.5.1 Usage in conjunction with PDPortDataCheck.CheckPeers**

This field shall be coded according to 4.3.1.4.16.

5.2.12.5.2 Usage in conjunction with PDPortDataReal

This field shall be coded according to 4.3.1.4.16. It shall contain the NameOfPort part of the received LLDP_PortID, if coded according to 4.3.1.4.16. Otherwise the LengthPeerPortName shall be zero.

5.2.12.6 Coding of the field LengthPeerStationName

This field shall be coded as data type Unsigned8.

5.2.12.7 Coding of the field PeerStationName**5.2.12.7.1 General**

This field shall be coded as data type OctetString[] with 1 to 255 octets.

5.2.12.7.2 Usage in conjunction with PDPortDataCheck.CheckPeers

This field shall be coded according to 4.3.1.4.15.

5.2.12.7.3 Usage in conjunction with PDPortDataReal

This field shall be coded according to 4.3.1.4.15. It shall contain the NameOfStation part of the received LLDP_PortID or LLDP_ChassisID, if coded according to 4.3.1.4.15. Otherwise the LengthPeerStationName shall be zero.

5.2.12.8 Coding of the field LengthOwnStationName

This field shall be coded as data type Unsigned8 according to 4.3.1.4.15.

5.2.12.9 Coding of the field OwnStationName

This field shall be coded as data type OctetString[] according to 4.3.1.4.15.

5.2.12.10 Coding of the field LineDelay

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 30: LineDelay.Value

This field shall be set according to Table 538, Table 539, Figure 27 and Formula (26).

Table 538 – LineDelay.Value with LineDelay.FormatIndicator == 0

Value (hexadecimal)	Meaning
0x00000000	Line delay and cable delay unknown
0x00000001 – 0x7FFFFFFF	Line delay in nanoseconds

Table 539 – LineDelay.Value with LineDelay.FormatIndicator == 1

Value (hexadecimal)	Meaning
0x00000000	Reserved
0x00000001 – 0x7FFFFFFF	Cable delay in nanoseconds

Bit 31: LineDelay.FormatIndicator

This field shall be set according to Table 540.

Table 540 – LineDelay.FormatIndicator

Value (hexadecimal)	Meaning	Usage
0x00	LineDelay.Value is coded as line delay	Default, if line delay or cable delay is unknown Shall be used in conjunction with PDPortDataCheck
0x01	LineDelay.Value is coded as cable delay	Default, if cable delay is known Shall be used in conjunction with PDPortDataReal

5.2.12.11 Coding of the field PeerMACAddress

This field shall be coded as data type OctetString[6]. The value of the field PeerMACAddress shall be according to the 48-bit universal LAN MAC addresses of IEEE 802.

NOTE Octet 1 contains the Individual/Group Address Bit (Isb).

5.2.13 Coding section related to Physical Device IR Data

5.2.13.1 Coding of the field RxPort

This field shall be coded as data type Unsigned8. This field shall be coded with the values according to Table 541.

Table 541 – RxPort

Value (hexadecimal)	Meaning
0x00	Local interface
0x01	Port 1
0x02	Port 2
...	...
0xFF	Port 255

5.2.13.2 Coding of the field NumberOfTxPortGroups

This field shall be coded as data type Unsigned8 and shall be set according to Table 542. This field shall only count the succeeding TxPortGroupArray entries.

Table 542 – NumberOfTxPortGroups

Value (hexadecimal)	Meaning
0x01, 0x03, 0x05, 0x07, 0x09, 0x0A, 0x0C, 0x0F	Allowed values
0x11, 0x13, 0x15, 0x17, 0x19, 0x1A, 0x1C, 0x1F	Allowed values
0x21	Allowed values
other	Reserved

5.2.13.3 Coding of the field TxPortGroupArray

The field TxPortGroupArray is an array of octets that shall contain at least one and at most 33 octets referred to as TxPortGroup octet. A TxPortGroup octet shall consist of at least one and at most 8 TxPort entries referred to as TxPortGroup entry 0 to TxPortGroup entry 7. Therefore, the number of TxPortGroup octet corresponds to the number of ports within a device and shall be calculated as follows

$$N = M_{\text{highest}} \text{ DIV } 8 + 1 \quad (57)$$

where

N	is the number of TxPortGroup octets or the number of array elements
M_{highest}	is the highest number of TxPorts within a device (maximum 255)
8	is the bit size of a octet

$$G = m \text{ DIV } 8 + 1 \quad (58)$$

where

G	is the number of the TxPortGroup octet containing the TxPort
m	is the number of the current TxPort with $1 \leq m \leq M$
8	is the bit size of a octet

$$B = m \text{ MOD } 8 \quad (59)$$

where

B	is the number of the bit in the TxPortGroup octet containing the TxPort
m	is the number of the current TxPort with $1 \leq m \leq M$
8	is the bit size of a octet

NOTE The term DIV stands for division without remainder. The term MOD stands for the remainder of the division.

The last TxPortGroup octet (octet N) may not contain all 8 TxPort entries. If $M_{\text{highest}} \text{ MOD } 8 \neq 7$ the octet N is not fully filled and the remaining bits shall be set to zero referred to as Padding Bits.

The TxPortGroup of the devices TxPorts shall be structured in ascending order without gaps. The coding of this field shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0: TxPortEntry_0

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \text{ MOD } 8 = 0$ is present. A padding bit shall be used if no TxPort is present.

Table 543 – TxPortEntry

Value (hexadecimal)	Meaning
0x00	Transmission off
0x01	Transmission on

The TxPort of local injection is always placed in TxPortEntry_0 of the TxPortGroup octet number one.

Bit 1: TxPortEntry_1

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \text{ MOD } 8 = 1$ is present. A padding bit shall be used if no TxPort is present.

Bit 2: TxPortEntry_2

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \text{ MOD } 8 = 2$ is present. A padding bit shall be used if no TxPort is present.

Bit 3: TxPortEntry_3

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \text{ MOD } 8 = 3$ is present. A padding bit shall be used if no TxPort is present.

Bit 4: TxPortEntry_4

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \text{ MOD } 8 = 4$ is present. A padding bit shall be used if no TxPort is present.

Bit 5: TxPortEntry_5

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \bmod 8 = 5$ is present. A padding bit shall be used if no TxPort is present.

Bit 6: TxPortEntry_6

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \bmod 8 = 6$ is present. A padding bit shall be used if no TxPort is present.

Bit 7: TxPortEntry_7

This bit shall be set with the values according to Table 543 if the TxPort with the number that meets $m \bmod 8 = 7$ is present. A padding bit shall be used if no TxPort is present.

5.2.13.4 Coding of the field FrameDetails

This field shall be coded according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0 – 1: FrameDetails.SyncFrame

This field shall be coded with the values according to Table 544 and Table 545.

Table 544 – FrameDetails.SyncFrame in conjunction with FrameDataProperties.ForwardingMode := “Absolute mode”

Value (hexadecimal)	Meaning	Description
0x00	No sync frame	Mandatory
0x01	Primary sync frame	Legacy
0x02	Secondary sync frame	Primary and secondary sync frame use the same FrameID. At one time only one frame shall be valid.
0x03	Reserved	—

Table 545 – FrameDetails.SyncFrame in conjunction with FrameDataProperties.ForwardingMode := “Relative mode”

Value (hexadecimal)	Meaning	Description
0x00	No sync frame	Mandatory
Other	Reserved	—

Bit 2 – 3: FrameDetails.MeaningFrameSendOffset

This field shall be coded with the values according to Table 546.

Table 546 – FrameDetails.MeaningFrameSendOffset

Value (hexadecimal)	Meaning	Usage
0x00	The content of the field FrameSendOffset specifies the point of time for transmitting a frame. For the receiver the content of the field specifies the point of time for receiving a frame.	Mandatory
other	Reserved	—

Bit 4 – 6: FrameDetails.reserved

This field shall be set to zero.

Bit 7: FrameDetails.MediaRedundancyWatchDog

This field shall be set according to Table 547.

Table 547 – FrameDetails.MediaRedundancyWatchDog

Value (hexadecimal)	Meaning	Meaning
0x00	Disable	Do not monitor the quality of a MRPD connection
0x01	Enable	Monitor the quality of a MRPD connection and signal a change

5.2.13.5 Coding of the field FrameDataProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0: FrameDataProperties.ForwardingMode

This field shall be set according to Table 548.

Table 548 – FrameDataProperties.ForwardingMode

Value (hexadecimal)	Meaning
0x00	Absolute mode The FrameSendOffset contains the sending time for a forwarded frame
0x01	Relative mode The FrameSendOffset shall contain the sending time of a forwarded frame or the "Best effort" value.

Bit 1 – 2: FrameDataProperties.FastForwardingMulticastMACAdd

This field shall be set according to Table 549.

Table 549 – FrameDataProperties.FastForwardingMulticastMACAdd

Value (hexadecimal)	Meaning
0x00	Legacy Use interface MAC destination unicast address
0x01	Use RT_CLASS_3 destination multicast address
0x02	Use FastForwardingMulticastMACAdd
0x03	Reserved

Bit 3 – 4: FrameDataProperties.FragmentationMode

This field shall be coded with the values according to Table 550.

Table 550 – FrameDataProperties.FragmentationMode

Value (hexadecimal)	Meaning	Usage
0x00	No fragmentation	Mandatory
0x01	Fragmentation enabled Maximum fragment size for static fragmentation 128 Bytes	Optional
0x02	Fragmentation enabled Maximum fragment size for static fragmentation 256 Bytes	Optional
0x03	Reserved	—

Bit 5 – 15: FrameDataProperties.reserved_1

This field shall be set to zero.

Bit 16 – 31: FrameDataProperties.reserved_2

This field shall be set according to 3.4.2.2.

5.2.13.6 Coding of the field AdjustProperties

This field shall be coded as data type Unsigned16 with the value zero.

5.2.13.7 Coding of the field MAUType

This field shall be coded as data type Unsigned16 with the values according to RFC 4836, Table 551 and Table 552.

Table 551 – MAUType

Value (hexadecimal)	Meaning	Usage
0x0000	Reserved for MediaType “Copper cable” and “Fiber optic cable”	Default for LinkState.Link != “UP”
	Used for MediaType “Radio communication”	Default for radio communication
0x0005	10BaseT	PDPortDataReal
0x000A	10BaseTXHD	PDPortDataReal
0x000B	10BaseTXFD	PDPortDataReal
0x000C	10BaseFLHD	PDPortDataReal
0x000D	10BaseFLFD	PDPortDataReal
0x000F	100BaseTXHD	PDPortDataReal
0x0010	Default (MediaType Copper) 100BaseTXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0011	100BaseFXHD	PDPortDataReal
0x0012	Default (MediaType Fiber optic) 100BaseFXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0015	1000BaseXHD	PDPortDataReal
0x0016	1000BaseXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0017	1000BaseLXHD	PDPortDataReal
0x0018	1000BaseLXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0019	1000BaseSXHD	PDPortDataReal
0x001A	1000BaseSXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x001D	1000BaseTHD	PDPortDataReal
0x001E	1000BaseTFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x001F	10GbaseFX	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0020	10GbaseLX4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0021	10GbaseR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck

Value (hexadecimal)	Meaning	Usage
0x0022	10GbaseER	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0023	10GbaseLR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0024	10GbaseSR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0025	10GbaseW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0026	10GbaseEW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0027	10GbaseLW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0028	10GbaseSW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0029	10GbaseCX4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002A	2BaseTL	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002B	10PassTS	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002C	100BaseBX10D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002D	100BaseBX10U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002E	100BaseLX10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002F	1000BaseBX10D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0030	1000BaseBX10U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0031	1000BaseLX10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0032	1000BasePX10D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0033	1000BasePX10U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0034	1000BasePX20D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0035	1000BasePX20U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0036	10GBaseT or 100BasePXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0037	10GBaseLRM	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0038	1000BaseKX	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0039	1000BaseKX4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003A	1000BaseKR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003B	10G1GBasePRXD1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck

Value (hexadecimal)	Meaning	Usage
0x003C	10G1GBasePRXD2	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003D	10G1GBasePRXD3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003E	10G1GBasePRXU1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003F	10G1GBasePRXU2	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0040	10G1GBasePRXU3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0041	10GBasePRD1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0042	10GBasePRD2	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0043	10GBasePRD3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0044	10GBasePRU1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0045	10GBasePRU3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0046	40GbaseKR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0047	40GbaseCR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0048	40GbaseSR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0049	40GbaseFR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004A	40GbaseLR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004B	100GbaseCR10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004C	100GbaseSR10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004D	100GbaseLR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004E	100GbaseER4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
IANA dependent	100BasePXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
Useable well known numbers	Speed of 100 Mbit/s (and more) and duplexity full	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
Unuseable well known numbers	Speed less than 100 Mbit/s or duplexity half	PDPortDataReal
other	Reserved	—

NOTE The MAUType 100BasePXFD has two codings due to legacy reasons.

Table 552 – Valid combinations between MAUType and LinkState

LinkState.Port	LinkState.Link	MAUType	Usage
Unknown Disabled / Discarding Blocking Listening Learning Forwarding Broken	Up (ready to pass packets)	10BaseT	PDPortDataReal
		10BaseTXHD	
		10BaseTXFD	
		10BaseFLHD	
		10BaseFLFD	
		100BaseTXHD	
		100BaseTXFD (Default)	
		100BaseFXHD	
		100BaseFXFD	
		1000BaseXHD	
		1000BaseXFD	
		1000BaseLXHD	
		1000BaseLXFD	
		1000BaseSXHD	
		1000BaseSXFD	
		1000BaseTHD	
		1000BaseTFD	
		10GigBaseFX	
		100BaseLX10	
		100BasePXFD	
Reserved	Down	Useable well known numbers	Reserved
		Unuseable well known numbers	
		Testing (in some test mode)	
		Unknown (status cannot determined)	
Reserved	Reserved		

5.2.13.8 Coding of the field CheckSyncMode

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0: CheckSyncMode.CableDelay

This field shall be set according to Table 553.

Table 553 – CheckSyncMode.CableDelay

Value (hexadecimal)	Meaning	Usage
0x00	OFF	No check
0x01	ON	Check cable delay difference between local and remote measured cable delay versus 50 ns.

Bit 1: CheckSyncMode.SyncMaster

This field shall be set according to Table 554.

Table 554 – CheckSyncMode.SyncMaster

Value (hexadecimal)	Meaning	Usage
0x00	OFF	No check
0x01	ON	Check PTCP_MasterSourceAddress between local and remote using LLDP_PNIO_PTCPSTATUS.

Bit 2 – 15: CheckSyncMode.reserved

This field shall be set according to 3.4.2.2.

5.2.13.9 Coding of the field MAUTypeMode

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0: MAUTypeMode.Check

This field shall be set according to Table 555.

Table 555 – MAUTypeMode.Check

Value (hexadecimal)	Meaning	Usage
0x00	OFF	No check
0x01	ON	Check MAU type difference between local and remote detected value.

Bit 1 – 15: MAUTypeMode.reserved

This field shall be set according to 3.4.2.2.

5.2.13.10 Coding of the field DomainBoundaryIngress

The coding of this field shall be according to 3.4.2.5 and the individual bits shall be coded with the values according to Table 556.

Table 556 – DomainBoundaryIngress

Bit	Value	Meaning
0	1	Block an incoming frame with the multicast MAC address 01-0E-CF-00-04-20, 01-0E-CF-00-04-40 and 01-0E-CF-00-04-80
	0	Do not block an incoming frame with the multicast MAC address 01-0E-CF-00-04-20, 01-0E-CF-00-04-40 and 01-0E-CF-00-04-80
1..31	reserved	

5.2.13.11 Coding of the field DomainBoundaryEgress

The coding of this field shall be according to 3.4.2.5 and the individual bits shall be coded with the values according to Table 557.

Table 557 – DomainBoundaryEgress

Bit	Value	Meaning
0	1	Block an outgoing frame with the multicast MAC address 01-0E-CF-00-04-20, 01-0E-CF-00-04-40 and 01-0E-CF-00-04-80
	0	Do not block an outgoing frame with the multicast MAC address 01-0E-CF-00-04-20, 01-0E-CF-00-04-40 and 01-0E-CF-00-04-80
1..31	reserved	

For the PTCP-AnnoucePDU Table 556 and Table 557 shall be combined according to Table 558 to decode the domain boundary.

Table 558 – DomainBoundaryAnnounce

—	Domain-Boundary-Ingress	Domain-Boundary-Egress	Meaning
Bit	Value	Value	—
0	1	1	Discard frames with the multicast MAC address 01-0E-CF-00-04-00
	0	1	Do not block a frame with the multicast MAC address 01-0E-CF-00-04-00
	1	0	
	0	0	
1..31	reserved		

5.2.13.12 Coding of the field MulticastBoundary

This field shall be coded as data type Unsigned32. The individual bits shall be coded with the values according to Table 559.

Table 559 – MulticastBoundary

Bit	Value	Meaning
0	1	Block the multicast MAC address 01-0E-CF-00-02-00
	0	Do not block the multicast MAC address 01-0E-CF-00-02-00
...	1	Block the multicast MAC address 01-0E-CF-00-02-xx
	0	Do not block the multicast MAC address 01-0E-CF-00-02-xx
31	1	Block the multicast MAC address 01-0E-CF-00-02-1F
	0	Do not block the multicast MAC address 01-0E-CF-00-02-1F

This shall be applied for the first 32 RT_CLASS_2 multicast addresses from 01-0E-CF-00-02-00 to 01-0E-CF-00-02-1F.

5.2.13.13 Coding of the field PeerToPeerBoundary

This field shall be coded as data type Unsigned32. The individual bits shall be coded with the values according to Table 560.

Table 560 – PeerToPeerBoundary

Bit	Value	Meaning
0	1	The LLDP agent shall not send LLDP frames (egress filter).
	0	The LLDP agent shall send LLDP frames for this port.
1	1	The PTCP ASE shall not send PTCP_DELAY request frames (egress filter).
	0	The PTCP ASE shall send PTCP_DELAY request frames for this port.
2	1	The Time ASE shall not send PATH_DELAY request frames (egress filter).
	0	The Time ASE shall send PATH_DELAY request frames for this port.
...	0	Reserved
31	0	Reserved

5.2.13.14 Coding of the field DCPBoundary

This field shall be coded as data type Unsigned32. The individual bits shall be coded with the values according to Table 561.

Table 561 – DCPBoundary

Bit	Value	Meaning
0	1	Block an outgoing DCP_Identify frame (egress filter) with the multicast MAC address 01-0E-CF-00-00-00
	0	Do not block the multicast MAC address 01-0E-CF-00-00-00
1	1	Block an outgoing DCP_Hello frame (egress filter) with the multicast MAC address 01-0E-CF-00-00-01
	0	Do not block the multicast MAC address 01-0E-CF-00-00-01
...	0	Reserved
31	0	Reserved

5.2.13.15 Coding of the field PreambleLength

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0: PreambleLength.Length

This field shall be set according to Table 562.

Table 562 – PreambleLength.Length

Value (hexadecimal)	Meaning	Usage
0x00	Seven octets Preamble shall be used	Default For the Ethernet frames the PHY shall use seven octets preamble before the start delimiter (SD)
0x01	One octet Preamble shall be used	For the Ethernet frames the PHY shall use one octet preamble before the start delimiter (SD)

Bit 1 – 15: PreambleLength.reserved

This field shall be set according to 3.4.2.2.

5.2.13.16 Coding of the field LinkState

The coding of this field shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 7: LinkState.Link

This field shall be set according to Table 563.

Table 563 – LinkState.Link

Value (hexadecimal)	Meaning	Usage
0x00	Reserved	—
0x01	Up (ready to pass packets)	PDPortDataReal, CheckLinkState
0x02	Down	PDPortDataReal, AdjustLinkState
0x03	Testing (in some test mode)	PDPortDataReal
0x04	Unknown (status cannot be determined)	PDPortDataReal
0x05	Dormant	PDPortDataReal
0x06	NotPresent	PDPortDataReal
0x07	LowerLayerDown	PDPortDataReal
0x08 – 0xFF	Reserved	—

Bit 8 – 15: LinkState.Port

This field shall be set according to Table 564.

Table 564 – LinkState.Port

Value (hexadecimal)	Meaning	Usage
0x00	Unknown	Default Mandatory PDPortDataReal
0x01	Disabled / Discarding	Optional PDPortDataReal
0x02	Blocking	Optional PDPortDataReal
0x03	Listening	Optional PDPortDataReal
0x04	Learning	Optional PDPortDataReal

Value (hexadecimal)	Meaning	Usage
0x05	Forwarding	Optional PDPortDataReal
0x06	Broken	Optional PDPortDataReal
0x07 – 0xFF	Reserved	—

5.2.13.17 Coding of the field MediaType

This field shall be coded as data type Unsigned32 with the values according to Table 565.

Table 565 – MediaType

Value (hexadecimal)	Meaning	Usage
0x00	Unknown	PDPortDataReal
0x01	Copper cable	PDPortDataReal
0x02	Fiber optic cable	PDPortDataReal
0x03	Radio communication	PDPortDataReal
0x04 – 0xFFFFFFFF	Reserved	—

5.2.13.18 Coding of the field MaxBridgeDelay

This field shall be coded as data type Unsigned32 according to Table 566 in nanoseconds. Figure 28 shows the meaning of MaxBridgeDelay.

Table 566 – MaxBridgeDelay

Value (hexadecimal)	Meaning
0x00000000	Unknown
0x00000001 – 0x0000FFFF	From engineering used bridge delay for RT_CLASS_3 calculation
0x00010000 – 0xFFFFFFFF	Reserved

5.2.13.19 Coding of the field NumberOfPorts

This field shall be coded as data type Unsigned32 with values according to Table 567.

Table 567 – NumberOfPorts

Value (hexadecimal)	Meaning
0x00000000	Reserved
0x00000001 – 0x000000FF	Number of following port entries
0x00000100 – 0xFFFFFFFF	Reserved

5.2.13.20 Coding of the field MaxPortTxDelay

This field shall be coded as data type Unsigned32 according to Table 568 in nanoseconds. Figure 28 shows the meaning of MaxPortTxDelay.

Table 568 – MaxPortTxDelay

Value (hexadecimal)	Meaning
0x00000000	Unknown
0x00000001 – 0x0000FFFF	From engineering used port transmit delay for RT_CLASS_3 calculation
0x00010000 – 0xFFFFFFFF	Reserved

5.2.13.21 Coding of the field MaxPortRxDelay

This field shall be coded as data type Unsigned32 according to Table 569 in nanoseconds. Figure 28 shows the meaning of MaxPortRxDelay.

Table 569 – MaxPortRxDelay

Value (hexadecimal)	Meaning
0x00000000	Unknown
0x00000001 – 0x0000FFFF	From engineering used port receive delay for RT_CLASS_3 calculation
0x00010000 – 0xFFFFFFFF	Reserved

5.2.13.22 Coding of the field MaxLineRxDelay

This field shall be coded as data type Unsigned32 according to Table 570 in nanoseconds. Figure 28 shows the meaning of MaxLineRxDelay.

Table 570 – MaxLineRxDelay

Value (hexadecimal)	Meaning
0x00000000	Unknown No delay, forward as soon as possible
0x00000001 – 0x3B9AC9FF	From engineering used line delay for RT_CLASS_3 calculation
0x3B9ACA00 – 0xFFFFFFFF	Reserved

5.2.13.23 Coding of the field YellowTime

This field shall be coded as data type Unsigned32 according to Table 571 in nanoseconds. Figure 84 and Figure 86 shows the meaning and Figure 85 the calculation principle of YellowTime.

The YELLOW period allows a bridge to optimize the forwarding of frames as shown in Figure 85.

If the YellowTime, calculated according Formula (60) and (61), is less than the maximum Ethernet frame, the fragmentation shall be enabled.

NOTE The YellowTime is 125 µs if running without fragmentation.

$$\text{YellowTime} = (\text{InterFrameGAP} + \text{PreambleLength} + \text{StartFrameDelimiter} + \text{MAX(Selected Protocols)}) \times 80 \text{ ns} + \text{REDBeginSafetyMargin} \quad (60)$$

where

YellowTime	is the minimum time between two reserved periods
InterFrameGAP	is the minimum distance between two frames
PreambleLength	is the length of the Preamble
StartFrameDelimiter	is the time used to transmit the SFD
MAX(Selected Protocols)	is the length of the frames of the selected protocols
REDBeginSafetyMargin	is the time which covers internal delays of a switch

$$\text{MAX(Selected Protocols)} = \text{Maximum (PTCP, MRP, RT, IEEE 802.1AS)} \quad (61)$$

where

MAX(Selected Protocols)	is the maximum frame size of the protocol list
PTCP	is the maximum size of a frame used by the PTCP protocol Fragmentable protocol elements are ignored
MRP	is the maximum size of a frame used by the MRP protocol
RT	is the maximum frame length of a RT_CLASS_4_2 frame used in this specific environment
IEEE 802.1AS	is the maximum size of a frame used by the IEEE 802.1AS protocol Fragmentable protocol elements are ignored

Table 571 – YellowTime

Value (hexadecimal)	Meaning	Usage
0x00 – 0x1ABF	Reserved	—
0x1AE0	Yellow time in Nanoseconds Equal to 86 Octets at 100 Mbit/s	Fragmentation is supported. In this case Formula (61) may return the value 64 Octets.
0x1AE1 – 0x1E847	—	Fragmentation is supported. The frame length of RT is between 65 and 1521 Octets.
0x1E848	Default Yellow time in Nanoseconds Equal to 1562,5 Octets at 100 Mbit/s	Fragmentation is <i>not</i> supported. In this case Formula (61) returns the value 1522 Octets.
0x0x1E849 – 0xFFFFFFFF	Reserved	—

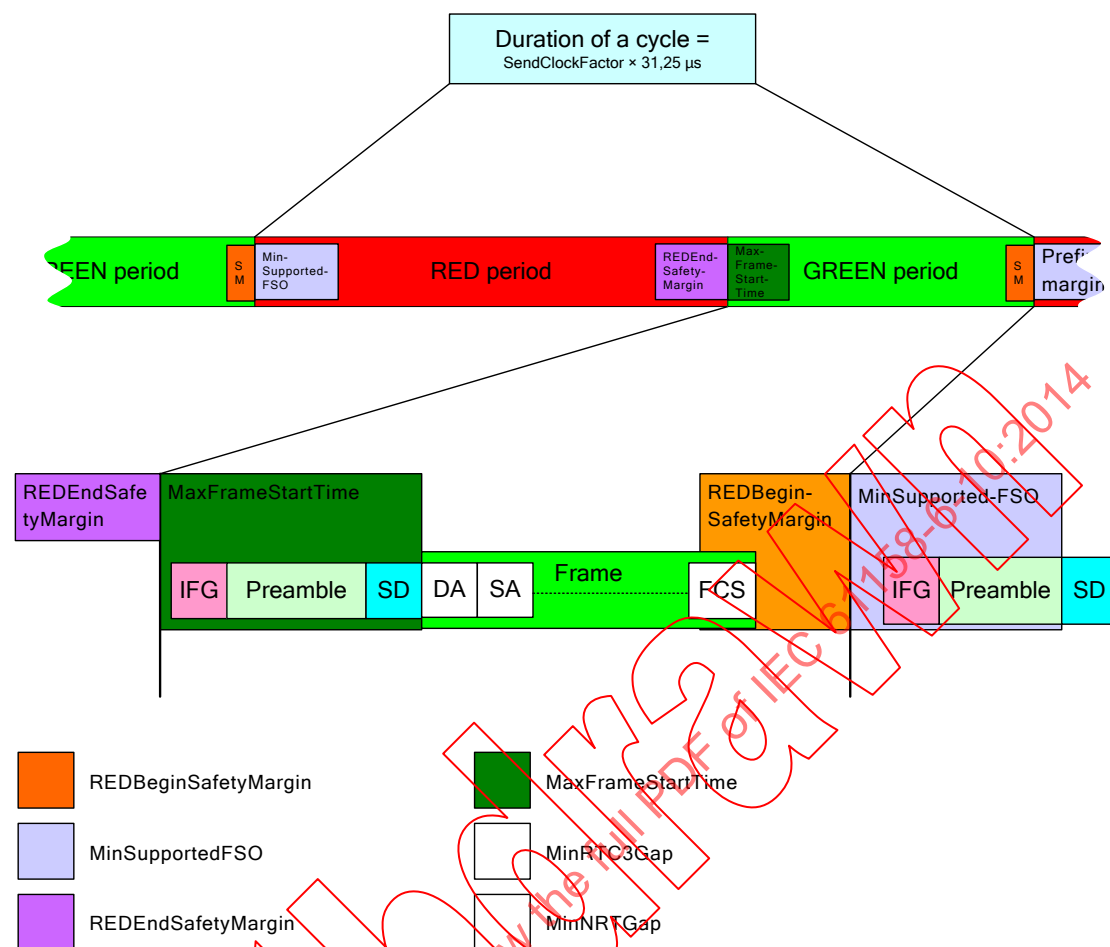


Figure 84 – Calculation principle for a cycle

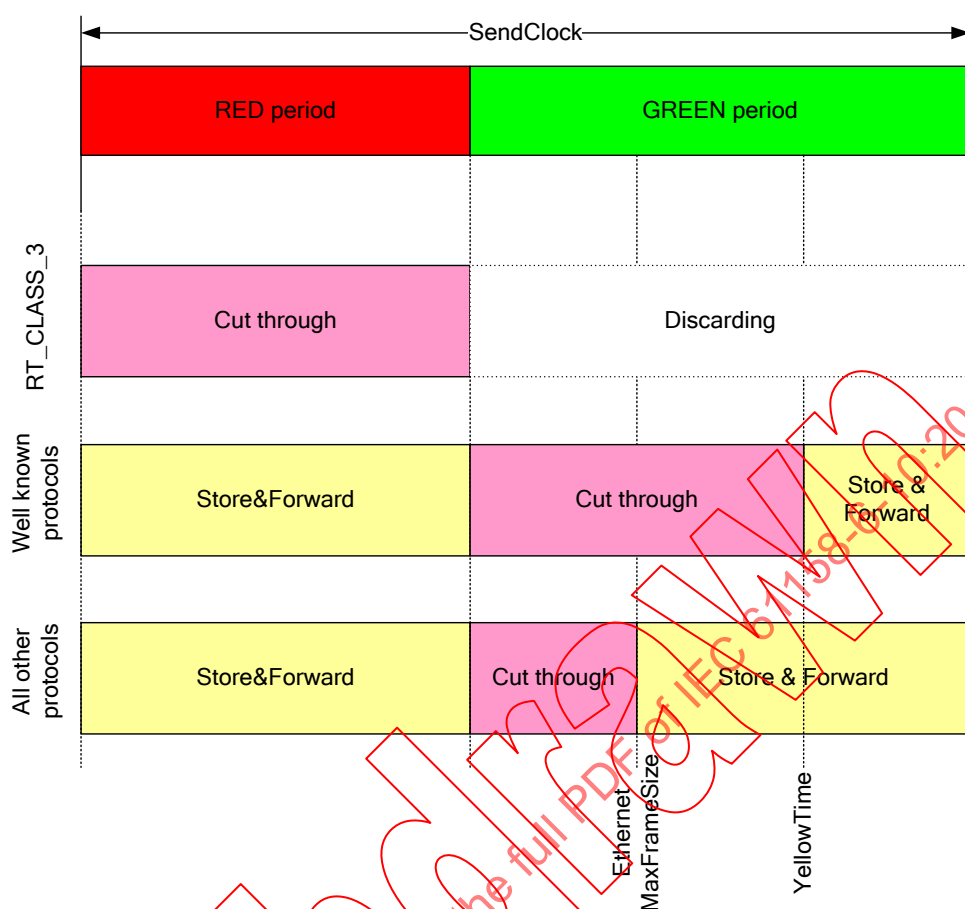


Figure 85 – Calculation principle for the minimum YellowTime

5.2.13.24 Coding of the field StartOfRedFrameID

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 572 and Table 574.

Table 572 – StartOfRedFrameID

Value (hexadecimal)	Meaning
0x0000 – 0x007F	Reserved
0x0080	Reserved Shall be accepted in case of legacy support for PTCP in RED
0x0081 – 0x00FF	Reserved
0x0100 – 0x0FFF	Mandatory 0x0100 is the default value
0x1000 – 0xFFFF	Reserved

5.2.13.25 Coding of the field EndOfRedFrameID

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 573 and Table 574.

Table 573 – EndOfRedFrameID

Value (hexadecimal)	Meaning
0x0000 – 0x00FF	Reserved
0x0100 – 0x0FFF	Mandatory 0x0FFF is the default value
0x1000 – 0xFFFF	Reserved

Table 574 – Dependencies of StartOfRedFrameID and EndOfRedFrameID

Value (hexadecimal)	Meaning
StartOfRedFrameID < EndOfRedFrameID	Valid
StartOfRedFrameID ≥ EndOfRedFrameID	Not used

5.2.13.26 Coding of the field NumberOfAssignments

This field shall be coded as data type Unsigned32. This field shall be coded with the values according to Table 575.

Table 575 – NumberOfAssignments

Value (hexadecimal)	Meaning
0x00000000	Reserved
0x00000001 – 0x00000004	Mandatory, if only phases with reserved periods (RED) exist, four entries shall be supported. Number of following assignment entries
0x00000005	Mandatory, if phases without reserved periods (RED) exist, five entries shall be supported. Number of following assignment entries
0x00000006 – 0x00000010	Optional Number of following assignment entries
0x00000011 – 0xFFFFFFFF	Reserved

If there is no reserved period for a port, than the NumberOfAssignments shall be coded with one and RedOrangePeriodBegin, OrangePeriodBegin and GreenPeriodBegin shall be coded with zero. Furthermore, the NumberOfPhases shall be coded with one and AssignedValueForReservedBegin, AssignedValueForOrangeBegin and AssignedValueForReservedEnd shall be coded with zero.

5.2.13.27 Coding of the field NumberOfPhases

This field shall be coded as data type Unsigned32. This field shall be coded with the values according to Table 576.

Table 576 – NumberOfPhases

Value (hexadecimal)	Meaning
Other	Reserved
0x00000001	Mandatory One following phase assignment entry
0x00000002	Mandatory Two following phase assignment entries
0x00000004	Mandatory Four following phase assignment entries
0x00000008	Mandatory Eight following phase assignment entries
0x00000010	Mandatory Sixteen following phase assignment entries

NOTE If more than 16 phases exist due to a ReductionRatio greater sixteen, the engineering shall logically reduce the amount of begin and end values to fit again in this structure.

5.2.13.28 Coding of the field PhaseAssignment

The field PhaseAssignment shall be according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 3: AssignedValueForReservedBegin

This bit shall be set with the values according to Table 577.

Table 577 – AssignedValueForReservedBegin

Value (hexadecimal)	Meaning
0x00	RedOrangePeriodBegin of entry one of the XXBeginEndAssignment shall be used
0x01 – 0x0E	...
0x0F	RedOrangePeriodBegin of entry sixteen of the XXBeginEndAssignment shall be used

Bit 4 – 7: AssignedValueForOrangeBegin

This bit shall be set with the values according to Table 578.

Table 578 – AssignedValueForOrangeBegin

Value (hexadecimal)	Meaning
0x00	OrangePeriodBegin of entry one of the XXBeginEndAssignment shall be used
0x01 – 0x0E	...
0x0F	OrangePeriodBegin of entry sixteen of the XXBeginEndAssignment shall be used

Bit 8 – 11: AssignedValueForReservedEnd

This bit shall be set with the values according to Table 579.

Table 579 – AssignedValueForReservedEnd

Value (hexadecimal)	Meaning
0x00	GreenPeriodBegin of entry one of the XXBeginEndAssignment shall be used
0x01 – 0x0E	...
0x0F	GreenPeriodBegin of entry sixteen of the XXBeginEndAssignment shall be used

Bit 12 – 15: Reserved

This bit shall be set to zero.

5.2.13.29 Coding of the field RedOrangePeriodBegin

This field shall be coded as data type Unsigned32 with values according to Table 580 and Table 581. The time base is 1 ns.

Table 580 – Values of RedOrangePeriodBegin

Value (hexadecimal)	Meaning	Use
0	Relative offset to the start of the related cycle	Mandatory
1 – 0x003D08FF	Relative offset to the start of the related cycle	Optional Use not intended
0x003D0900 – 0xFFFFFFFF	Reserved	—

Table 581 – Dependencies of RedOrangePeriodBegin, OrangePeriodBegin and GreenPeriodBegin

Value (hexadecimal)	Meaning	Usage
RedOrangePeriodBegin < OrangePeriodBegin < GreenPeriodBegin	Not used	—
RedOrangePeriodBegin = OrangePeriodBegin < GreenPeriodBegin	Not used	—
RedOrangePeriodBegin < OrangePeriodBegin = GreenPeriodBegin	Only a RED period is defined	Mandatory
RedOrangePeriodBegin = OrangePeriodBegin = GreenPeriodBegin = 0	No reserved period is defined	Mandatory
RedOrangePeriodBegin > OrangePeriodBegin	Not used	—
RedOrangePeriodBegin > GreenPeriodBegin	Not used	—
OrangePeriodBegin > GreenPeriodBegin	Not used	—

5.2.13.30 Coding of the field OrangePeriodBegin

This field shall be coded as data type Unsigned32 with values according to Table 581, Table 582, and Table 583. The time base is 1 ns.

Table 582 – Values of OrangePeriodBegin with ARProperties.StartupMode == Legacy

Value (hexadecimal)	Meaning	Use
0 – 0x003D08FF	Relative offset to the start of the related cycle	Optional
0x003D0900 – 0xFFFFFFFF	Reserved	—

Table 583 – Values of OrangePeriodBegin with ARProperties.StartupMode == Advanced

Value (hexadecimal)	Meaning	Use
GreenPeriodBegin	OrangePeriodBegin shall be equal to GreenPeriodBegin	—
other	Reserved	—

5.2.13.31 Coding of the field GreenPeriodBegin

This field shall be coded as data type Unsigned32 with values according to Table 581 and Table 584. The time base is 1 ns.

Table 584 – Values of GreenPeriodBegin

Value (hexadecimal)	Meaning	Use
0 – 0x0007A120	Relative offset to the start of the related cycle	Mandatory
0x0007A121 – 0x003D08FF	Relative offset to the start of the related cycle	Optional
0x003D0900 – 0xFFFFFFFF	Reserved	—

5.2.13.32 Coding of the field EtherType

This field shall be coded as Unsigned16 as described for the field LT in Table 22. This specification uses the value according to Table 585.

Table 585 – EtherType

Value (hexadecimal)	Meaning
0x8892	This standard

5.2.14 Coding section related to Physical Sync Data**5.2.14.1 Coding of the field PTCPLengthSubdomainName**

This field shall be coded as data type Unsigned8.

5.2.14.2 Coding of the field PTCPSubdomainName

This field shall be coded as data type OctetString with 1 to 240 octets according to 4.3.1.4.15.1.

NOTE The field PTCPSubdomainName is not terminated by zero.

5.2.14.3 Coding of the field SyncProperties

This field shall be coded according to 3.4.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: SyncProperties.Role

This field shall be coded with the values according to Table 586.

Table 586 – SyncProperties.Role

Value (hexadecimal)	Meaning	Usage
0x00	Reserved	—
0x01	External sync	Working Clock Slave
0x02	Internal sync	Working Clock Master
0x03	Reserved	—

Bit 2 – 7: SyncProperties.reserved

This field shall be set to zero.

Bit 8 – 12: SyncProperties.SyncID

This field shall be coded according to Table 587.

Table 587 – SyncProperties.SyncID

Value (hexadecimal)	Meaning	Usage
0x00	SyncID 0	Working Clock synchronization
other	reserved	—

Bit 13 – 15: SyncProperties.reserved

This field shall be set to zero.

5.2.14.4 Coding of the field ReservedIntervalBegin

This field shall be coded as data type Unsigned32 with values according to Table 588, Table 589, Table 592, and Figure 86. The time base is one nanosecond.

Table 588 – ReservedIntervalBegin with ARProperties.StartupMode == Legacy

Value (hexadecimal)	Meaning
0x00000000	Begin of ORANGE period
other	Reserved

Table 589 – ReservedIntervalBegin with ARProperties.StartupMode == Advanced

Value (hexadecimal)	Meaning
0x00000000	Mandatory
other	Reserved

5.2.14.5 Coding of the field ReservedIntervalEnd

This field shall be coded as data type Unsigned32 with values according to Table 590, Table 591, Table 592, and Figure 86. The time base is one nanosecond.

Table 590 – ReservedIntervalEnd with ARProperties.StartupMode == Legacy

Value (hexadecimal)	Meaning
0x00000001 – 50% SendClock	End of ORANGE period
other	Reserved

Table 591 – ReservedIntervalEnd with ARProperties.StartupMode == Advanced

Value (hexadecimal)	Meaning
0x00000000	Mandatory
other	Reserved

The dependencies between the field ReservedIntervalBegin and the field ReservedIntervalEnd are shown in Table 592.

Table 592 – Dependencies of ReservedIntervalBegin and ReservedIntervalEnd

Value (hexadecimal)	Meaning
ReservedIntervalBegin < ReservedIntervalEnd	An ORANGE period is defined
ReservedIntervalBegin = ReservedIntervalEnd = 0	Default
ReservedIntervalBegin > ReservedIntervalEnd	Not used

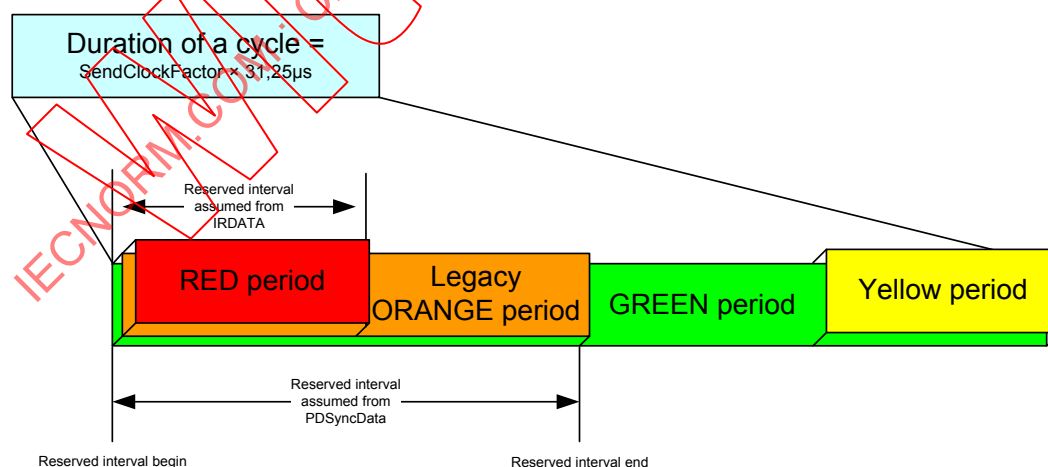


Figure 86 – Definition of the reserved interval

5.2.14.6 Coding of the field SyncSendFactor

This field shall be coded as data type Unsigned32 with values according to Table 593. The time base is 31,25 µs.

Table 593 – SyncSendFactor

Value (hexadecimal)	Meaning	SyncID (Clock)	SyncID (Time)
0x0000	Reserved	—	—
0x0001 – 0x03FF	Optional	—	—
0x03C0	Default	Mandatory (30 ms)	Optional (30 ms)
0x03C1 – 0x0C7F	Optional	—	—
0x0C80	Optional	—	Optional (100 ms)
0x0C81 – 0x176FF	Optional	—	—
0x17700	Default	—	Optional (3 s)
0x17701 – 0x4E1FF	Optional	—	—
0x4E200	Optional	—	Optional (10 s)
0x4E200 – 0xEA5FF	Optional	—	—
0xEA600	Default	—	Mandatory (30 s)
0x000EA601 – 0xA4CB7FFF	Optional	—	—
0xA4CB8000	Default	—	Optional (24 h)
0xA4CB8001 – 0xFFFFFFFF	Reserved	—	—

The SyncSendFactor may be rounded to full 10 ms before usage.

Each SyncProperties.SyncID shall require its own SyncSendFactor. The SyncSendInterval shall be calculated according to Formula (62).

$$\text{SyncSendInterval} = \text{SyncSendFactor} \times 31,25 \mu\text{s} \quad (62)$$

where

SyncSendInterval is the sync send interval

SyncSendFactor is the factor for the calculation of the sync send interval

5.2.14.7 Coding of the field PTCPTimeoutFactor

This field shall be coded as data type Unsigned16. The time base is the value of the field SyncSendFactor. The value range shall be according to Table 594.

Table 594 – PTCPTimeoutFactor

Value (hexadecimal)	Meaning
0x0000	Disabled
0x0001 – 0x0002	Optional
0x0003 – 0x0005	Mandatory
0x0006	Default, mandatory
0x0007 – 0x000F	Mandatory
0x0010 – 0x01FF	Optional
0x0200 – 0xFFFF	Reserved

Each SyncProperties.SyncID shall require its own PTCPTimeoutFactor. The Timeout shall be calculated according to Formula (63).

$$\text{Timeout} = \text{PTCPTimeoutFactor} \times \text{SyncSendFactor} \times 31,25 \mu\text{s} \quad (63)$$

where

Timeout is the time out
PTCPTimeoutFactor is the PTCP timeout factor for the calculation of the time out
SyncSendFactor is the factor for the calculation of the sync send interval

5.2.14.8 Coding of the field PTCPTakeoverTimeoutFactor

This field shall be coded as data type Unsigned16 with the values according to Table 595. The time base is the value of the field SyncSendFactor.

Table 595 – PTCPTakeoverTimeoutFactor

Value (hexadecimal)	Meaning
0x0000	Disabled
0x0001	Optional
0x0002	Mandatory Default for sync slaves
0x0003	Mandatory Default for sync masters
0x0004 – 0x000F	Mandatory
0x0010 – 0x01FF	Optional
0x0200 – 0xFFFF	Reserved

Each SyncProperties.SyncID shall require its own PTCPTakeoverTimeoutFactor. The Timeout shall be calculated according to Formula (64).

$$\text{Timeout} = \text{PTCPTakeoverTimeoutFactor} \times \text{SyncSendFactor} \times 31,25 \mu\text{s} \quad (64)$$

where

Timeout is the time out
PTCPTakeoverTimeoutFactor is the PTCP takeover factor for the calculation of the time out
SyncSendFactor is the factor for the calculation of the sync send interval
31,25 μs is the time base for the calculation

5.2.14.9 Coding of the field PTCPMasterStartupTime

This field shall be coded as data type Unsigned16 with the values according to Table 596. The time base is one second.

Table 596 – PTCPMasterStartupTime

Value (hexadecimal)	Meaning
0x0000	Disabled
0x0001 – 0x0004	Optional
0x0005 – 0x003B	Mandatory
0x003C	Default, mandatory
0x003D – 0x012C	Optional
0x012D – 0xFFFF	Reserved

Each SyncProperties.SyncID shall require its own PTCPMasterStartupTime.

5.2.14.10 Coding of the field PLLWindow

This field shall be coded as data type Unsigned32 with the values according to Table 597. The time base shall be one nanosecond.

Table 597 – PLLWindow

Value (hexadecimal)	Meaning	SyncID := 0 (Working Clock)
0x00	Disabled	—
0x0001 – 0x03E7	Optional	—
0x03E8	Default	Mandatory (1 µs)
0x03E9 – 0x270F	Optional	—
0x2710	Default	Optional (10 µs)
0x2710 – 0x000F423F	Optional	—
0x000F4240	Default	Optional (1 ms)
0x000F423F – 0x98967F	Optional	—
0x989680	Default	Optional (10 ms)
0x989681 – 0xFFFFFFFF	Reserved	—

Each SyncProperties.SyncID shall require its own PLLWindow.

The definition of the PLLWindow is shown in Figure 87 and Figure 88.

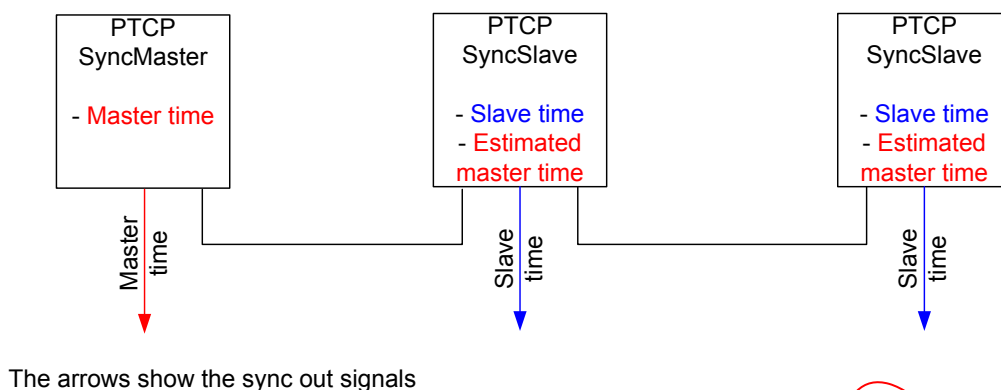


Figure 87 – Toplevel view to the PLL window

NOTE 1 The SyncSlaves estimates the master time by means of PTCP. It control its slave time to follow the master time, but in real life it follows the estimated master time. Because of the noisy environment the estimated master time is more or less equal to the master time.

NOTE 2 The external measurement of the PLLWindow is done between master time and slave time. The internal measurement of a SyncSlave is done between estimated master time and slave time.

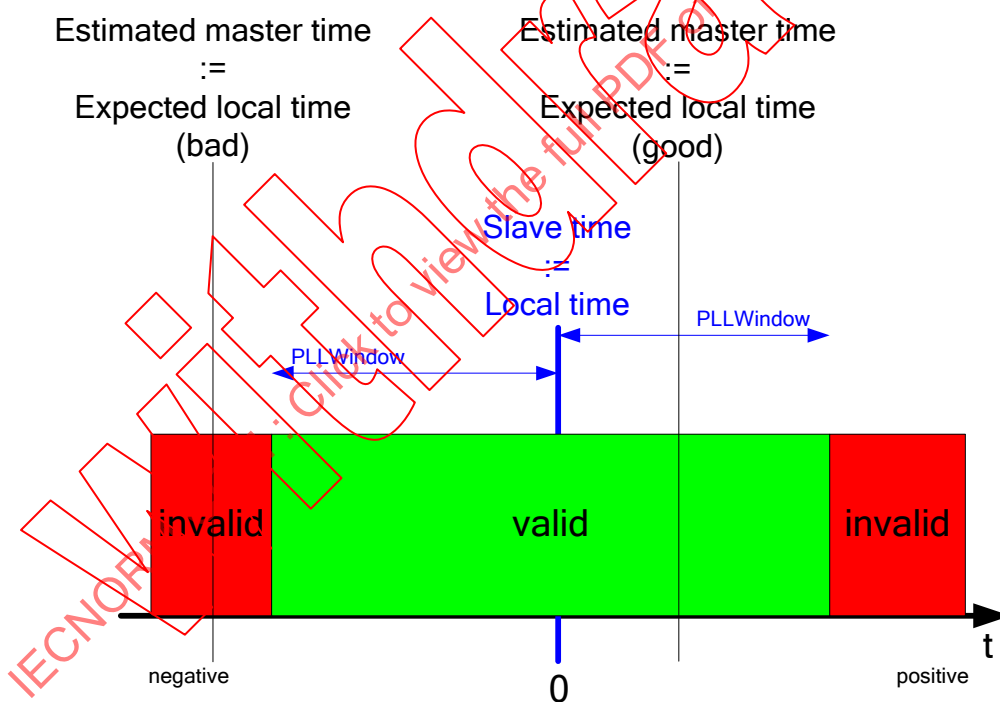


Figure 88 – Definition of PLL window

5.2.15 Coding section related to Isochrone Mode Data

5.2.15.1 Coding of the field TimeIObase

This field shall be coded as data type Unsigned32 with values according to Table 598. The time base is ns.

Table 598 – TimeIObase

Value (decimal)	Meaning
1 – 32 000 000	Optional
other	Reserved

5.2.15.2 Coding of the field TimeDataCycle

This field shall be coded as data type Unsigned16 with values according to Table 599. The time base is 31,25 μ s.

Table 599 – TimeDataCycle

Value (hexadecimal)	Meaning	Comment
0x00	Reserved	—
0x01	Optional	Minimum time of 31,25 μ s
0x02 – 0x03FF	Optional	...
0x0400	Optional	Maximum time of 32 ms.
0x401 – 0xFFFF	Reserved	—

The calculation of the TimeDC shall be done by Formula (65).

$$\text{TimeDC} = \text{TimeDataCycle} \times 31,25 \mu\text{s} \quad (65)$$

where

TimeDC is the time for the isochronous data cycle
TimeDataCycle is the factor for the TimeDC
31,25 μ s is the time base for the calculation

5.2.15.3 Coding of the field TimeIOInput

This field shall be coded as data type Unsigned32 with values according to Table 600. The time base is ns.

Table 600 – TimeIOInput

Value (decimal)	Meaning
0 – 32 000 000	Optional
Other	Reserved

5.2.15.4 Coding of the field TimeIOOutput

This field shall be coded as data type Unsigned32 with values according to Table 601. The time base is ns.

Table 601 – TimelOOutput

Value (decimal)	Meaning
0 – 32 000 000	Optional
other	Reserved

5.2.15.5 Coding of the field TimelOInputValid

This field shall be coded as data type Unsigned32 with values according to Table 602. The time base is ns.

Table 602 – TimelOInputValid

Value (decimal)	Meaning
0 – 32 000 000	Optional
other	Reserved

5.2.15.6 Coding of the field TimelOOutputValid

This field shall be coded as data type Unsigned32 with values according to Table 603. The time base is ns.

Table 603 – TimelOOutputValid

Value (decimal)	Meaning
0 – 32 000 000	Optional
other	Reserved

5.2.15.7 Coding of the field ControllerApplicationCycleFactor

This field shall be coded as data type Unsigned16 with values according to Table 604.

Table 604 – ControllerApplicationCycleFactor

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001 – 0x0400	Optional
0x0401 – 0xFFFF	Reserved

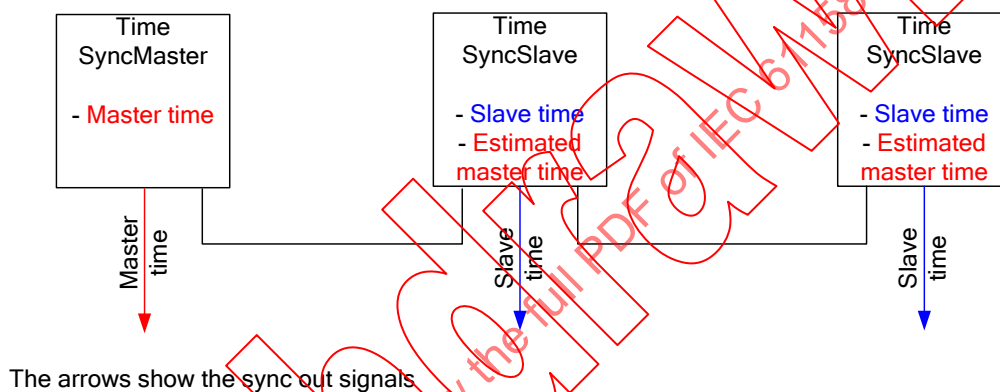
5.2.16 Coding section related to Physical Time Data**5.2.16.1 Coding of the field TimePLLWindow**

This field shall be coded as data type Unsigned32 with the values according to Table 605. The time base shall be one nanosecond.

Table 605 – TimePLLWindow

Value (hexadecimal)	Meaning	SyncID (Time)
0x0000	Disabled	—
0x03E8	Default	Optional (1 μ s)
0x2710	Default	Optional (10 μ s)
0x000186A0	Default	Mandatory (100 μ s)
0x000F4240	Default	Optional (1 ms)
0x00989680	Default	Optional (10 ms)
other	Reserved	—

The definition of the TimePLLWindow is shown in Figure 89 and Figure 90.

**Figure 89 – Toplevel view to the time PLL window**

NOTE 1 The SyncSlaves estimates the master time by means of IEEE 802.1AS. It control its slave time to follow the master time, but in real life it follows the estimated master time. Because of the noisy environment the estimated master time is more or less equal to the master time.

NOTE 2 The external measurement of the TimePLLWindow is done between master time and slave time. The internal measurement of a SyncSlave is done between estimated master time and slave time.

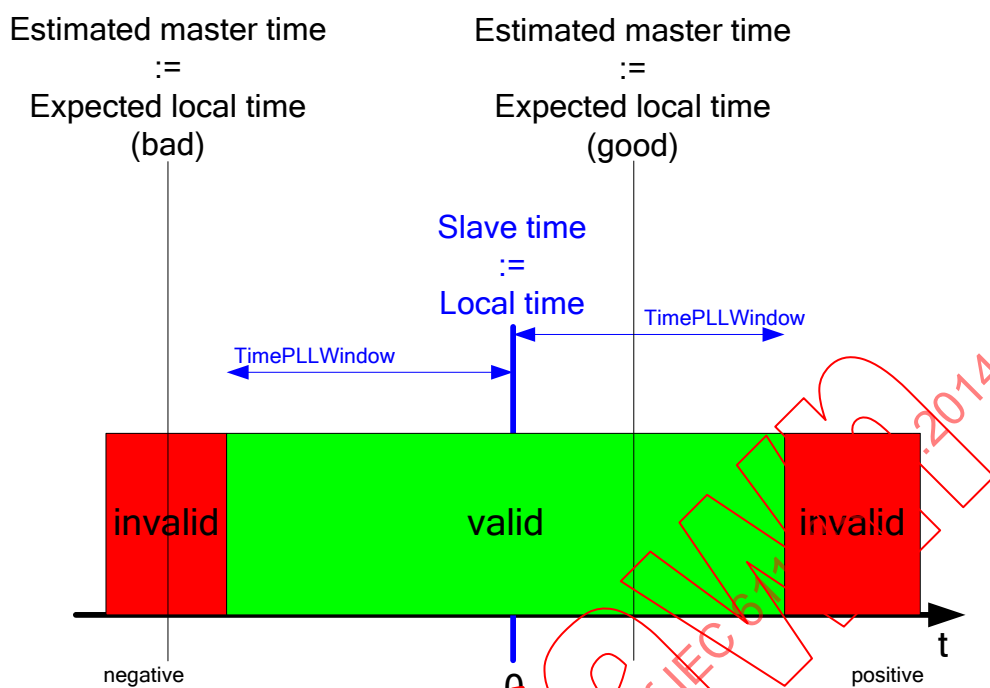


Figure 90 – Definition of time PLL window

5.2.16.2 Coding of the field TimeMasterPriority1

This field shall be coded as Unsigned8 with the value according to Table 606.

Table 606 – TimeMasterPriority1

Value (hexadecimal)	Usage	Meaning
0x00	Reserved	—
0x01 – 0x7F	Master	Optional
0x80	Master	Default for the role Time Master
0x81 – 0xFE	Master	Optional
0xFF	Slave	Default for the role Time slave

5.2.16.3 Coding of the field TimeMasterPriority2

This field shall be coded as Unsigned8 with the value according to Table 607.

Table 607 – TimeMasterPriority2

Value (hexadecimal)	Usage	Meaning
0x00 – 0xF7	Reserved	—
0xF8	Default	Default
0xF9 – 0xFF	Reserved	—

5.2.17 Coding section related to Media Redundancy

5.2.17.1 Coding of the field MRP_RingState

This field shall be coded as Unsigned16 with the values according to Table 608.

Table 608 – MRP_RingState

Value (hexadecimal)	Meaning	Usage
0x0000	Ring open	Redundancy manager in Ring open state
0x0001	Ring closed	Redundancy manager in Ring closed state
0x0002 – 0xFFFF	Reserved	—

5.2.17.2 Coding of the field MRP_DomainUUID

This field shall be coded as UUID with the values according to Table 609.

Table 609 – MRP_DomainUUID

Value (UUID)	Meaning	Usage
00000000-0000-0000-0000-000000000000	—	Reserved
00000000-0000-0000-0000-000000000001 – FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFFE	UUID for media redundancy domain	Optional
FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFFF	Default UUID for media redundancy domain	Mandatory

5.2.17.3 Coding of the field MRP_LengthDomainName

This field shall be coded as data type Unsigned8 and shall be set according to Table 611.

Table 610 – MRP_LengthDomainName

Value (hexadecimal)	Meaning
0x00	Reserved
0x01 – 0xF0	Allowed values
0xF1 – 0xFF	Reserved

5.2.17.4 Coding of the field MRP_DomainName

This field shall be coded as data type OctetString with 1 to 240 octets according to 4.3.1.4.15.1.

NOTE The field MRP_DomainName is not terminated by zero.

5.2.17.5 Coding of the field MRP_Role

This field shall be coded as data type Unsigned16 and shall be set according to Table 611.

Table 611 – MRP_Role

Value (hexadecimal)	Meaning
0x0000	Media Redundancy disabled
0x0001	Media Redundancy Client
0x0002	Media Redundancy Manager
0x0003 – 0xFFFF	Reserved

5.2.17.6 Coding of the field MRP_Version

This field shall be coded as data type Unsigned16 and set according to Table 612.

Table 612 – MRP_Version

Value (hexadecimal)	Meaning
0x0001	Version 1
other	Reserved

5.2.17.7 Coding of the field MRP_Prio

This field shall be coded as Unsigned16 and set according to Table 613.

Table 613 – MRP_Prio

Value (hexadecimal)	Meaning
0x0000	Highest priority redundancy manager
0x1000 – 0x7000	High priorities
0x8000	Default priority for redundancy manager
0x9000 – 0xE000	Low priorities
0xF000	Lowest priority redundancy manager
other	Reserved

5.2.17.8 Coding of the field MRP_TOPchgT

This field defines the common point of time that shall be used to invoke the Flush Filtering Data base service and shall be coded as data type Unsigned16. The value shall be set according to Table 614 with a time base of 10 ms.

Table 614 – MRP_TOPchgT

Value (decimal)	Meaning	Usage
0	0 ms	Clear filtering database (FDB table) immediately
1	10 ms	Mandatory
2 – 100	20 ms – 1 s	Optional
101 – 65 535	Reserved	—

5.2.17.9 Coding of the field MRP_TOPNRmax

This field shall be coded as data type Unsigned16 and set according to Table 615.

Table 615 – MRP_TOPNRmax

Value (decimal)	Meaning	Usage
0	Reserved	—
1	1 iteration	Optional
2	2 iterations	Optional
3	3 iterations	Mandatory (200 ms reconfiguration time)
4	4 iterations	Optional
5	5 iterations	Optional
6 – 65 535	Reserved	—

5.2.17.10 Coding of the field MRP_TSTshortT

This field shall be coded as data type Unsigned16 and set according to Table 616.

Table 616 – MRP_TSTshortT

Value (decimal)	Meaning	Usage
0	Reserved	—
1 – 9	1 – 9 ms	Optional (short test interval)
10	10 ms	Mandatory (200 ms reconfiguration time)
10 – 500	10 + 500 ms	Optional (short test interval)
501 – 65 535	Reserved	—

5.2.17.11 Coding of the field MRP_TSTdefaultT

This field shall be coded as data type Unsigned16 and set according to Table 617.

Table 617 – MRP_TSTdefaultT

Value (decimal)	Meaning	Usage
0	Reserved	—
1 – 19	1 – 19 ms	Optional (default test interval)
20	20 ms	Mandatory (200 ms reconfiguration time)
21 – 1 000	21 ms – 1 s	Optional (default test interval)
1 001 – 65 535	Reserved	—

5.2.17.12 Coding of the field MRP_TSTNRmax

This field shall be coded as data type Unsigned16 and set according to Table 618.

Table 618 – MRP_TSTNRmax

Value (decimal)	Meaning	Usage
0 – 1	Reserved	—
2	2 outstanding test indications cause ring failure	Optional
3	3 outstanding test indications cause ring failure	Mandatory (200 ms reconfiguration time)
4 – 10	4 – 10 outstanding test indications cause ring failure	Optional
11 – 65 535	Reserved	—

5.2.17.13 Coding of the field MRP_LNKdownT

This field shall be coded as data type Unsigned16. The coding shall be according to Table 619.

Table 619 – MRP_LNKdownT

Value (decimal)	Meaning	Usage
0	Reserved	—
1 – 19	1 – 19 ms Link Down interval	Optional
20	20 ms Link Down interval	Mandatory
21 – 1 000	21 – 1 000 ms Link Down interval	Optional
1 001 – 65 535	Reserved	—

5.2.17.14 Coding of the field MRP_LNKupT

This field shall be coded as data type Unsigned16 according to Table 620.

Table 620 – MRP_LNKupT

Value (decimal)	Meaning	Usage
0	Reserved	—
1 – 19	1 – 19 ms Link Up interval	Optional
20	20 ms Link Up interval	Mandatory
21 – 1 000	21 – 1 000 ms Link Up interval	Optional
1 001 – 65 535	Reserved	—

5.2.17.15 Coding of the field MRP_LNKNRmax

This field shall be coded as data type Unsigned16 according to Table 621.

Table 621 – MRP_LNKNRmax

Value (decimal)	Meaning	Usage
0	Reserved	—
1	1 iteration	Optional
2	2 iterations	Optional
3	3 iterations	Optional
4	4 iterations	Mandatory
5	5 iterations	Optional
6 – 65 535	Reserved	—

5.2.17.16 Coding of the field MRP_Check

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0: MRP_Check.MediaRedundancyManager

This field shall be set according to Table 622.

Table 622 – MRP_Check.MediaRedundancyManager

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Disable MediaRedundancyManager diagnosis
0x01	ON	Enable MediaRedundancyManager diagnosis (see Table 516)

Bit 1: MRP_Check.MRP_DomainUUID

This field shall be set according to Table 623.

Table 623 – MRP_Check.MRP_DomainUUID

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Disable the check of the MRP_DomainUUID vs. LLDP_PNIO_MRPPORTSTATUS
0x01	ON	Enable the check of the MRP_DomainUUID vs. LLDP_PNIO_MRPPORTSTATUS and the generation of "Peer MRP domain mismatch" (see Table 515)

Bit 2 – 23: MRP_Check.reserved_1

This field shall be set according to 3.4.2.2.

Bit 24 – 31: MRP_Check.reserved_2

This field shall be set to zero.

5.2.18 Coding section related to fiber optics**5.2.18.1 Coding of the field VendorBlockType**

This field shall be coded as data type Unsigned16 and set according to Table 624.

Table 624 – VendorBlockType

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Vendor specific

5.2.18.2 Coding of the field FiberOpticType

This field shall be coded as data type Unsigned32 and set according to Table 625.

Table 625 – FiberOpticType

Value (hexadecimal)	Meaning
0x00000000	No fiber type adjusted
0x00000001	9 µm single mode fiber
0x00000002	50 µm multi mode fiber
0x00000003	62,5 µm multi mode fiber
0x00000004	SI-POF ^a , NA = 0,5
0x00000005	SI-PCF ^a , NA = 0,36
0x00000006	LowNA-POF ^a , NA = 0,3
0x00000007	GI-POF ^a
0x00000008	GI-PCF ^a
0x00000009 – 0x0000007F	Reserved
0x00000080 – 0x000000FF	Vendor specific
0x00000100 – 0xFFFFFFFF	Reserved
^a See IEC 61158-2.	

5.2.18.3 Coding of the field FiberOpticCableType

This field shall be coded as data type Unsigned32. The coding shall be according to Table 626.

Table 626 – FiberOpticCableType

Value (hexadecimal)	Meaning
0x0000	No cable specified
0x0001	Inside/outside cable, fixed installation
0x0002	Inside/outside cable, flexible installation
0x0003	Outdoor cable, fixed installation
0x0004 – 0xFFFFFFFF	Reserved

5.2.18.4 Coding of the field FiberOpticPowerBudgetType

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 30: FiberOpticPowerBudgetType.Value

This field shall be set according to Table 627.

Table 627 – FiberOpticPowerBudgetType.Value

Value (hexadecimal)	Meaning	Usage
0	0 dB	Mandatory for maintenance demanded
0x0001 – 0x0013	0,1 dB to 1,9 dB	Optional
0x0014	2 dB	Mandatory for maintenance required
0x0015 – 0x03E7	2,1 dB to 99,9 dB	Optional
0x03E8 – 0x7FFFFFFF	Reserved	—

Bit 31: FiberOpticPowerBudgetType.CheckEnable

This field shall be set according to Table 628.

Table 628 – FiberOpticPowerBudgetType.CheckEnable

Value (hexadecimal)	Meaning
0x0	OFF
0x1	ON Comparison value is FiberOpticPowerBudgetType.Value

5.2.18.5 Coding of the field FiberOpticManufacturerSpecific

This field shall be coded as data type OctetString.

5.2.19 Coding section related network components**5.2.19.1 Coding of the field NCDropBudgetType**

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 30: NCDropBudgetType.Value

This field shall be set according to Table 629.

Table 629 – NCDropBudgetType.Value

Value (hexadecimal)	Meaning	Usage
0	Reserved	—
0x0001 – 0x0002	Number of dropped frames	Optional
0x0003	Number of dropped frames	Mandatory for maintenance required
0x0004 – 0x0009	Number of dropped frames	Optional
0x000A	Number of dropped frames	Mandatory for maintenance demanded
0x000B – 0x03E7	Number of dropped frames	Optional
0x03E8 – 0x7FFFFFFF	Reserved	—

Bit 31: NCDropBudgetType.CheckEnable

This field shall be set according to Table 630.

Table 630 – NCDropBudgetType.CheckEnable

Value (hexadecimal)	Meaning
0x0	OFF
0x1	ON Comparison value is NCDropBudgetType.Value

The checking of the dropped frames shall be done according to Figure 91 and Figure 92.

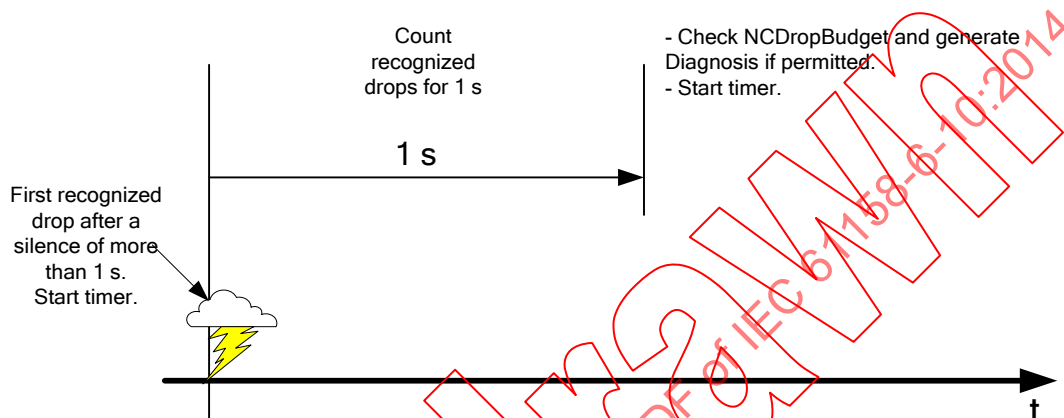


Figure 91 – Detection of dropped frames — appear

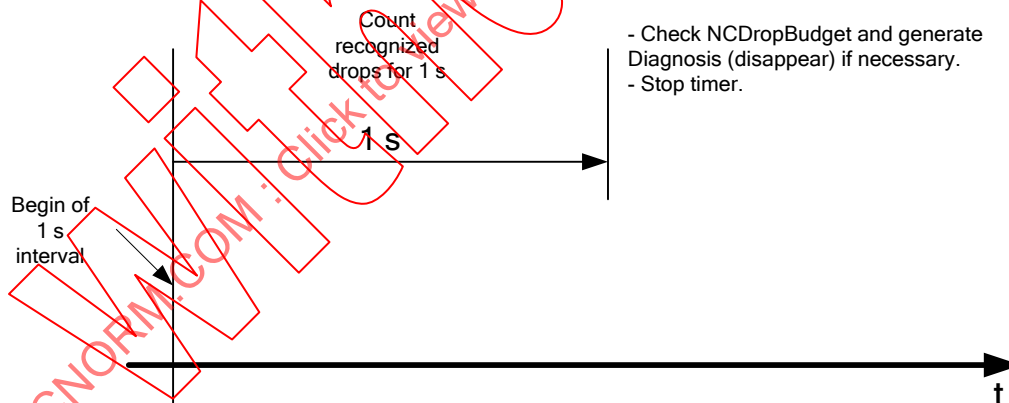


Figure 92 – Detection of dropped frames — disappear

5.2.20 Coding section related port statistic

5.2.20.1 Coding of the field ifInOctets

This field shall be coded as data type Unsigned32 with the values according to IETF RFC 1213. The value zero shall be used if not supported.

5.2.20.2 Coding of the field ifOutOctets

This field shall be coded as data type Unsigned32 with the values according to IETF RFC 1213. The value zero shall be used if not supported.

5.2.20.3 Coding of the field ifInDiscards

This field shall be coded as data type Unsigned32 with the values according to IETF RFC 1213. The value zero shall be used if not supported.

5.2.20.4 Coding of the field ifOutDiscards

This field shall be coded as data type Unsigned32 with the values according to IETF RFC 1213. The value zero shall be used if not supported.

5.2.20.5 Coding of the field ifInErrors

This field shall be coded as data type Unsigned32 with the values according to IETF RFC 1213. The value zero shall be used if not supported.

5.2.20.6 Coding of the field ifOutErrors

This field shall be coded as data type Unsigned32 with the values according to IETF RFC 1213. The value zero shall be used if not supported.

5.2.21 Coding section related to fast start up

5.2.21.1 Coding of the field FSHelloMode

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 1: FSHelloMode.Mode

This field shall be set according to Table 631.

Table 631 – FSHelloMode.Mode

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Default
0x01	Send DCP_Hello.req on LinkUp	—
0x02	Send DCP_Hello.req on LinkUp after HelloDelay	—
0x03	Reserved	Reserved

Bit 2 – 23: FSHelloMode.reserved_1

This field shall be set according to 3.4.2.2.

Bit 24 – 31: FSHelloMode.reserved_2

This field shall be set to zero.

5.2.21.2 Coding of the field FSHelloInterval

This field shall be coded as data type Unsigned32. The coding shall be according to Table 632.

Table 632 – FSHelloInterval

Value (hexadecimal)	Meaning	Usage
0x00000001E	30 ms Wait this time after the first DCP_Hello.req before conveying a second DCP_Hello.req	Default
0x000000032	50 ms Wait this time after the first DCP_Hello.req before conveying a second DCP_Hello.req	—
0x000000064	100 ms Wait this time after the first DCP_Hello.req before conveying a second DCP_Hello.req	—
0x00000012C	300 ms Wait this time after the first DCP_Hello.req before conveying a second DCP_Hello.req	—
0x0000001F4	500 ms Wait this time after the first DCP_Hello.req before conveying a second DCP_Hello.req	—
0x0000003E8	1 000 ms Wait this time after the first DCP_Hello.req before conveying a second DCP_Hello.req	—
other	Reserved	—

5.2.21.3 Coding of the field FSHelloRetry

This field shall be coded as data type Unsigned32. The coding shall be according to Table 633.

Table 633 – FSHelloRetry

Value (hexadecimal)	Meaning	Usage
0x00000000	Reserved	—
0x00000001 – 0x00000002	Number of retransmission of the Hello.req	—
0x00000003	Number of retransmission of the Hello.req	Default
0x00000004 – 0x0000000F	Number of retransmission of the Hello.req	—
0x00000010 – 0xFFFFFFFF	Reserved	—

5.2.21.4 Coding of the field FSHelloDelay

This field shall be coded as data type Unsigned32. The coding shall be according to Table 634.

Table 634 – FSHelloDelay

Value (hexadecimal)	Meaning	Usage
0x000000000	OFF	Default
0x000000032	50 ms Wait this time after the first LinkUp.ind before conveying a DCP_Hello.req	—
0x000000064	100 ms Wait this time after the first LinkUp.ind before conveying a DCP_Hello.req	—
0x0000001F4	500 ms Wait this time after the first LinkUp.ind before conveying a DCP_Hello.req	—

Value (hexadecimal)	Meaning	Usage
0x0000003E8	1 000 ms Wait this time after the first LinkUp.ind before conveying a DCP_Hello.req	—
other	Reserved	—

5.2.21.5 Coding of the field FSPParameterMode

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 1: FSPParameterMode.Mode

This field shall be set according to Table 635.

Table 635 – FSPParameterMode.Mode

Value (hexadecimal)	Meaning	Usage
0x00	OFF	Default
0x01	ON	—
0x02	Reserved	—
0x03	Reserved	—

Bit 2 – 23: FSPParameterMode.reserved_1

This field shall be set according to 3.4.2.2.

Bit 24 – 31: FSPParameterMode.reserved_2

This field shall be set to zero.

5.2.21.6 Coding of the field FSPParameterUUID

This field shall be coded as data type UUID with values according to Table 636.

Table 636 – FSPParameterUUID

Value (UUID)	Meaning
00000000-0000-0000-0000-000000000000	Reserved
00000000-0000-0000-0000-000000000001 – FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFFFF	UUID for the record data (including physical device data, ...) delivered between IODConnectRes and IODControlReq.

5.2.22 Coding section related to DFP

5.2.22.1 Coding of the field NumberOfSubframeBlocks

This field shall be coded as data type Unsigned16 with values according to Table 637.

Table 637 – NumberOfSubframeBlocks

Value (hexadecimal)	Meaning
0x0000	Reserved
0x0001 – 0x00FF	Number of the following SubframeBlocks
0x0100 – 0xFFFF	Reserved

5.2.22.2 Coding of the field SFIOCRProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0 – 7: SFIOCRProperties.DistributedWatchDogFactor

This field shall be set according to Table 638.

Table 638 – SFIOCRProperties.DistributedWatchDogFactor

Value (hexadecimal)	Meaning	Description
0x00	Optional	Distributed WatchDog shall be disabled
0x01 – 0x02	Optional	A value “1” leads to an AR termination for one missed frame, a value “2” leads to an AR termination for two missed frames
0x03 – 0x1F	Mandatory	An expiration should be signaled to the application as a hint for a possible communication disturbance.
0x20 – 0xFF	Optional	An expiration should be signaled to the application as a hint for a possible communication disturbance.

The time base is the SendClockFactor multiplied with ReductionRatio of the monitored provider.

The DistributedWatchDogTime shall be calculated according to Formula (66).

$$\text{Distributed-WatchDogTime} = \text{Distributed-WatchDogFactor} \times \text{SendClockFactor} \times \text{ReductionRatio} \times 31,25 \mu\text{s} \quad (66)$$

where

DistributedWatchDogTime is the distributed watch dog time

DistributedWatchDogFactor is the factor for the calculation of the distributed watch dog

SendClockFactor is the send clock factor

ReductionRatio is the reduction ratio

Bit 8 – 15: SFIOCRProperties.RestartFactorForDistributedWD

This field shall be set according to Table 639.

Table 639 – SFIOCRProperties.RestartFactorForDistributedWD

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	No restart delay necessary
0x01 – 0x09	Optional	Less than 1 s restart delay
0x0A – 0x50	Mandatory	1 s to 8 s restart delay
0x51 – 0xFF	Optional	More than 8 s restart delay

The time base is 100 ms. The RestartForDistributedWDTime shall be calculated according to Formula (67).

$$\text{RestartForDistributed-WDTime} = \text{RestartFactorFor-DistributedWDFactor} \times 100 \text{ ms} \quad (67)$$

where

RestartForDistributedWdTime is the restart for distributed watch dog time

RestartFactorForDistributedWdFactor is the factor for the calculation of the restart for distributed watch dog time

The RestartFactorForDistributedWD shall only be used if the SFIOCRProperties.DistributedWatchDogFactor is not set to zero.

NOTE The RestartForDistributedWdTime is defined as minimum time having an accuracy between '-0' and '+1 s'.

Bit 16 – 23: SFIOCRProperties.DFPMode

This field shall be set according to Table 640.

Table 640 – SFIOCRProperties.DFPMode

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	End node mode
0x01 – 0x3F	Mandatory	Concatenating mode Own Subframe position
0x40 – 0xFF	Optional	Concatenating mode Own Subframe position

Bit 24 – 27: SFIOCRProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 28: SFIOCRProperties.reserved_2

This field shall be set to zero.

Bit 29: SFIOCRProperties.DFPDirection

This field shall be set according to Table 641.

Table 641 – SFIOCRProperties.DFPDirection

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	This Frame is intended to be handled by the DFP_INBOUND state machine.
0x01	Mandatory	This Frame is intended to be handled by the DFP_OUTBOUND state machine.

Bit 30: SFIOCRProperties.DFPRedundantPathLayout

This field shall be set according to Table 642.

Table 642 – SFIOCRProperties.DFPRedundantPathLayout

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	The Frame for the redundant path, coded by the FrameID with the least significant bit set to one, shall contain the ordering shown by SubframeData.
0x01	Mandatory	The Frame for the redundant path, coded by the FrameID with the least significant bit set to one, shall contain the inverse ordering of the SubframeData.

Bit 31: SFIOCRProperties.SFCRC16

This field shall be set according to Table 643.

Table 643 – SFIOCRProperties.SFCRC16

Value (hexadecimal)	Description
0x00	SFCRC16 and SFCycleCounter shall be created or set to zero by the sender and not checked by the receiver.
0x01	SFCRC16 and SFCycleCounter shall be created by the sender and checked by the receiver.

5.2.22.3 Coding of the field SubframeData

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

NOTE The SFCRC16 of the frame header and the SFEndDelimiter are implicit. See CSE_SDU.

Bit 0 – 6: SubframeData.Position

This field shall be set according to Table 644.

Table 644 – SubframeData.Position

Value (hexadecimal)	Meaning
0x00	Reserved
0x01 – 0x7F	SFPosition.Position of the Subframe

Bit 7: SubframeData.reserved_1

This field shall be set according to 3.4.2.2.

Bit 8 – 15: SubframeData.DataLength

This field shall be set according to Table 645.

Table 645 – SubframeData.DataLength

Value (hexadecimal)	Meaning
0x00	Reserved
0x01 – 0xFF	Number of octets of the C_SDU of the SUBFRAME

Bit 16 – 31: SubframeData.reserved_2

This field shall be set according to 3.4.2.2.

5.2.22.4 Frame late error

The checking of the DFP frame late error shall be done according to Figure 93.

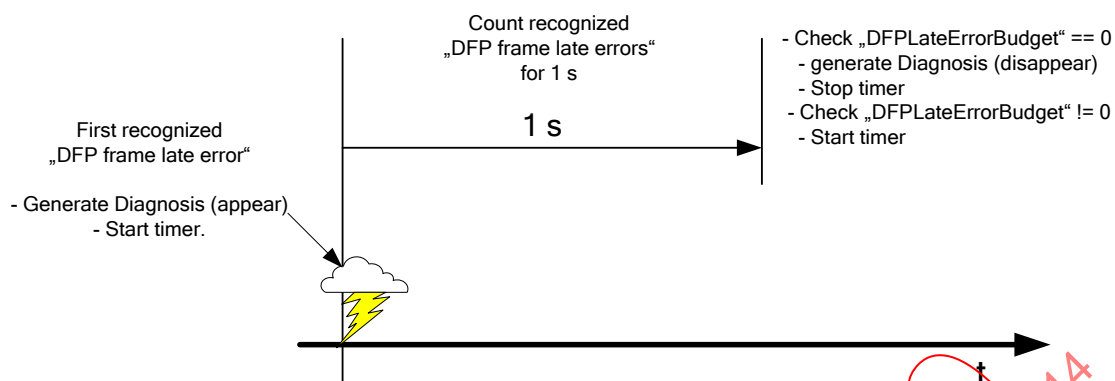


Figure 93 – Detection of DFP late error — appear and disappear

Table 646 – Event function table

Function name	Operations
Appear	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= Dynamic Frame Packing problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelProperties.Maintenance := MaintenanceDemanded ChannelErrorType := "Dynamic frame packing function mismatch" ExtChannelErrorType := "Frame late error" (see Formula (55)) Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Disappear	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= Dynamic Frame Packing problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Disappears ChannelProperties.Maintenance := MaintenanceDemanded ChannelErrorType := "Dynamic frame packing function mismatch" ExtChannelErrorType := "Frame late error" (see Formula (55)) Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Do nothing	This state change shall not indicate an event

5.2.22.5 Coding of the field SubframeOffset

This field shall be coded as data type Unsigned16 with values according to Table 647.

Table 647 – SubframeOffset

Value (hexadecimal)	Meaning
0x0000 – 0x0005	Reserved
0x0006 – 0x059B	Number of octets from the begin of the frame C_SDU to the begin of the according subframe C_SDU
0x059C – 0xFFFF	Reserved

5.2.23 Coding section related to MRPD

5.2.23.1 Overview

The following rules shall be applied for the MediaRedundancyWatchDog of MRPD. The time value (e.g. 8 s) of the MediaRedundancyWatchDog shall be defined locally to avoid “false positive” events in case of frame errors. It shall be activated if MRPD is used.

5.2.23.2 MediaRedundancyWatchDog expired

The checking whether the MediaRedundancyWatchDog (as an example with the time value 1 s) is expired shall be done according to Figure 94.

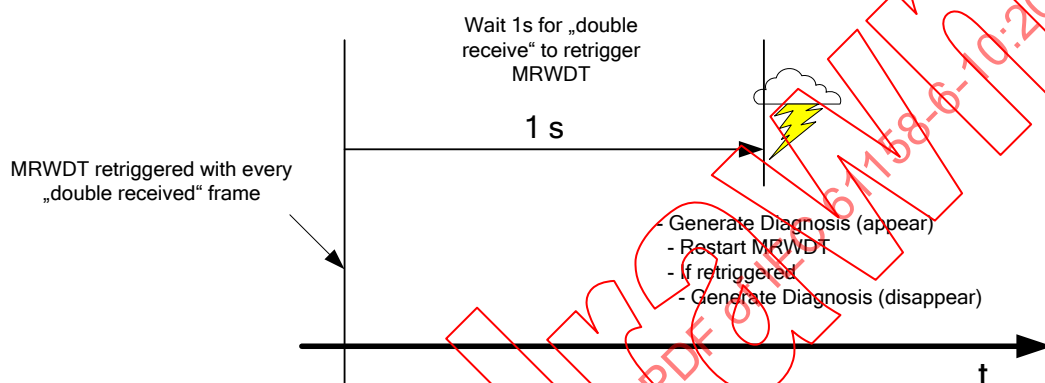


Figure 94 – MediaRedundancyWatchDog expired — appear and disappear

Table 648 – Event function table

Function name	Operations
Appear	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= MRPD problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelProperties.Maintenance :=MaintenanceDemanded ChannelErrorType := “Media redundancy with planned duplication mismatch” ExtChannelErrorType := “MRPD duplication void” (see Formula (56)) Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Disappear	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= MRPD problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Disappears ChannelProperties.Maintenance :=MaintenanceDemanded ChannelErrorType := “Media redundancy mismatch” ExtChannelErrorType := “MRPD duplication void” (see Formula (56)) Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Do nothing	This state change shall not indicate an event

5.2.24 Coding section related to auto configuration

5.2.24.1 Overview

The following coding section applies for the auto configuration functionality.

5.2.24.2 Coding of the field SendClockFactorArray

The field SendClockFactorArray is an array of 16 SCFEntry octets. It shall be used to code the supported SendClockFactors. Each bit of the array corresponds with one SendClockFactor and shall be calculated as shown in Formula (68) and (69).

$$N = (\text{SendClockFactor} - 1) \text{ DIV } 8 \quad (68)$$

where

N	is the number of the octet
SendClockFactor	is the supported SendClockFactor
8	is the bit size of a octet

$$M = (\text{SendClockFactor} - 1) \text{ MOD } 8 \quad (69)$$

where

M	is the number of the bit within the octet
SendClockFactor	is the supported SendClockFactor
8	is the bit size of a octet

NOTE The term DIV stands for division without remainder. The term MOD stands for the remainder of the division.

The SendClockFactorArray shall be structured in ascending order without gaps.

The coding of the SCFEntry shall be according to 3.4.2.3 and the individual bits shall have the following meaning:

Bit 0: SCFEntry_0

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Table 649 – SCFEntry

Value (hexadecimal)	Meaning
0x00	SendClockFactor not supported
0x01	SendClockFactor supported

Bit 1: SCFEntry_1

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Bit 2: SCFEntry_2

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Bit 3: SCFEntry_3

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Bit 4: SCFEntry_4

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Bit 5: SCFEntry_5

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Bit 6: SCFEntry_6

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

Bit 7: SCFEntry_7

This bit shall be set with the values according to Table 649 if the SendClockFactor with the number that meets Formula (68) and (69) is present.

5.2.24.3 Coding of the field ACCommunicationProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0: ACCommunicationProperties.DFP

This field shall be set according to Table 650.

Table 650 – ACCommunicationProperties.DFP

Value (hexadecimal)	Meaning
0x00	Not supported
0x01	Supported

Bit 1 – 2: ACCommunicationProperties.RTC3

This field shall be set according to Table 651.

Table 651 – ACCommunicationProperties.RTC3

Value (hexadecimal)	Meaning
0x00	Not supported
0x01	PDIRFrameData with local and forwarding (including the FrameSendOffset) information needed
0x02	PDIRFrameData with local information needed
0x03	Reserved

Bit 3: ACCommunicationProperties.RTCUDP

This field shall be set according to Table 652.

Table 652 – ACCommunicationProperties.RTCUDP

Value (hexadecimal)	Meaning
0x00	RT_CLASS_UDP not supported
0x01	RT_CLASS_UDP supported

Bit 3 – 15: ACCommunicationProperties.reserved_1

This field shall be set according to 3.4.2.2.

Bit 16 – 31: ACCommunicationProperties.reserved_2

This field shall be set to zero.

5.2.24.4 Coding of the field ACMinDeviceInterval

This field shall be coded as data type Unsigned16 according to Table 653.

Table 653 – ACMinDeviceInterval

Value (hexadecimal)	Meaning
0x00	Reserved
0x01	31,25 μ s MinDeviceInterval as a factor of 31,25 μ s.
...	...
0x1000	128 ms MinDeviceInterval as a factor of 31,25 μ s.
0x1001 – 0xFFFF	Reserved

5.2.25 Coding section related to controller to controller communication**5.2.25.1 Overview**

The following coding section applies for the controller to controller communication.

5.2.25.2 Coding of the field FromOffsetData

This field shall be coded as data type Unsigned32 according to Table 654.

Table 654 – FromOffsetData

Value (hexadecimal)	Meaning	Description
0x00000000 – 0xFFFFFFFF	Mandatory	Contains the offset in octets as base offset for the request and the delivered chunk of data in the response.

5.2.25.3 Coding of the field NextOffsetData

This field shall be coded as data type Unsigned32 according to Table 655.

Table 655 – NextOffsetData

Value (hexadecimal)	Meaning	Description
0x00000000	Mandatory	The last chunk of data is read.
Other	Mandatory	Delivers the next to be used offset value for convenience. More data is available.

5.2.25.4 Coding of the field TotalSize

This field shall be coded as data type Unsigned32 according to Table 656.

Table 656 – TotalSize

Value (hexadecimal)	Meaning	Description
0x00000000	Reserved	—
Other	Mandatory	Contains the total size in octets of the requested object.

5.2.26 Coding section related to system redundancy

5.2.26.1 Overview

The following coding section applies for the system redundancy functionality.

5.2.26.2 Coding of the field RedundancyInfo

The coding of this field shall be according to 3.7.3.4 and the individual bits shall have the following meaning:

Bit 0: RedundancyInfo.EndPoint1

This field shall be coded according to Table 657.

Table 657 – RedundancyInfo.EndPoint1

Value (hexadecimal)	Meaning
0x00	The position of the delivering node in a rack is not the left or below one.
0x01	The delivering node is the left or below one.

Bit 1: RedundancyInfo.EndPoint2

This field shall be coded according to Table 658.

Table 658 – RedundancyInfo.EndPoint2

Value (hexadecimal)	Meaning
0x00	The delivering node is not the right or above one.
0x01	The delivering node is the right or above one.

Valid combinations of RedundancyInfo.EndPoint1 and RedundancyInfo.EndPoint2 are shown in Table 659.

Table 659 – Valid combination of RedundancyInfo.EndPoint1 and RedundancyInfo.EndPoint2

RedundancyInfo. EndPoint1	RedundancyInfo. EndPoint2	Meaning
0x00	0x00	Reserved
0x00	0x01	Valid combination
0x01	0x00	Valid combination
0x01	0x01	Reserved

Bit 2 – 15: RedundancyInfo.reserved

This field shall be set to zero.

5.2.26.3 Coding of the field SRProperties

The coding of this field shall be according to 3.4.2.5 and the individual bits shall have the following meaning:

Bit 0: SRProperties.InputValidOnBackupAR

This field shall be coded according to Table 660.

Table 660 – SRProperties.InputValidOnBackupAR

Value (hexadecimal)	Meaning
0x00	The device may deliver valid input data
0x01	The device shall deliver valid input data

Bit 1: SRProperties.ActivateRedundancyAlarm

This field shall be coded according to Table 661.

Table 661 – SRProperties.ActivateRedundancyAlarm

Value (hexadecimal)	Meaning
0x00	The device shall not send Redundancy alarm
0x01	The device shall send Redundancy alarm

Bit 2 – 15: SRProperties.Reserved_1

This field shall be set to zero.

Bit 16 – 31: SRProperties.Reserved_2

This field shall be set according to 3.4.2.2.

5.2.26.4 Coding of the field RedundancyDataHoldFactor

This field shall be coded as data type Unsigned16 according to Table 662. The time base shall be 1 ms.

Table 662 – RedundancyDataHoldFactor

Value (hexadecimal)	Meaning	Description
0x0000 – 0x0002	Reserved	—
0x0003 – 0x00C7	Optional	An expiration of the time leads to an AR termination.
0x00C8 – 0xFFFF	Mandatory	An expiration of the time leads to an AR termination.

The RedundancyDataHoldTime is calculated according to Formula (70).

$$\text{RDHT} = \text{RedundancyDataHoldFactor} \times 1 \text{ ms} \quad (70)$$

where

RDHT is the redundancy data hold

RedundancyDataHoldFactor is the factor for the calculation of the redundancy data hold time

5.2.26.5 Coding of the field NumberOfEntries

This field shall be coded as data type Unsigned16 according to Table 663.

Table 663 – NumberOfEntries

Value (hexadecimal)	Meaning	Use
0x0000 – 0xFFFF	Number of entries	—

5.2.27 PDU checking rules

5.2.27.1 General

5.2.27.1.1 Overview

The following rules shall be applied to check FAL PDUs at the receiver. PDU code checking rules provide redundant information in a condensed way. In case of a contradiction of PDU code checking rules with other parts of the specification, other parts of the specification shall have precedence.

NOTE A PDU is structured with blocks which are identified by their BlockType. Unknown or unexpected blocks lead to a structure fault.

5.2.27.1.2 Additional checking rules, given by profiles

Application profiles like system redundancy define addition PDU code checking rules. Thus, a parameter shown as valid in the following condensed way of the PDU code checking rules may be signaled as invalid in their context.

5.2.27.2 IODConnectReq

5.2.27.2.1 ArgsLength

This field shall be checked according to Table 664.

Table 664 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	Okay, check IODConnectReq-PDU
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRDataRequest	ArgsLength	!= Sum of all BlockLength + Number of Blocks × 4	CMRPC/CMINA: ArgsLength invalid
IODConnectReq	ARBlockReq	Not first block	RSP-, 0xDB, 0x81, 0x01, 0
IODConnectReq		Case ARProperties.DeviceAccess == 1: other blocks than ARBlockReq	CMRPC/CMINA: Unknown blocks

	Parameter	Checking rules	Behavior on match
IODConnectReq	IOCRBlockReq	Case ARProperties.Device-Access == 0: Number of IOCRBlockReq with CRType == "Input CR" == 0 OR Number of IOCRBlockReq with CRType == "Output CR" == 0	CMRPC/CMINA: IOCR missing
IODConnectReq	AlarmCRBlockReq	Case ARProperties.Device-Access == 0: Number of AlarmCRBlockReq != 1	CMRPC/CMINA: Wrong AlarmCRBlock count

5.2.27.2.2 ARBlockReq

This field shall be checked according to Table 665.

Table 665 – ARBlockReq – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0101	RSP-, 0xDB, 0x81, 0x01, 0
BlockLength	!= 54 + StationNameLength	RSP-, 0xDB, 0x81, 0x01, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x01, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x01, 3
ARType	== reserved OR not supported	RSP-, 0xDB, 0x81, 0x01, 4
ARUUID	== NIL	RSP-, 0xDB, 0x81, 0x01, 5
CMInitiatorMacAdd	If (IOCRProperties.RTClass == RT_CLASS_UDP) AND (CMInitiatorMacAdd != zero)	RSP-, 0xDB, 0x81, 0x01, 7
CMInitiatorMacAdd	== multicast address	RSP-, 0xDB, 0x81, 0x01, 7
CMInitiatorObjectUUID	<see next 5 lines>	
.. time low	!= 0xDEA00000	RSP-, 0xDB, 0x81, 0x01, 8
.. time mid	!= 0x6C97	RSP-, 0xDB, 0x81, 0x01, 8
.. time high version	!= 0x11D1	RSP-, 0xDB, 0x81, 0x01, 8
.. Clock[2]	!= 0x82, 0x71	RSP-, 0xDB, 0x81, 0x01, 8
.. Node[6] (contains instance, device, vendor)	Not checked	—
ARProperties.State	ARProperties.State != Active	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.Supervisor-TakeoverAllowed	Don't care	—
ARProperties.Parameterization Server	== External PrmServer	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.reserved_1	Not checked	—
ARProperties.DeviceAccess	== 1 AND ARType != IOSAR	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.CompanionAR	== 3	RSP-, 0xDB, 0x81, 0x01, 9

Parameter	Checking rules	Behavior on match
CMInitiatorActivityTimeoutFactor or	!= 1 to 1 000 (base 100 ms)	RSP-, 0xDB, 0x81, 0x01, 10
InitiatorUDPRTPort	If (IOCRProperties.RTClass == RT_CLASS_UDP) == 0 to 0x03FF else ignore	RSP-, 0xDB, 0x81, 0x01, 11
StationNameLength	== 0 OR > 240	RSP-, 0xDB, 0x81, 0x01, 12
CMInitiatorStationName	NOT (VisibleString, according to RFC 5890)	RSP-, 0xDB, 0x81, 0x01, 13

5.2.27.2.3 IOCRBlockReq

This field shall be checked according to Table 666.

Table 666 – IOCRBlockReq – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0102	—
BlockLength	!= 42 + NumberOfAPI × (8 + NumberOfIODataObjects × 6 + NumberOfIOCS × 6)	RSP-, 0xDB, 0x81, 0x02, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x02, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x02, 3
IOCRType	!= 0x01 to 0x04	RSP-, 0xDB, 0x81, 0x02, 4
IOCRReference	Is not unique OR (IOCRType == 0x04 AND no corresponding MCRBlock)	RSP-, 0xDB, 0x81, 0x02, 5
LT	Case IOCRProperties.RTClass != 0x04: != 0x8892 Case IOCRProperties.RTClass == 0x04: != 0x0800	RSP-, 0xDB, 0x81, 0x02, 6
IOCRProperties.RTClass	!= 0x01 to 0x04 OR not supported in GSDML	RSP-, 0xDB, 0x81, 0x02, 7
IOCRProperties.reserved_1	!= 0x0	RSP-, 0xDB, 0x81, 0x02, 7
IOCRProperties.reserved_2	!= 0x0	RSP-, 0xDB, 0x81, 0x02, 7
DataLength	If (IOCRProperties.RTClass == 0x04 AND DataLength != 12 to 1 440) OR ((IOCRProperties.RTClass == 0x01 OR IOCRProperties.RTClass == 0x02 OR IOCRProperties.RTClass == 0x03) AND DataLength != 40 to 1 440)	RSP-, 0xDB, 0x81, 0x02, 8

Parameter	Checking rules	Behavior on match
FrameID	If (IOCRTType==(MULTICAST_CONSUMER_CR OR MULTICAST_PROVIDER_CR)) AND (IOCRProperties.RTClass == 0x01 AND FrameID!=0xF800 to 0xFBFF) OR (IOCRProperties.RTClass == 0x02 AND FrameID!=0xBC00 to 0xBFFF) OR (IOCRProperties.RTClass == 0x03 AND FrameID!=0x0100 to 0x7FFF) OR (IOCRProperties.RTClass == 0x04 AND FrameID!=0xF800 to 0xFBFF)	RSP-, 0xDB, 0x81, 0x02, 9
FrameID	If IOCRTType==INPUT_CR AND ((IOCRProperties.RTClass == 0x01 AND FrameID!=0xC000 to 0xF7FF) OR (IOCRProperties.RTClass == 0x02 AND FrameID!=0x8000 to 0xBBFF) OR (IOCRProperties.RTClass == 0x03 AND FrameID!=0x0100 to 0x7FFF) OR (IOCRProperties.RTClass == 0x04 AND FrameID!=0xC000 to 0xF7FF))	RSP-, 0xDB, 0x81, 0x02, 9
SendClockFactor	!=1 to 128 OR not supported in GSDML OR (IOCRProperties.RTClass == 0x3 AND SendClockFactor != LocalClock) OR (IOCRProperties.RTClass == 0x3 AND SendClockFactor × ReductionRatio < LocalClock)	RSP-, 0xDB, 0x81, 0x02, 10
ReductionRatio	If IOCRProperties.RTClass == 0x01 to 0x03 AND (ReductionRatio != 1 to 512 OR not supported in GSDML) OR (ReductionRatio ≥ 256 AND SendClockFactor > 64) OR (ReductionRatio == 512 AND SendClockFactor > 32)	RSP-, 0xDB, 0x81, 0x02, 11

Parameter	Checking rules	Behavior on match
ReductionRatio	If IOCProperties.RTClass == 0x04 AND (ReductionRatio != 1 to 16 384 OR not supported in GSDML) OR (ReductionRatio ≥ 8 192 AND SendClockFactor > 64) OR (ReductionRatio == 16 384 AND SendClockFactor > 32))	RSP-, 0xDB, 0x81, 0x02, 11
Phase	If Phase == 0x0 OR Phase > ReductionRatio	RSP-, 0xDB, 0x81, 0x02, 12
Sequence	not checked	—
FrameSendOffset	If (FrameSendOffset ≥ SendClockFactor × 31 250 ns) OR ((FrameSendOffset ≥ 0x003D0900) AND (FrameSendOffset ≤ 0xFFFFFFFF))	RSP-, 0xDB, 0x81, 0x02, 14
DataHoldFactor (legacy)	if DataHoldFactor (legacy) != DataHoldFactor	RSP-, 0xDB, 0x81, 0x02, 15
DataHoldFactor	If DataHoldFactor != 0x0001 to 0x1E00 OR (IOCProperties.RTClass == 0x04 AND DataHoldFactor × ReductionRatio × SendClockFactor × 31,25 > 61 440 000) OR (IOCProperties.RTClass == 0x01 to 0x03 AND DataHoldFactor × ReductionRatio × SendClockFactor × 31,25 > 1 920 000)	RSP-, 0xDB, 0x81, 0x02, 16
IOCRTagHeader.IOCRVLANID	Not checked	—
IOCRTagHeader.IOUserPriority	!= IO CR Priority	RSP-, 0xDB, 0x81, 0x02, 17
IOCRMulticastMACAdd	Case IOCRTType == 0x03 or 0x04: Is not multicast address	RSP-, 0xDB, 0x81, 0x02, 18
NumberOfAPI	NumberOfAPI == 0	RSP-, 0xDB, 0x81, 0x02, 19
API	API not in corresponding ExpectedSubmoduleBlockReq	RSP-, 0xDB, 0x81, 0x02, 20
NumberOfIODataObjects	== 0x0 AND NumberOfIOCS = 0x0	RSP-, 0xDB, 0x81, 0x02, 21
SlotNumber	Not in corresponding ExpectedSubmoduleBlockReq	RSP-, 0xDB, 0x81, 0x02, 22

Parameter	Checking rules	Behavior on match
SubslotNumber	Is not unique (in conjunction with SlotNumber) OR Not in corresponding ExpectedSubmoduleBlockReq OR ((corresponding SubmoduleProperties.Type == 0x0 to 0x1) AND (IOCRTType == 0x2 OR 0x4)) OR ((corresponding SubmoduleProperties.Type == 0x2) AND (IOCRTType == 0x1 OR 0x3))	RSP-, 0xDB, 0x81, 0x02, 23
IODataObjectFrameOffset	>= DataLength OR (IODataObjectFrameOffset + (effective SubmoduleDataLength + effective LengthIOPS of corresponding submodule) >= DataLength) OR (IOCRTType != "multicast consumer CR" AND (IODataObjectFrameOffset + (effective SubmoduleDataLength + effective LengthIOPS of corresponding submodule) overlap with other IODataObjects or IOCSes))	RSP-, 0xDB, 0x81, 0x02, 24
NumberOfIOCS	== 0x0 AND NumberOfIODataObjects == 0x0	RSP-, 0xDB, 0x81, 0x02, 25
SlotNumber	Not in corresponding ExpectedSubmoduleBlockReq	RSP-, 0xDB, 0x81, 0x02, 26
SubslotNumber	Is not unique (in conjunction with SlotNumber) OR Not in corresponding ExpectedSubmoduleBlockReq OR ((corresponding SubmoduleProperties.Type == 0x0 to 0x1) AND (IOCRTType == 0x1)) OR ((corresponding SubmoduleProperties.Type == 0x2) AND (IOCRTType == 0x2))	RSP-, 0xDB, 0x81, 0x02, 27
IOCSFrameOffset	>= DataLength OR not unique OR (IOCSFrameOffset + (effective LengthIOCS of corresponding submodule) >= DataLength) OR (IOCSFrameOffset + (effective LengthIOCS of corresponding submodule) overlap with other IODataObjects or IOCSes)	RSP-, 0xDB, 0x81, 0x02, 28

5.2.27.2.4 AlarmCRBlockReq

This field shall be checked according to Table 667.

Table 667 – AlarmCRBlockReq – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0103	—
BlockLength	!= 22	RSP-, 0xDB, 0x81, 0x04, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x04, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x04, 3
AlarmCRType	!= 0x0001	RSP-, 0xDB, 0x81, 0x04, 4
LT	Case AlarmCRProperties.Transport == 0x00: != 0x8892 Case AlarmCRProperties.Transport == 0x01: != 0x0800	RSP-, 0xDB, 0x81, 0x04, 5
AlarmCRProperties.Priority	Not checked	—
AlarmCRProperties.Transport	Not checked	—
AlarmCRProperties.reserved	!= 0x0	RSP-, 0xDB, 0x81, 0x04, 6
RTATimeoutFactor	!= 0x0001 to 0x0064 optional: AND != 0x0065 to 0xFFFF	RSP-, 0xDB, 0x81, 0x04, 7
RTARetries	!= 3 to 15	RSP-, 0xDB, 0x81, 0x04, 8
LocalAlarmReference	Not checked	—
MaxAlarmDataLength	!= 200 to 1 432	RSP-, 0xDB, 0x81, 0x04, 10
AlarmCRTagHeaderHigh. AlarmCRVLANID	Not checked	—
AlarmCRTagHeaderHigh. AlarmUserPriority	!= Alarm CR Priority High	RSP-, 0xDB, 0x81, 0x04, 11
AlarmCRTagHeaderLow. AlarmCRVLANID	Not checked	—
AlarmCRTagHeaderLow. AlarmUserPriority	!= Alarm CR Priority Low	RSP-, 0xDB, 0x81, 0x04, 12

5.2.27.2.5 ExpectedSubmoduleBlockReq

This field shall be checked according to Table 668.

Table 668 – ExpectedSubmoduleBlockReq – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0104	—
BlockLength	!= 4 + NumberOfAPI × (14 + NumberOfSubmodules × (8 + NumberOfSubsequentDataDescriptionBlocks × 6)	RSP-, 0xDB, 0x81, 0x03, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x03, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x03, 3
NumberOfAPI	== 0	RSP-, 0xDB, 0x81, 0x03, 4
API	> 0x0 AND not supported in GSDML	RSP-, 0xDB, 0x81, 0x03, 5
SlotNumber	!= 0x0 to 0x7FFF OR not supported in GSDML OR not unique	RSP-, 0xDB, 0x81, 0x03, 6

Parameter	Checking rules	Behavior on match
ModuleIdentNumber	== 0x00000000	RSP-, 0xDB, 0x81, 0x03, 7
ModuleProperties.reserved	!= 0x0	RSP-, 0xDB, 0x81, 0x03, 8
NumberOfSubmodules	== 0	RSP-, 0xDB, 0x81, 0x03, 9
SubslotNumber	Case ARProperties.PullModuleAlarmAllowed (=0): != 0x0001 to 0x7FFF OR not supported in GSDML OR not unique OR not used in at least one IOCR OR == 0x8000 to 0x8FFF AND API != 0 Case ARProperties.PullModuleAlarmAllowed (=1): != 0x0000 to 0x7FFF OR not supported in GSDML OR not unique OR not used in at least one IOCR OR == 0x8000 to 0x8FFF AND API != 0	RSP-, 0xDB, 0x81, 0x03, 10
SubmoduleProperties.Type	== (NO_IO OR INPUT) AND !(input description block follows) OR == (OUTPUT) AND !(output description block follows) OR == (IO) AND !(input description block follows AND output description block follows)	RSP-, 0xDB, 0x81, 0x03, 12
SubmoduleProperties.SharedInput	If SubmoduleProperties.Type == (OUTPUT DATA) AND SubmoduleProperties.SharedInput == 0x1	RSP-, 0xDB, 0x81, 0x03, 12
SubmoduleProperties.ReduceInputSubmoduleDataLength	Not checked	—
SubmoduleProperties.ReduceOutputSubmoduleDataLength	Not checked	—
SubmoduleProperties.DiscardIOXS	Not checked	—
SubmoduleProperties.reserved	!= 0x0	RSP-, 0xDB, 0x81, 0x03, 12
DataDescription.Type	If DataDescription.Type == (0x00 OR 0x03) OR DataDescription.Type == 0x01 AND SubmoduleProperties.Type == 0x02 OR DataDescription.Type == 0x02 AND SubmoduleProperties.Type != (0x02 OR 0x03)	RSP-, 0xDB, 0x81, 0x03, 13
DataDescription.reserved	!= 0x0	RSP-, 0xDB, 0x81, 0x03, 13
SubmoduleDataLength	!= 0 to 1 439	RSP-, 0xDB, 0x81, 0x03, 14
LengthIOPS	!= 0x01	RSP-, 0xDB, 0x81, 0x03, 15
LengthIOCS	!= 0x01	RSP-, 0xDB, 0x81, 0x03, 16

5.2.27.2.6 PrmServerBlock

This field shall be checked according to Table 669.

Table 669 – PrmServerBlock – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0105	—
BlockLength	!= 26 + StationNameLength	RSP-, 0xDB, 0x81, 0x05, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x05, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x05, 3
ParameterServerObjectUUID	Not checked	—
ParameterServerProperties	Not checked	—
CMIInitiatorActivityTimeoutFactor	!= 1 to 1 000	RSP-, 0xDB, 0x81, 0x05, 6
StationNameLength	!= 1 to 240	RSP-, 0xDB, 0x81, 0x05, 7
ParameterServerStationName	Not visible string according to RFC 5890	RSP-, 0xDB, 0x81, 0x05, 8

5.2.27.2.7 MCRBlock

This field shall be checked according to Table 670.

Table 670 – MCRBlockReq – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0106	—
BlockLength	!= (12 + StationNameLength + Padding) AND (BlockLength mod 4 == 0)	RSP-, 0xDB, 0x81, 0x06, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x06, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x06, 3
IOCRReference	Is not unique OR Corresponding IOCRTYPE != 0x4	RSP-, 0xDB, 0x81, 0x06, 4
AddressResolutionProperties. Protocol	!= 0x01 OR 0x02	RSP-, 0xDB, 0x81, 0x06, 5
AddressResolutionProperties. Reserved	!= 0x0	RSP-, 0xDB, 0x81, 0x06, 5
AddressResolutionProperties. Factor	!= 0x0001 to 0x0064 optional: AND != 0x0065 to 0xFFFF	RSP-, 0xDB, 0x81, 0x06, 5
MCITimeoutFactor	> 100	RSP-, 0xDB, 0x81, 0x06, 6
StationNameLength	!= 1 to 240	RSP-, 0xDB, 0x81, 0x06, 7
ProviderStationName	Not visible string according to RFC 5890	RSP-, 0xDB, 0x81, 0x06, 8
Padding	Not checked	—

5.2.27.2.8 ARRPCBlockReq

This field shall be checked according to Table 671.

Table 671 – ARPCBlockReq – request check

Parameter	Checking rules	Behavior on match
BlockType	$\neq 0x0107$	—
BlockLength	$\neq 4$	RSP-, 0xDB, 0x81, 0x07, 1
BlockVersionHigh	$\neq 0x01$	RSP-, 0xDB, 0x81, 0x07, 2
BlockVersionLow	$\neq 0x01$	RSP-, 0xDB, 0x81, 0x07, 3
InitiatorRPCServerPort	$\neq 0x0400$ to $0xFFFF$	RSP-, 0xDB, 0x81, 0x07, 4

5.2.27.2.9 ARVendorBlockReq

This field shall be checked regarding definitions to BlockType 0x0108. No detailed checking rules are defined.

5.2.27.2.10 IRInfoBlock

This field shall be checked according to Table 672.

Table 672 – IRInfoBlock – request check

Parameter	Checking rules	Behavior on match
BlockType	$\neq 0x0109$	—
BlockLength	$\neq 1 + 1 + 2(\text{Padding}) + 16 + 2(\text{Padding}) + 2 + \text{NumberOfIOCRs} * (2 + 2 + 4)$	RSP-, 0xDB, 0x81, 0x09, 1
BlockVersionHigh	$\neq 0x01$	RSP-, 0xDB, 0x81, 0x09, 2
BlockVersionLow	$\neq 0x00$	RSP-, 0xDB, 0x81, 0x09, 3
Padding	Not checked	—
IRDataUUID	$== 0$	RSP-, 0xDB, 0x81, 0x09, 5
Padding	Not checked	—
NumberOfIOCRs	$\neq 0$ OR 2	RSP-, 0xDB, 0x81, 0x09, 7
IOCRReference	no matching IOCRBlockReq or not unique	RSP-, 0xDB, 0x81, 0x09, 8
SubframeOffset	$\geq \text{DataLength} - (2 + 2)$	RSP-, 0xDB, 0x81, 0x09, 9
SubframeData	(SubframeData.Position == 0) OR (SubframeData.DataLength == 0) OR (SubframeOffset + SubframeData.DataLength > DataLength - (2 + 2))	RSP-, 0xDB, 0x81, 0x09, 10

5.2.27.2.11 SRInfoBlock

This field shall be checked according to Table 673.

Table 673 – SRInfoBlock – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x010A	—
BlockLength	!= 2 + 2 + 4	RSP-, 0xDB, 0x81, 0x0A, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDB, 0x81, 0x0A, 2
BlockVersionLow	!= 0x00	RSP-, 0xDB, 0x81, 0x0A, 3
RedundancyDataHoldFactor	(== 0 to 2 OR not supported in GSDML)	RSP-, 0xDB, 0x81, 0x0A, 4
SRProperties. InputValidOnBackupAR	Not checked	—
SRProperties. ActivateRedundancyAlarm	Not checked	—
SRProperties. Reserved_1	!= 0	RSP-, 0xDB, 0x81, 0x0A, 5
SRProperties. Reserved_2	Not checked	—

5.2.27.2.12 ARFSUBBlock

This field shall be checked regarding definitions to BlockType 0x010B. No detailed checking rules are defined.

5.2.27.3 IODConnectRes

5.2.27.3.1 ArgsLength

This field shall be checked according to Table 674.

Table 674 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	Okay, check IODConnectRes-PDU
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRData-Response	ArgsLength	!= Sum of all BlockLength + Number of Blocks × 4	CMCTL: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the response	CMCTL: Unknown blocks
IODConnectRes	—	—	—

5.2.27.3.2 ARBlockRes

This field shall be checked according to Table 675.

Table 675 – ARBlockRes – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8101	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 30	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARType	!= ARBlockReq.ARTYPE	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARUUID	!= ARBlockReq.ARUUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	== ARBlockReq.SessionKey	n. a.
SessionKey	!= ARBlockReq.SessionKey	ignore
CMResponderMacAdd	Is not unicast	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ResponderUDPRTPort	If (IOCRProperties.RTClass == RT_CLASS_UDP) == 0 to 0x03FF else ignore	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.3.3 IOCRBlockRes

This field shall be checked according to Table 676.

Table 676 – IOCRBlockRes – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8102	n. a.
BlockLength	!= 8	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
IOCRType	!= IOCRBlockReq.IOCRType	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
IOCRReference	!= IOCRBlockReq.IOCRReference	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

Parameter	Checking rules	Behavior on match
FrameID	If (IOCRBlockReq.IOCRType == "Input CR" OR IOCRBlockReq.IOCRType == "Multicast Consumer CR") AND IOCRBlockReq.FrameID != FrameID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
FrameID	If (IOCRBlockReq.IOCRType == "Output CR") AND ((IOCRProperties.RTClass == 0x01 AND FrameID!=0xC000 to 0xF7FF) OR (IOCRProperties.RTClass == 0x02 AND FrameID!=0x8000 to 0xBBFF) OR (IOCRProperties.RTClass == 0x03 AND FrameID!=0x0100 to 0x7FFF AND IOCRBlockReq.FrameID != FrameID) OR (IOCRProperties.RTClass == 0x04 AND FrameID!=0xC000 to 0xF7FF))	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.3.4 AlarmCRBlockRes

This field shall be checked according to Table 677.

Table 677 – AlarmCRBlockRes – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8103	—
BlockLength	!= 8	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
AlarmCRType	!= 0x0001	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
LocalAlarmReference	Not checked	—
MaxAlarmDataLength	!= 200 to 1 432	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.3.5 ModuleDiffBlock

This field shall be checked according to Table 678.

NOTE For legacy reasons, a ModuleDiffBlock containing NumberOfAPIs == 1, API == 0 and NumberOfModules == 0, may be seen in Connect.rsp or ApplRdy.req.

Table 678 – ModuleDiffBlock – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8104	—
BlockLength	!= 4 + NumberOfAPI × (6 + NumberOfModules × (10 + NumberOfSubmodules × 8))	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
NumberOfAPIs	== 0	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
API	!= ExpectedSubmoduleBlockReq.API	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
NumberOfModules	== 0	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SlotNumber	!= ExpectedSubmoduleBlockReq.SlotNumber	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ModuleState	== 0x0 to 0x0003	—
ModuleState	!= 0x0 to 0x0003	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
NumberOfSubmodules	Not checked	—
SubslotNumber	!= ExpectedSubmoduleBlockReq.SubslotNumber	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SubmoduleIdentNumber	Not checked	—
SubmoduleState.AddInfo	Not checked	—
SubmoduleState.QualifiedInfo	Not checked	—
SubmoduleState.Maintenance-Required	Not checked	—
SubmoduleState.Maintenance-Demanded	Not checked	—
SubmoduleState.DiagInfo	Not checked	—
SubmoduleState.ARInfo	If SubmoduleState.FormatIndicator == 1 != 0x0 to 0x04	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SubmoduleState.IdentInfo	If SubmoduleState.FormatIndicator == 1 != 0x0 to 0x03 and Table 457	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SubmoduleState.FormatIndicator	Not checked	—
SubmoduleState.Detail	If SubmoduleState.FormatIndicator == 0 != (0x0 to 0x02 OR 0x04 OR 0x07)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.3.6 PrmServerBlockRes

This field shall be checked regarding definitions to BlockType 0x8105. No additional checking rules are defined.

5.2.27.3.7 ARServerBlockRes

This field shall be checked according to Table 679.

Table 679 – ARServerBlockRes – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8106	—
BlockLength	!= Calculated length	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
StationNameLength	!= Length of the CMResponderStationName	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
CMResponderStationName	!= Local stored / expected CMResponderStationName	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.3.8 RRPCBlockRes

This field shall be checked regarding definitions to BlockType 0x8107. No additional checking rules are defined.

5.2.27.3.9 ARVendorBlockRes

This field shall be checked regarding definitions to BlockType 0x8108. No additional checking rules are defined.

5.2.27.4 IODControlReq

5.2.27.4.1 ArgsLength

This field shall be checked according to Table 680.

Table 680 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	Okay, check IODControlReq-PDU
	RPCOperationNmb	== 5 (Read Implicit)	—

	Parameter	Checking rules	Behavior on match
NDRDataRequest	ArgsLength	!= Sum of all BlockLength + Number of Blocks × 4	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the response	CMRPC/CMINA: Unknown blocks
IODControlReq	—	—	—

5.2.27.4.2 ControlBlockConnect request

This field shall be checked according to Table 681.

Table 681 – ControlBlockConnect(PrmEnd) – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0110	RSP-, 0xDD, 0x81, 0x14, 0
BlockLength	!= 28	RSP-, 0xDD, 0x81, 0x14, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDD, 0x81, 0x14, 2
BlockVersionLow	!= 0x00	RSP-, 0xDD, 0x81, 0x14, 3
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x14, 4
ARUUID	!= ARBlockReq.ARUUID	CMRPC/CMINA: AR UUID unknown
SessionKey	!= ARBlockReq.SessionKey	RSP-, 0xDD, 0x81, 0x14, 6
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x14, 7
ControlCommand	!= 0x0001 (PrmEnd)	RSP-, 0xDD, 0x81, 0x14, 8
ControlBlockProperties.reserved	!= 0x0000	RSP-, 0xDD, 0x81, 0x14, 9

5.2.27.4.3 ControlBlockPlug

This field shall be checked according to Table 682.

Table 682 – ControlBlockPlug(PrmEnd) – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0111	RSP-, 0xDD, 0x81, 0x15, 0
BlockLength	!= 28	RSP-, 0xDD, 0x81, 0x15, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDD, 0x81, 0x15, 2
BlockVersionLow	!= 0x00	RSP-, 0xDD, 0x81, 0x15, 3
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x15, 4
ARUUID	!= ARBlockReq.ARUUID	CMRPC/CMINA: AR UUID unknown
SessionKey	!= ARBlockReq.SessionKey	RSP-, 0xDD, 0x81, 0x15, 6
AlarmSequenceNumber	!= AlarmSpecifier.SequenceNumber of corresponding AlarmNotification-PDU	RSP-, 0xDD, 0x81, 0x15, 7
ControlCommand	!= 0x0001 (PrmEnd)	RSP-, 0xDD, 0x81, 0x15, 8
ControlBlockProperties.reserved	!= 0x0000	RSP-, 0xDD, 0x81, 0x15, 9

5.2.27.4.4 ControlBlockConnect request

This field shall be checked according to Table 683.

Table 683 – ControlBlockConnect(PrmBegin) – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0118	RSP-, 0xDD, 0x81, 0x18, 0
BlockLength	!= 28	RSP-, 0xDD, 0x81, 0x18, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDD, 0x81, 0x18, 2
BlockVersionLow	!= 0x00	RSP-, 0xDD, 0x81, 0x18, 3
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x18, 4
ARUUID	!= ARBlockReq.ARUUID	CMRPC/CMINA: AR UUID unknown
SessionKey	!= ARBlockReq.SessionKey	RSP-, 0xDD, 0x81, 0x18, 6
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x18, 7
ControlCommand	!= 0x0040 (PrmBegin)	RSP-, 0xDD, 0x81, 0x18, 8
ControlBlockProperties.reserved	!= 0x0000	RSP-, 0xDD, 0x81, 0x18, 9

5.2.27.4.5 SubmoduleListBlock

This field shall be checked according to Table 684.

Table 684 – SubmoduleListBlock – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0119	RSP-, 0xDD, 0x81, 0x19, 0
BlockLength	!= Calculated length	RSP-, 0xDD, 0x81, 0x19, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDD, 0x81, 0x19, 2
BlockVersionLow	!= 0x00	RSP-, 0xDD, 0x81, 0x19, 3
NumberOfEntries	!= Fitting to the BlockLength	RSP-, 0xDD, 0x81, 0x19, 4
API	!= Fitting to the ExpectedIdentification	RSP-, 0xDD, 0x81, 0x19, 5
SlotNumber	!= Fitting to the ExpectedIdentification	RSP-, 0xDD, 0x81, 0x19, 6
SubslotNumber	!= Fitting to the ExpectedIdentification	RSP-, 0xDD, 0x81, 0x19, 7

5.2.27.5 IODControlRes

5.2.27.5.1 ArgsLength

This field shall be checked according to Table 685.

Table 685 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—

	Parameter	Checking rules	Behavior on match
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	Okay, check IODControlRes-PDU
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRDataResponse	ArgsLength	!= Sum of all BlockLength + Number of Blocks × 4	CMCTL: ArgsLength invalid
	ArgsLength	General: If one OR more unknown blocks are contained in the response	CMCTL: Unknown blocks
IODControlRes	—	—	—

5.2.27.5.2 ControlBlockConnect

This field shall be checked according to Table 686.

Table 686 – ControlBlockConnect – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8110	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 28	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARUID	!= IODControlReq.ControlBlockConnect. ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	!= IODControlReq.ControlBlockConnect. SessionKey	Ignore
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlCommand	!= 0x0008 (Done)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlBlockProperties.reserved	!= 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.5.3 ControlBlockPlug

This field shall be checked according to Table 687.

Table 687 – ControlBlockPlug – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8111	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 28	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARUID	!= IODControlReq.ControlBlockPlug. ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	!= IODControlReq.ControlBlockPlug. SessionKey	Ignore
AlarmSequenceNumber	!= AlarmSpecifier.SequenceNumber of corresponding AlarmNotification-PDU	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlCommand	!= 0x0008 (Done)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlBlockProperties.reserved	!= 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.5.4 ControlBlockConnect

This field shall be checked according to Table 688.

Table 688 – ControlBlockConnect(PrmBegin) – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8118	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 28	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

Parameter	Checking rules	Behavior on match
ARUID	$\neq$ IODControlReq.ControlBlockConnect.PrmBegin.ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	$\neq$ IODControlReq.ControlBlockConnect.PrmBegin.SessionKey	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	$\neq$ 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlCommand	$\neq$ 0x0008 (Done)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlBlockProperties.reserved	$\neq$ 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.6 IOXControlReq

5.2.27.6.1 ArgsLength

This field shall be checked according to Table 689.

Table 689 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	$==$ 0 (Connect)	—
	RPCOperationNmb	$==$ 1 (Release)	—
	RPCOperationNmb	$==$ 2 (Read)	—
	RPCOperationNmb	$==$ 3 (Write)	—
	RPCOperationNmb	$==$ 4 (Control)	Okay, check IOXControlReq-PDU
	RPCOperationNmb	$==$ 5 (Read Implicit)	—
NDRDataRequest	ArgsLength	$\neq$ Sum of all BlockLength + Number of Blocks $\times$ 4	CMRPC / CTLRPC: ArgsLength invalid
	ArgsLength	General: If one OR more unknown blocks are contained in the request	CMRPC / CTLRPC: Unknown blocks
IOXControlReq	—	—	—

5.2.27.6.2 ControlBlockConnect

This field shall be checked according to Table 690.

Table 690 – ControlBlockConnect(AppIRdy) – request check

Parameter	Checking rules	Behavior on match
BlockType	$\neq$ 0x0112	RSP-, 0xDD, 0x81, 0x16, 0
BlockLength	$\neq$ 28	RSP-, 0xDD, 0x81, 0x16, 1
BlockVersionHigh	$\neq$ 0x01	RSP-, 0xDD, 0x81, 0x16, 2
BlockVersionLow	$\neq$ 0x00	RSP-, 0xDD, 0x81, 0x16, 3
Padding	$\neq$ 0x00	RSP-, 0xDD, 0x81, 0x16, 4

Parameter	Checking rules	Behavior on match
ARUID	!= ARBlockReq.ARUID	CMRPC/CMINA: AR UID unknown
SessionKey	!= ARBlockReq.SessionKey	RSP-, 0xDD, 0x81, 0x16, 6
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x16, 7
ControlCommand	!= 0x0002 (ApplicationReady)	RSP-, 0xDD, 0x81, 0x16, 8
ControlBlockProperties.reserved	!= 0x0000	RSP-, 0xDD, 0x81, 0x16, 9

5.2.27.6.3 ControlBlockPlug

This field shall be checked according to Table 691.

Table 691 – ControlBlockPlug(AppIRdy) – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0113	RSP-, 0xDD, 0x81, 0x17, 0
BlockLength	!= 28	RSP-, 0xDD, 0x81, 0x17, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDD, 0x81, 0x17, 2
BlockVersionLow	!= 0x00	RSP-, 0xDD, 0x81, 0x17, 3
Padding	!= 0x00	RSP-, 0xDD, 0x81, 0x17, 4
ARUID	!= ARBlockReq.ARUID	CMRPC/CMINA: AR UID unknown
SessionKey	!= ARBlockReq.SessionKey	RSP-, 0xDD, 0x81, 0x17, 6
AlarmSequenceNumber	!= AlarmSpecifier.SequenceNumber of corresponding AlarmNotification-PDU	RSP-, 0xDD, 0x81, 0x17, 7
ControlCommand	!= 0x0002 (ApplicationReady)	RSP-, 0xDD, 0x81, 0x17, 8
ControlBlockProperties.reserved	!= 0x0000	RSP-, 0xDD, 0x81, 0x17, 9

5.2.27.6.4 ExpectedSubmoduleBlockReq

This field shall be checked according to Table 668.

5.2.27.6.5 ModuleDiffBlock

This field shall be checked according to Table 678 using the ErrorCodes from Table 480 row “Faulty ModuleDiffBlock”.

5.2.27.7 IOXControlRes

5.2.27.7.1 ArgsLength

This field shall be checked according to Table 692.

Table 692 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—

	Parameter	Checking rules	Behavior on match
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	Okay, check IOXControlRes-PDU
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRDataResponse	ArgsLength	!= Sum of all BlockLength + Number of Blocks × 4	CMRPC/CMINA: ArgsLength invalid
	ArgsLength	General: If one OR more unknown blocks are contained in the response	CMRPC/CMINA: Unknown blocks
IOXControlRes	—	—	—

5.2.27.7.2 ControlBlockConnect

This field shall be checked according to Table 693.

Table 693 – ControlBlockConnect – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8112	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 28	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARUID	!= IOXControlReq.ControlBlockConnect. ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	!= IOXControlReq.ControlBlockConnect. SessionKey	ignore
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlCommand	!= 0x0008 (Done)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlBlockProperties.reserved	!= 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.7.3 ControlBlockPlug

This field shall be checked according to Table 694.

Table 694 – ControlBlockPlug – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8113	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 28	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARUUID	!= IOXControlReq.ControlBlockPlug. ARUUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	!= IOXControlReq.ControlBlockPlug. SessionKey	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
AlarmSequenceNumber	!= AlarmSpecifier.SequenceNumber of corresponding AlarmNotification-PDU	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlCommand	!= 0x0008 (Done)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlBlockProperties.reserved	!= 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.8 IODReleaseReq**5.2.27.8.1 ArgsLength**

This field shall be checked according to Table 695.

Table 695 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	Okay, check IODReleaseReq-PDU
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRDataRequest	ArgsLength	!= ReleaseBlock.BlockLength + 4	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the request	CMRPC/CMINA: Unknown blocks
IODReleaseReq	—	—	—

5.2.27.8.2 ReleaseBlock

This field shall be checked according to Table 696.

Table 696 – ReleaseBlock – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0114	RSP-, 0xDC, 0x81, 0x28, 0
BlockLength	!= 28	RSP-, 0xDC, 0x81, 0x28, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDC, 0x81, 0x28, 2
BlockVersionLow	!= 0x00	RSP-, 0xDC, 0x81, 0x28, 3
Padding	!= 0x00	RSP-, 0xDC, 0x81, 0x28, 4
ARUUID	!= ARBlockReq.ARUUID	CMRPC/CMINA: AR UUID unknown
SessionKey	!= ARBlockReq.SessionKey	RSP-, 0xDC, 0x81, 0x28, 6
Padding	!= 0x00	RSP-, 0xDC, 0x81, 0x28, 7
ControlCommand	!= 0x0004 (Release)	RSP-, 0xDC, 0x81, 0x28, 8
ControlBlockProperties.reserved	!= 0x0000	RSP-, 0xDC, 0x81, 0x28, 9

5.2.27.9 IODReleaseRes**5.2.27.9.1 ArgsLength**

This field shall be checked according to Table 697.

Table 697 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	Okay, check IODReleaseRes-PDU
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRDataResponse	ArgsLength	!= ReleaseBlock.BlockLength + 4	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the response	CMRPC/CMINA: Unknown blocks
IODReleaseRes	—	—	—

5.2.27.9.2 ReleaseBlock

This field shall be checked according to Table 698.

Table 698 – ReleaseBlock – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8114	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 28	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ARUID	!= IODReleaseReq.ReleaseBlock.ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SessionKey	!= IODReleaseReq.ReleaseBlock.SessionKey	ignore
Padding	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlCommand	!= 0x0008 (Done)	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
ControlBlockProperties.reserved	!= 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT

5.2.27.10 IODWriteReq

5.2.27.10.1 ArgsLength

This field shall be checked according to Table 699.

Table 699 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	Okay, check IODWriteReq-PDU
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	—
NDRDataRequest	ArgsLength	!= (IODWriteReqHeader. BlockLength + 4 + IODWriteReqHeader. RecordDataLength)	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the request	CMRPC/CMINA: Unknown blocks
IODWriteReq	—	—	—

5.2.27.10.2 IODWriteReqHeader

This field shall be checked according to Table 700.

Table 700 – IODWriteReqHeader – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0008	RSP-, 0xDF, 0x81, 0x08, 0
BlockLength	!= 60	RSP-, 0xDF, 0x81, 0x08, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDF, 0x81, 0x08, 2
BlockVersionLow	!= 0x00	RSP-, 0xDF, 0x81, 0x08, 3
SeqNumber	Not checked	—
ARUID	!= ARBlockReq.ARUID	CMRPC/CMINA: AR UID unknown
API	> 0x0 AND not supported in GSDML	RSP-, 0xDF, 0x80, 0xB4, 6
SlotNumber	!= 0x0 to 0x7FFF OR not supported in GSDML	RSP-, 0xDF, 0x80, 0xB2, 7
SubslotNumber	!= 0x0 to 0x8FFF OR not supported in GSDML	RSP-, 0xDF, 0x80, 0xB2, 8
Padding	!= 0x0000	RSP-, 0xDF, 0x80, 0xB7, 9
Index	Not supported by application	RSP-, 0xDF, 0x80, 0xB0, 10
RecordDataLength	!= consistent with ArgsLength	RSP-, 0xDF, 0x81, 0x08, 11
RWPadding	Not checked	—

5.2.27.10.3 RecordDataWrite

This field shall be not checked.

5.2.27.11 IODWriteRes**5.2.27.11.1 ArgsLength**

This field shall be checked according to Table 701.

Table 701 – ArgsLength check

	Parameter	Checking rules	Behavior on match
NDRDataResponse	ArgsLength	!= IODWriteResHeader.BlockLength + 4	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
	—	General: If one OR more unknown blocks are contained in the response	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
IODReleaseRes	—	—	—

5.2.27.11.2 IODWriteResHeader

This field shall be checked according to Table 702.

Table 702 – IODWriteResHeader – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8008	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 60	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SeqNumber	Not checked	—
ARUID	!= IODWriteReq.IODWriteReqHeader. ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
API	!= IODWriteReq.IODWriteReqHeader.API	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SlotNumber	!= IODWriteReq.IODWriteReqHeader.SlotNumber	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SubslotNumber	!= IODWriteReq.IODWriteReqHeader.SubslotNumber	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Index	!= IODWriteReq.IODWriteReqHeader.Index	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
RecordDataLength	Not checked	—
AdditionalValue1	Not checked	—
AdditionalValue2	Not checked	—
PNIOStatus	Not checked	—
RWPadding	Not checked	—

5.2.27.12 IODWriteMultipleReq

5.2.27.12.1 ArgsLength

This field shall be checked according to Table 703.

Table 703 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	—
	RPCOperationNmb	== 3 (Write)	Okay, check IODWriteMultipleReq-PDU
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	—

	Parameter	Checking rules	Behavior on match
NDRDataRequest	ArgsLength	$!= \text{IODWriteReqHeader.BlockLength} + 4 + \text{Sum of all IODWriteReqHeader.BlockLength} + \text{Number of WriteBlocks} \times 4$	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the request	CMRPC/CMINA: Unknown blocks
IODWriteReq	—	—	—

5.2.27.12.2 IODWriteReqHeader (multiple)

This field shall be checked according to Table 700. The value of the parameter index shall be 0xE040.

5.2.27.12.3 IODWriteReqHeader

This field shall be checked according to Table 700.

5.2.27.12.4 RecordDataWrite

This field shall be not checked.

5.2.27.12.5 Padding

The number of padding octets shall be 0, 1, 2 and 3 to have 32 bit alignment to the next IODWriteReqHeader. The value of the padding octets is not checked.

5.2.27.13 IODWriteMultipleRes

5.2.27.13.1 ArgsLength

This field shall be checked according to Table 704.

Table 704 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	$== 0$ (Connect)	—
	RPCOperationNmb	$== 1$ (Release)	—
	RPCOperationNmb	$== 2$ (Read)	—
	RPCOperationNmb	$== 3$ (Write)	Okay, check IODWriteMultipleRes-PDU
	RPCOperationNmb	$== 4$ (Control)	—
	RPCOperationNmb	$== 5$ (Read Implicit)	—
NDRData-Response	ArgsLength	$!= \text{IODWriteResHeader.BlockLength} + 4 + \text{Sum of all IODWriteResHeader.BlockLength} + \text{Number of IODWriteResHeaderBlocks} \times 4$	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the response	CMRPC/CMINA: Unknown blocks
IODWriteMultiple-Res	—	—	—

5.2.27.13.2 IODWriteResHeader (multiple)

This field shall be checked according to Table 702. The value of the parameter index shall be 0xE040.

5.2.27.13.3 IODWriteResHeader

This field shall be checked according to Table 702.

5.2.27.14 IODReadReq

5.2.27.14.1 ArgsLength

This field shall be checked according to Table 705.

Table 705 – ArgsLength check

	Parameter	Checking rules	Behavior on match
RPCHeader	RPCOperationNmb	== 0 (Connect)	—
	RPCOperationNmb	== 1 (Release)	—
	RPCOperationNmb	== 2 (Read)	Okay, check IODReadReq-PDU
	RPCOperationNmb	== 3 (Write)	—
	RPCOperationNmb	== 4 (Control)	—
	RPCOperationNmb	== 5 (Read Implicit)	Okay, check IODReadReq-PDU
NDRDataRequest	ArgsLength	!= IODReadReqHeader.BlockLength + 4 OR != IODReadReqHeader.BlockLength + 4 + RecordDataReadQuery.BlockLength + 4	CMRPC/CMINA: ArgsLength invalid
	—	General: If one OR more unknown blocks are contained in the request	CMRPC/CMINA: Unknown blocks
IOReadReq	—	—	—

5.2.27.14.2 IODReadReqHeader

This field shall be checked according to Table 706.

Table 706 – IODReadReqHeader – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0009	RSP-, 0xDE, 0x81, 0x08, 0
BlockLength	!= 60	RSP-, 0xDE, 0x81, 0x08, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDE, 0x81, 0x08, 2
BlockVersionLow	!= 0x00	RSP-, 0xDE, 0x81, 0x08, 3
SeqNumber	Not checked	—
ARUUID	Case RPCOperationNmb == 2 (Read): != ARBlockReq.ARUUID	CMRPC/CMINA: AR UUID unknown

Parameter	Checking rules	Behavior on match
ARUID	Case RPCOperationNmb == 5 (Read Implicit): != NIL	RSP-, 0xDE, 0x81, 0x08, 5
API	> 0x0 AND not supported in GSDML	RSP-, 0xDE, 0x80, 0xB4, 6
SlotNumber	!= 0x0 to 0x7FFF OR not supported in GSDML	RSP-, 0xDE, 0x80, 0xB2, 7
SubslotNumber	!= 0x0 to 0x8FFF OR not supported in GSDML	RSP-, 0xDE, 0x80, 0xB2, 8
Padding	!= 0x0000	RSP-, 0xDE, 0x80, 0xB7, 9
Index	Not supported by application	RSP-, 0xDE, 0x80, 0xB0, 10
RecordDataLength	Not consistent with ArgsMaximum	RSP-, 0xDE, 0x81, 0x08, 11
TargetARUID	Case RPCOperationNmb == 2 (Read): TargetARUID != NIL	RSP-, 0xDE, 0x81, 0x08, 12
RWPadding	Not checked	—

5.2.27.14.3 RecordDataReadQuery

This field shall be checked according to Table 707.

Table 707 – RecordDataReadQuery – request check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x0500 OR not supported in GSDML	CMRPC/CMINA: Unknown blocks
BlockLength	< 0x3 OR not consistent with ArgsLength	RSP-, 0xDE, 0x81, 0x08, 1
BlockVersionHigh	!= 0x01	RSP-, 0xDE, 0x81, 0x08, 2
BlockVersionLow	!= 0x00	RSP-, 0xDE, 0x81, 0x08, 3
Data	Not checked	—

5.2.27.15 IODReadRes

5.2.27.15.1 ArgsLength

This field shall be checked according to Table 708.

Table 708 – ArgsLength check

	Parameter	Checking rules	Behavior on match
NDRDataResponse	ArgsLength	!= IODReadResHeader.BlockLength + 4 + IODReadResHeader.RecordDataLength	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
	—	General: If one OR more unknown blocks are contained in the response	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
IODReadRes	—	—	—

5.2.27.15.2 IODReadResHeader

This field shall be checked according to Table 709.

Table 709 – IODReadResHeader – response check

Parameter	Checking rules	Behavior on match
BlockType	!= 0x8009	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockLength	!= 60	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionHigh	!= 0x01	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
BlockVersionLow	!= 0x00	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SeqNumber	Not checked	—
ARUID	!= IODReadReqHeader.ARUID	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
API	!= IODReadReqHeader.API	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SlotNumber	!= IODReadReqHeader.SlotNumber	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
SubslotNumber	!= IODReadReqHeader.SubslotNumber	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Padding	!= 0x0000	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
Index	!= IODReadReqHeader.Index	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
RecordDataLength	Not consistent with ArgsLength	ERR-RTA, RTA_ERR_CLS_PROTOCOL, RTA_ERR_ABORT
AdditionalValue1	Not checked	—
AdditionalValue2	Not checked	—
RWPPadding	Not checked	—

5.2.27.15.3 RecordDataRead

This field shall be not checked.

5.3 FAL protocol state machines**5.3.1 Overall structure****5.3.1.1 Overview**

The FAL protocol state machine structure is as defined in Figure 66. The general structure is according to the IEC 61158-6 series protocol machine model.

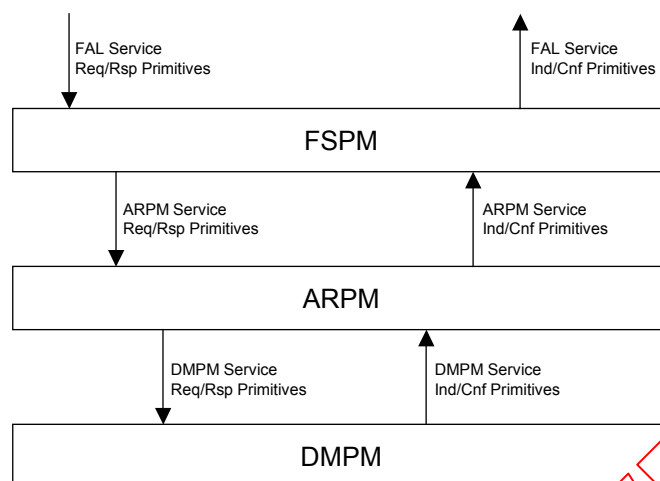


Figure 95 – Relationship among Protocol Machines

The behavior of the FAL is specified by three integrated protocol machines. The FSPM is the service interface between the FAL services that are part of the FAL Class specification and the particular AREP.

The FAL provides two kinds of protocol machine architectures, one for IO controller and one for IO devices.

The FSPM is responsible for the following activities:

- To accept service primitives from the FAL service user and convert them into FAL internal primitives.
- To select the ARPM state machine based on the implicit addressing mechanism and send FAL internal primitives with the service parameters to the ARPM.
- To accept FAL internal primitives from the ARPM and convert them into service primitives for the FAL service user.
- To deliver the FAL service primitives to the FAL user.

The ARPM specifies the conveyance type for the application relation.

The DMPM specifies the mapping to the Data Link Layer. The DMPM defines therefore two protocol machines, the LMPM and the MAC protocol machines.

5.3.1.2 FAL Service Protocol Machines

The FSPM State Machines co-ordinate the underlying state machines used for processing of the various services and application relations.

The FSPM basically is a mapping protocol machine. The main task is to pass the service to the protocol machine responsible for that service and forward confirmations and responses to the user. In addition a basic redundancy control scheme is included that allow to collaborate two or four AR into a single entity with higher availability.

For this task, the following support shall be offered by an FSPM:

- Support of an AR set (only for redundant devices)
- Coordination of the Primary / Backup APDU_Status.DataStatus.State (information)
- Coordination of the Primary / Backup failure situation using APDU_Status.DataStatus.Redundancy (information)

- Issuing switchover procedures (signal Primary request in case of an IO-Device)
- Collaboration of the set and get data services of the AR set

5.3.1.3 IO controller to IO device

The PPM, CPM, ALPMI, ALPMR, APMR, APMS, CMDEV, CMCTL, CMRPC, CMINA, CTRLRPC, CTLDINA State Machines are responsible for the cyclic, acyclic and alarm data transfer between an IO controller and an IO device. These are used to establish an IO AR.

5.3.1.4 IO supervisor to IO device

The PPM, CPM, ALPMI, ALPMR, APMR, APMS, CMDEV, CMCTL, CMRPC, CMINA, CTRLRPC, CTLDINA State Machines are responsible for the cyclic, acyclic and alarm data transfer between an IO supervisor and an IO device. These are used to establish a Supervisor AR.

5.3.1.5 DLL Mapping Protocol Machines

The DL Mapping Protocol Machines (DMPM) connects the other State machines and Layer 2. DMPM provides the coordination of all state machines concerning the configuration and error handling of the Data Link Layer Usage. The functions are mapped by the DMPM to the DLL services of Layer 2. The DMPM generates the necessary Layer 2 parameters of the service, receives the confirmations and indications from Layer 2 and passes them to the appropriate DMPM-User.

The DMPM is directly put upon the DLL User – DLL interface as described in Data Link Layer Service Definitions (Refer to Enhanced Internal Sublayer Services IEEE 802.3).

The DMPM uses for the mapping of the DMPM functions on the Layer 2 the following services:

- MA_UNITDATA.req
- MA_UNITDATA.ind

For the purpose of this specification there is an explicit confirmation for the MA_UNITDATA request service primitive indicating that the DLPDU has been transmitted.

5.4 AP-Context state machine

There is no AP-Context State Machine defined for this Protocol.

5.5 FAL Service Protocol Machines

5.5.1 Overview

This type specifies one FAL Service Protocol Machine (FSPM) state machine to transfer the FAL user service primitives into FAL internal service primitives and vice versa. There are two kinds of FSPM defined, one for the IO device (FSPMDEV) and one for the IO controller (FSPMCTL).

5.5.2 FAL Service Protocol Machine Device

5.5.2.1 Overview

The FSPMDEV is responsible for the startup of the underlying protocol machines (LMPM, LLDP, DHCP/DCP, IP, UDP, RPC, CMDEV, CMCTL, DCP, PTCP, ALPM, PPM, CPM...).

5.5.2.2 Primitive definitions

Table 710 shows the service primitives including their associated parameters issued by the AP-Context (FAL user) and received by the FSPMDEV with the mapping to underlying services.

Table 710 – Primitives issued by AP-Context (FAL user) to FSPMDEV

Primitive name	Associated parameters	Primitive mapped to	Parameter mapped to
local AR Abort.req	AREP	CM_Abort_req (AREP)	AREP=AREP
local Create LogBook Entry	info	CreateLogBookEntry (info)	info = info
local Get Input IOCS.req	AREP CREP Slot Number Subslot Number	CPM_Get_Data_req (CREP)	MAP_GINIOCS_REQ
local Get Output.req	AREP CREP Slot Number Subslot Number	CPM_Get_Data_req (CREP), CPM_Get_Status_req (CREP)	CREP=CREP
local Set AR State.req	AREP ARState	Set_Variable (Arstate)	AREP=AREP ARState=ARState
local Set Command.rsp (+)	List Of Response	DCP_Set_rsp (+)	CREP DA ListOfResponse = List Of Response
local Set Command.rsp (-)	ERRCLS ERRCODE	DCP_Set_rsp (-)	CREP DA ERRCLS = ERRCLS ERRCODE = ERRCODE
local Set Input.req	AREP CREP Slot Number Subslot Number IOPS Subslot Input Data	PPM_Set_Data_req (CREP, Data)	MAP_SIN_REQ
local Set Output IOCS.req	AREP CREP Slot Number Subslot Number IOCS	PPM_Set_Data_req (CREP, Data)	MAP_OIOCS_REQ
local Set Provider State.req	AREP ProviderState Flag	PPM_Set_Status_req (CREP, D_Status)	MAP_SPS_REQ
local Set Redundancy.req	AREP Redundancy Flag	PPM_Set_Status_req (CREP, D_Status)	MAP_SRS_REQ
local Set State.req	AREP State Flag	PPM_Set_Status_req (CREP, D_Status)	MAP_SSS_REQ
Application Ready.req	AREP Session Key Alarm Sequence Number Module Diff Block	CM_Ccontrol.req (AREP, ControlBlock, ModuleDiffBlock)	MAP_AREADY_REQ
Connect.rsp (-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Connect.rsp (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_CON_RSP-

Primitive name	Associated parameters	Primitive mapped to	Parameter mapped to
Connect.rsp (+)	AREP AR Response Block List of IO CR Response Blocks Alarm CR Response Block Module Diff Block AR RPC Block AR Vendor Block	CM_Connect.rsp (+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock, ARRPCBlock, ARVendorBlock)	MAP_CON_RSP+
Prm Begin.rsp (-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Dcontrol.rsp (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_BOP_RSP-
Prm Begin.rsp (+)	AREP Session Key	CM_Dcontrol.rsp (+) (AREP, ControlBlock)	MAP_BOP_RSP+
Prm End.rsp (-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Dcontrol.rsp (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_EOP_RSP-
Prm End.rsp (+)	AREP Session Key Alarm Sequence Number	CM_Dcontrol.rsp (+) (AREP, ControlBlock)	MAP_EOP_RSP+
Read.rsp (-)	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Read.rsp (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_RSP-
Read.rsp (+)	AREP Seq Number Length Data	CM_Read.rsp (+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RSP+
Write.rsp (-)	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Write.rsp (-) (AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_RSP-
Write.rsp (+)	AREP Seq Number	CM_Write.rsp (+) (AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_RSP+
Alarm.req	AREP API Alarm Priority Alarm Type Slot Number Subslot Number Alarm Specifier Module Ident Number Submodule Ident Number	ALPMI_Alarm_Notification.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, Alarm_User_Data_Structure_Identifier,	MAP_AN_REQ

Primitive name	Associated parameters	Primitive mapped to	Parameter mapped to
	Alarm Item	Alarm_User_Data)	
Alarm.rsp (-)	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier PNIO Status	ALPMR_Alarm_Ack.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_ALARM_RSP-
Alarm.rsp (+)	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier PNIO Status	ALPMR_Alarm_Ack.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_ALARM_RSP+

Table 711 shows the service primitives including their associated parameters issued by the FSPMDEV and received by the AP-Context (FAL user).

Table 711 – Primitives issued by FSPMDEV to AP-Context (FAL user)

Primitive mapped from	Parameter mapped from	Primitive name	Associated parameters
CM_Abort_cnf (-)	AREP=AREP	local AR Abort.cnf (-)	AREP
CM_Abort_cnf (+)	AREP=AREP	local AR Abort.cnf (+)	AREP
CMDEV_state_ind (state, SessionKey)	if (state == ABORT) then AREP=AREP SessionKey=SessionKey else ignore	local AR Abort.ind	AREP, Session Key
CM_Release.ind (AREP, ReleaseBlock)	MAP_REL_IND	local AR Abort.ind	AREP Session Key
CMDEV_state_ind (state)	if (state == DATA) then AREP=AREP else ignore	local AR In Data.ind	AREP
CPM_NewData_ind (AREP, CREP, APDU_Status)	MAP_NEWSTATUS_IND	local Data State Changed.ind	AREP CREP DataValid Flag State Flag Redundancy Flag ProviderState Flag ProblemIndicator Flag
DiagnosisEvent (AREP, CREP, API, Diagnosis Data)	MAP_DIAG_EVT_IND	local Diagnosis Event	AREP CREP Alarm Item
CPM_Get_Status_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_GINIOCS_CNF-	local Get Input IOCS.cnf (-)	AREP
CPM_Get_Status_cnf (+) (CREP, Status, RecvCounter)	MAP_GINIOCS_CNF+	local Get Input IOCS.cnf (+)	AREP IOCS
CPM_Get_Data_cnf (-) (	MAP_GOUTD_CNF-	local Get	AREP

Primitive mapped from	Parameter mapped from	Primitive name	Associated parameters
CREP, ERRCLS, ERRCODE)		Output.cnf (-)	
CPM_Get_Data_cnf (+) (CREP, Data, New_Flag)	MAP_GOUTD_CNF+	local Get Output.cnf (+)	AREP IOPS Subslot Output Data New Flag IOCS
CPM_NewData_Ind (AREP, CREP, APDU_Status, data)	MAP_NEWDATA_IND	local New Output.ind	AREP CREP Slot Number Subslot Number InData Flag
if (Set_Variable (Arstate)) == FALSE	AREP=AREP	local Set AR State.cnf (-)	AREP
if (Set_Variable (Arstate)) == TRUE	AREP=AREP	local Set AR State.cnf (+)	AREP
DCP_Set.ind	CREP, ListOfData, ListOfControlCommands	local Set Command.ind	SA List Of Data = ListOfData List Of Control Commands = ListOfControlCommands
PPM_Set_Data_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_SIN_CNF	local Set Input.cnf (-)	AREP
PPM_Set_Data_cnf (+) (CREP)	MAP_SIN_CNF	local Set Input.cnf (+)	AREP
PPM_Set_Data_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_OIOCS_CNF	local Set Output IOCS.cnf (-)	AREP
PPM_Set_Data_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_OIOCS_CNF	local Set Output IOCS.cnf (+)	AREP
PPM_Set_Status_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_SPS_CNF	local Set Provider State.cnf (-)	AREP
PPM_Set_Status_cnf (+) (CREP)	MAP_SPS_CNF	local Set Provider State.cnf (+)	AREP
PPM_Set_Status_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_SPS_CNF	local Set Redundancy.cnf (-)	AREP
PPM_Set_Status_cnf (+) (CREP)	MAP_SPS_CNF	local Set Redundancy.cnf (+)	AREP
PPM_Set_Status_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_SPS_CNF	local Set State.cnf (-)	AREP
PPM_Set_Status_cnf (+) (CREP)	MAP_SPS_CNF	local Set State.cnf (+)	AREP
SyncStateChange_ind (Status)	MAP_SYSTI_IND	local Sync State Info.ind	Sync Data Sync Error Status
TickEvent (CycleCounter, CycleOffset)	MAP_SYNCH_IND	local SYNCH Event.ind	Slot Subslot Global Cycle Counter

Primitive mapped from	Parameter mapped from	Primitive name	Associated parameters
			Phase Status
CM_Ccontrol.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_AREADY_CNF-	Application Ready.cnf (-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Ccontrol.cnf (+)(AREP, ControlBlock)	MAP_AREADY_CNF+	Application Ready.cnf (+)	AREP Session Key Alarm Sequence Number
CM_Connect.ind (AREP, ARBlockReq, ListOfIOCRBlockReq, ListOfExpectedSubmoduleBlockReq, AlarmCRBlockReq, ParameterServerBlock, ListOfMulticastCRBlock, ARRPCBlock, IRInfosBlock, SRInfoBlock, ARVendorBlock, ARServerBlock, ARFSUBBlock)	MAP_CON_IND	Connect.ind	AREP AR Parameter Block List of IO CR Parameter Blocks List of Expected Submodule Blocks Alarm CR Parameter Block Parameter Server Block List of Multicast CR Block AR RPC Block IR Info Block SR Info Block AR Vendor Block AR Server Block AR FSU Block
CM_Dcontrol.ind (AREP, ControlBlock, ListOfSubmodules)	MAP_BOP_IND	Prm Begin.ind	AREP Session Key List of Submodules
CM_Dcontrol.ind (AREP, ControlBlock)	MAP_EOP_IND	Prm End.ind	AREP Session Key Alarm Sequence Number
CM_Read.ind (AREP, API, TargetARUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IND	Read.ind	AREP API Target ARUID Slot Number Subslot Number Index Seq Number Length
CM_Write.ind (AREP, SlotNumber, SubslotNumber, Index, PrmFlag, SeqNumber, Length, Data)	MAP_WRITE_IND	Write.ind	AREP API Slot Number Subslot Number Index Prm Flag Seq Number Length Data
ALPMI_Alarm_Notification.cnf (+)(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, Alarm_User_Data_Structure_Identifier, Alarm_User_Data, PNIO_Status)	MAP_AA_IND+	Alarm.cnf (+)	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier PNIO Status

Primitive mapped from	Parameter mapped from	Primitive name	Associated parameters
ALPMI_Alarm_Notification.cnf (-)(CREP, ERRCLS, ERRCODE)	MAP_AN_CNF-	Alarm.cnf (-)	AREP Status
ALPMR_Alarm_Notification.ind (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, User_Structure_Identifier, User_Data)	MAP_AN_IND	Alarm.ind	AREP API Alarm Priority Alarm Type Slot Number Subslot Number Alarm Specifier Module Ident Number Submodule Ident Number Alarm Item

5.5.2.3 State transition diagram

The FSPMDEV defines no state machine. Therefore no states are described.

5.5.2.4 State machine description

The FSPMDEV defines no state machine. Therefore no state machine description exists.

5.5.2.5 FSPMDEV state table

The FSPMDEV defines no state machine because all services are transferred to the underlying protocol machines.

5.5.2.6 Functions, Macros, Timers and Variables

Table 712 shows the functions, macros, timers and variables defined for FSPMDEV, which are used by service primitives issued by the FAL user.

Table 712 – Functions, Macros, Timers and Variables used by the AP-Context (FAL user) to FSPMDEV

Name	Function
MAP_ALARM_RSP-	if (PNIO Status != GOOD) then CREP=AREP.CREP API Alarm_Type=Alarm Type Slot_Number=Slot Number Subslot_Number=Subslot Number Alarm_Specifier=Alarm Specifier Sequence_Number PNIO_Status=PNIO Status
MAP_ALARM_RSP+	if (PNIO Status == GOOD) then CREP=AREP.CREP API Alarm_Type=Alarm Type Slot_Number=Slot Number Subslot_Number=Subslot Number Alarm_Specifier=Alarm Specifier Sequence_Number PNIO_Status=PNIO Status
MAP_AN_REQ	map service parameter 1:1 (=)

Name	Function
MAP_AREADY_REQ	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number) ModuleDiffBlock=Module Diff Block
MAP_BOP_RSP-	map service parameter 1:1 (=)
MAP_BOP_RSP+	AREP=AREP ControlBlockConnect.SessionKey=Session Key
MAP_CON_RSP-	map service parameter 1:1 (=)
MAP_CON_RSP+	AREP=AREP ARBlockRes=AR Response Block ListOfIOCRBlockRes=List of IO CR Response Blocks AlarmCRBlockRes=Alarm CR Response Block ModuleDiffBlock=Module Diff Block ARRPCBlock=AR RPC Block ARVendorBlock=AR Vendor Block
MAP_EOP_RSP-	map service parameter 1:1 (=)
MAP_EOP_RSP+	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number)
MAP_GINIOCS_REQ	CREP=CREP
MAP_OIOCS_REQ	CREP=CREP Data=IOCS
MAP_READ_RSP-	map service parameter 1:1 (=)
MAP_READ_RSP+	map service parameter 1:1 (=)
MAP_SIN_REQ	CREP=CREP Data={IOPS, Subslot Input Data}
MAP_SPS_REQ	CREP=AREP.CREP DataStatus={ProviderState=ProviderState Flag}
MAP_SRS_REQ	CREP=AREP.CREP DataStatus={Redundancy=Redundancy Flag}
MAP_SSS_REQ	CREP=AREP.CREP DataStatus={State=State Flag}
MAP_WRITE_RSP-	map service parameter 1:1 (=)
MAP_WRITE_RSP+	map service parameter 1:1 (=)

Table 713 shows the functions, macros, timers and variables defined for FSPMDEV, which are used by services primitives issued by the FSPMDEV.

Table 713 – Functions, Macros, Timers and Variables used by the FSPMDEV to AP-Context (FAL user)

Name	Function
MAP_AA_IND-	if (PNIO_Status != GOOD) then AREP=CREP.AREP API Alarm Type=Alarm_Type Slot Number=Slot_Number Subslot Number=Subslot_Number Alarm Specifier=Alarm_Specifier PNIO Status=PNIO_Status

Name	Function
MAP_AA_IND+	if (PNIO_Status == GOOD) then AREP=CREP.AREP API Alarm Type=Alarm_Type Slot Number=Slot_Number Subslot Number=Subslot_Number Alarm Specifier=Alarm_Specifier PNIO Status=PNIO_Status
MAP_AN_CNF-	AREP=CREP.AREP Status=(ERRCLS, ERRCODE)
MAP_AN_CNF+	AREP=CREP.AREP
MAP_AN_IND	AREP=CREP.AREP API Alarm Priority Alarm Type =Alarm_Type Slot Number =Slot_Number Subslot Number =Subslot_Number Alarm Specifier=Alarm_Specifier Module Ident Number=Module_Ident_Number Submodule Ident Number=Submodule_Ident_Number Alarm Item = {User_Structure_Identifier, User_Data}
MAP_AREADY_CNF-	map service parameter 1:1 (=)
MAP_AREADY_CNF+	AREP=AREP Session Key=ControlBlockPlug.SessionKey^ControlConnectPlug.SessionKey
MAP_BOP_IND	if (ControlBlockConnect.PrmEnd=TRUE) then AREP=AREP.CREP Session Key Alarm Sequence Number ControlCommand.PrmBegin=PrmBegin
MAP_CON_IND	AREP=AREP AR Parameter Block=ARBlockReq List of IO CR Parameter Blocks=ListOfIOCRBlockReq List of Expected Submodule Blocks=ListOfExpectedSubmoduleBlockReq Alarm CR Parameter Block=AlarmCRBlockReq Parameter Server Block=ParameterServerBlock List of Multicast CR Block=ListOfMulticastCRBlock AR RPC Block=ARRPCBlock IR Info Block=IRInfoBlock SR Info Block=SRInfoBlock AR Vendor Block=ARVendorBlock AR Server Block=ARServerBlock AR FSU Block=ARFSUBlock
MAP_DIAGEVT_IND	AREP =AREP CREP=CREP Alarm Item = Diagnosis Data
MAP_EOP_IND	if (List of Submodules=ListOfSubmodules) then AREP=AREP Session Key=ControlBlockConnect.SessionKey^ControlBlockPlug.SessionKey Alarm Sequence Number=ControlBlockPlug.AlarmSequenceNumber
MAP_GINIOCS_CNF-	AREP=AREP.CREP IOCS=Status RevCounter
MAP_GINIOCS_CNF+	AREP=AREP.CREP
MAP_GOUTD_CNF-	AREP=CREP.AREP other ignore
MAP_GOUTD_CNF+	AREP=CREP IOPS=Data Subslot Output Data=Data New Flag=New_Flag IOCS=Data

Name	Function
MAP_INDATA_IND	AREP=CREP.AREP CREP=CREP Slot Number =0 Subslot Number =0 InData Flag=TRUE
MAP_NEWDATA_IND	for each Slot/Subslot AREP=CREP.AREP CREP=CREP Slot Number=AREP.SlotNumber Subslot Number=AREP.SubSlotnumber InData Flag=data local New Output.ind(AREP, CREP, Slot Number, Subslot Number, InData Flag)if (data != NoData) AREP=CREP.AREP CREP=CREP
MAP_NEWSTATUS_IND	if (APDU_Status has been changed) AREP=CREP.AREP CREP=CREP DataValid Flag=APDU_Status.DataStatus.DataValid State Flag=APDU_Status.DataStatus.State Redundancy Flag=APDU_Status.DataStatus.Redundancy ProviderState Flag=APDU_Status.DataStatus.ProviderState ProblemIndicator Flag=APDU_Status.DataStatus.ProblemIndicator
MAP_NODATA_IND	AREP=CREP.AREP CREP=CREP Slot Number =0 Subslot Number =0 Watchdog Flag=TRUE
MAP_OIOCS_CNF	AREP=CREP.AREP other ignore
MAP_READ_IND	if ((Index < =0x7FFF) OR (0xAFF0<=Index<=0xAFFF)) then AREP=AREP API=API Target ARUUID = TargetARUUID Slot Number = SlotNumber Subslot Number =SubslotNumber Index =Index Seq Number =SeqNumber Length=Length else map to special FAL user Read service primitive
MAP_REL_IND	AREP=AREP Session Key=ReleaseBlock.SessionKey
MAP_SIN_CNF	AREP=CREP.AREP other ignore
MAP_SPS_CNF	AREP=CREP.AREP other ignore
MAP_SYNC_IND	Slot Subslot Global Cycle Counter=CycleCounter Phase Status
MAP_SYSTI_IND	Sync Data Sync Error Status=Status
MAP_WRITE_IND	if ((Index < =0x7FFF) OR (0xAFF0<=Index<=0xAFFF)) then AREP=AREP API=API Slot Number = SlotNumber Subslot Number =SubslotNumber Index =Index Multiple = Multiple Seq Number =SeqNumber Length=Length Data=Data else map to special FAL user Write service primitive

5.5.3 FAL Service Protocol Machine Controller

5.5.3.1 Overview

The FSPMCTL is responsible for the startup of the underlying protocol machines (LMPM, LLDP, DHCP/DCP, IP, UDP, RPC, CMDEV, CMCTL, DCP, PTCP, ALPM, PPM, CPM...).

5.5.3.2 Primitive definitions

Table 714 shows the service primitives including their associated parameters issued by the AP-Context (FAL user) and received by the FSPMCTL with the mapping to underlying services.

Table 714 – Primitives issued by AP-Context (FAL user) to FSPMCTL

Primitive name	Associated parameters	Primitive mapped to	Parameter mapped to
local AR Abort.req	AREP Session Key	CM_Release.req(AREP, ControlBlock)	MAP_REL_REQ
local Create LogBook Entry	info	CreateLogBookEntry (info)	info = info
local Get Input.req	AREP CREP Slot Number Subslot Number	CPM_Get_Data_req (CREP)	MAP_GIN_REQ
local Get Output IOCS.req	AREP CREP Slot Number Subslot Number	CPM_Get_Status_req (CREP)	MAP_GOUTIOCS_REQ
local Set AR State.req	AREP ARState	Set_Variable (Arstate)	AREP=AREP ARState=ARState
local Set Input IOCS.req	AREP CREP Slot Number Subslot Number IOCS	PPM_Set_Data_req (CREP, Data)	MAP_SINIOCS_REQ
local Set Output.req	AREP CREP Slot Number Subslot Number IOCS Subslot Output Data	PPM_Set_Data_req (CREP, Data)	MAP_SOUT_REQ
local Set Provider State.req	AREP Provider State Flag	PPM_Set_Status_req (CREP, D_Status)	MAP_SPS_REQ
local Set Redundancy.req	AREP Redundancy Flag	PPM_Set_Status_req (CREP, D_Status)	MAP_SRS_REQ
local Set State.req	AREP State Flag	PPM_Set_Status_req (CREP, D_Status)	MAP_SSS_REQ
Application Ready.rsp(-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Ccontrol.rsp (-)(AREP, ControlBlock)	MAP_AREADY_RES-
Application Ready.rsp(+)	AREP Session Key Alarm Sequence Number	CM_Ccontrol.rsp (+)(AREP, ControlBlock)	MAP_AREADY_RES+

Primitive name	Associated parameters	Primitive mapped to	Parameter mapped to
Connect.req	AREP AR Parameter Block List of IO CR Parameter Blocks List of Expected Submodule Blocks Alarm CR Parameter Block Parameter Server Block List of Multicast CR Blocks AR RPC Block IR Info Block SR Info Block AR Vendor Block AR Server Block AR FSU Block	CM_Connect.req (AREP, ARBlockReq, ListOfIOCRBlockReq, ListOfExpectedSubmoduleBlockReq, AlarmCRBlockReq, ParameterServerBlock, ListOfMulticastCRBlock, ARRPCBlock, IRInfosBlock, SRInfoBlock, ARVendorBlock, ARServerBlock, ARFSUBlock)	MAP_CON_REQ
Prm Begin.req	AREP Session Key Alarm Sequence Number List Of Submodules	CM_Dcontrol.req (AREP, ControlBlock, ListOfSubmodules)	MAP_PB_REQ
Prm End.req	AREP Session Key Alarm Sequence Number	CM_Dcontrol.req (AREP, ControlBlock)	MAP_EOP_REQ
Read.req	AREP API Target ARUUID Slot Number Subslot Number Index Seq Number Length	CM_Read.req (AREP, API, TargetARUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_REQ
Read.rsp (-)	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Read.rsp (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_RSP-
Read.rsp (+)	AREP Seq Number Length Data	CM_Read.rsp (+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RSP+
Write.req	AREP API Slot Number Subslot Number Index Seq Number Length Data	CM_Write.req (AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_REQ
Write.rsp (-)	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Write.rsp (-)(AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_RSP-

Primitive name	Associated parameters	Primitive mapped to	Parameter mapped to
Write.rsp (+)	AREP Seq Number	CM_Write.rsp (+) (AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_RSP+
Alarm.req	AREP API Alarm Priority Alarm Type Slot Number Subslot Number Alarm Specifier Module Ident Number Submodule Ident Number Alarm Item	ALPMI_Alarm_Notification.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, Alarm_User_Data_Structure_Identifier, Alarm_User_Data)	MAP_AN_REQ
Alarm.rsp (-)	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier PNIO Status	ALPMR_Alarm_Ack.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_ALARM_RSP-
Alarm.rsp (+)	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier PNIO Status	ALPMR_Alarm_Ack.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_ALARM_RSP+

Table 715 shows the service primitives including their associated parameters issued by the FSPMCTL and received by the AP-Context (FAL user).

Table 715 – Primitives issued by FSPMCTL to AP-Context (FAL user)

Primitive mapped to	Parameter mapped from	Primitive name	Associated parameters
CM_Abort.cnf (-)	AREP=AREP	local AR Abort.cnf (-)	AREP
CM_Abort.cnf (+)	AREP=AREP	local AR Abort.cnf (+)	AREP
CM_Abort.ind	AREP	local AR Abort.ind	AREP
CMCTL_state_ind (AREP, state)	if (state == DATA) then AREP=AREP else ignore	local AR In Data.ind	AREP
CPM_NewData_Ind (AREP, CREP, APDU_Status)	MAP_NEWSTATUS_IND	local Data State Changed.ind	AREP CREP DataValid Flag State Flag Redundancy Primary State Flag ProviderState Flag ProblemIndicator Flag
DiagnosisEvent (AREP, CREP, API, Diagnosis Data)	MAP_DIAGEVT_IND	local Diagnosis Event.ind	AREP CREP Alarm Item

Primitive mapped to	Parameter mapped from	Primitive name	Associated parameters
CPM_Get_Data_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_GIND_CNF-	local Get Input.cnf(-)	AREP
CPM_Get_Data_cnf (+) (CREP, Data, New_Flag)	MAP_GIND_CNF+	local Get Input.cnf(+)	AREP IOPS Subslot Input Data New Flag IOCS
CPM_Get_Status_cnf (-) (CREP, ERRCLS, ERRCODE)	MAP_GOUTIOCS_CNF-	local Get Output IOCS.cnf(-)	AREP
CPM_Get_Status_cnf (+) (CREP, Status, RecvCounter)	MAP_GOUTIOCS_CNF+	local Get Output IOCS.cnf(+)	AREP IOCS
CPM_NewData_Ind (AREP, CREP, APDU_Status)	MAP_NEWDATA_IND	local New Input.ind	AREP CREP Slot Number Subslot Number InData Flag
if (Set_Variable (Arstate)) == FALSE	AREP=AREP	local Set AR State.cnf (-)	AREP
if (Set_Variable (Arstate)) == TRUE	AREP=AREP	local Set AR State.cnf (+)	AREP
PPM_Set_Data_cnf(-) (CREP, ERRCLS, ERRCODE)	MAP_IIOCS_CNF	local Set Input IOCS.cnf(-)	AREP
PPM_Set_Data_cnf(+) (CREP)	MAP_IIOCS_CNF	local Set Input IOCS.cnf(+)	AREP
PPM_Set_Data_cnf(-) (CREP, ERRCLS, ERRCODE)	MAP_IIOCS_CNF	local Set Output.cnf(-)	AREP
PPM_Set_Data_cnf(+) (CREP)	MAP_IIOCS_CNF	local Set Output.cnf(+)	AREP
PPM_Set_Status_cnf(-) (CREP, ERRCLS, ERRCODE)	MAP_SPS_CNF	local Set Provider State.cnf(-)	AREP
PPM_Set_Status_cnf(+) (CREP)	MAP_SPS_CNF	local Set Provider State.cnf(+)	AREP
PPM_Set_Status_cnf(-) (CREP, ERRCLS, ERRCODE)	MAP_SRS_CNF	local Set Redundancy.cnf(-)	AREP
PPM_Set_Status_cnf(+) (CREP)	MAP_SRS_CNF	local Set Redundancy.cnf(+)	AREP
PPM_Set_Status_cnf(-) (CREP, ERRCLS, ERRCODE)	MAP_SPS_CNF	local Set State.cnf(-)	AREP
PPM_Set_Status_cnf(+) (CREP)	MAP_SPS_CNF	local Set State.cnf(+)	AREP
SyncStateChange_ind (Status)	MAP_SYSTI_IND	local Sync State Info.ind	Sync Data Sync Error Status
TickEvent (CycleCounter, CycleOffset)	MAP_SYNCH_IND	local SYNCH Event.ind	Slot Subslot Global Cycle

Primitive mapped to	Parameter mapped from	Primitive name	Associated parameters
			Counter Phase Status
CM_Ccontrol.ind (AREP, ControlBlock, ModuleDiffBlock)	MAP_AREADY_IND	Application Ready.ind	AREP Session Key Alarm Sequence Number Module Diff Block
CM_Connect.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_CON_CNF-	Connect.cnf(-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Connect.cnf (+) (AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock, ARRPCBlock, ARVendorBlock)	MAP_CON_CNF+	Connect.cnf(+)	AREP AR Response Block List of IO CR Response Blocks Alarm CR Response Block Module Diff Block AR RPC Block AR Vendor Block
CM_Release.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_REL_CNF-	local AR Abort.cnf (-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Release.cnf (+)(AREP, ControlBlock)	MAP_REL_CNF+	local AR Abort.cnf (+)	AREP Session Key
CM_Dcontrol.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	AREP	Prm Begin.cnf(-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Dcontrol.cnf (+) (AREP, ControlBlock)	AREP	Prm Begin.cnf(+)	AREP Session Key
CM_Dcontrol.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_EOP_CNF-	Prm End.cnf(-)	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Dcontrol.cnf (+) (AREP, ControlBlock)	MAP_EOP_CNF+	Prm End.cnf(+)	AREP Session Key Alarm Sequence Number
CM_Read.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_CNF-	Read.cnf(-)	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf (+)(AREP, Seq Number,	MAP_READ_CNF+	Read.cnf(+)	AREP Seq Number Length

Primitive mapped to	Parameter mapped from	Primitive name	Associated parameters
Length, Data)			Data
CM_Read.ind (AREP, API, TargetARUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IND	Read.ind	AREP API Target ARUUID Slot Number Subslot Number Index Seq Number Length
CM_Write.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_CNF-	Write.cnf(-)	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Write.cnf (+)(AREP, Seq Number)	MAP_WRITE_CNF+	Write.cnf(+)	AREP Seq Number
CM_Write.ind (AREP, SlotNumber, SubslotNumber, Index, PrmFlag, SeqNumber, Length, Data)	MAP_WRITE_IND	Write.ind	AREP API Slot Number Subslot Number Index Prm Flag Seq Number Length Data
ALPMI_Alarm_Notification.cnf (+)(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, Alarm_User_Data_Structure_Identifier, Alarm_User_Data, PNIO_Status)	MAP_AA_IND+	Alarm.cnf (+)	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier PNIO Status
ALPMI_Alarm_Notification.cnf (-)(CREP, ERRCLS, ERRCODE)	MAP_AN_CNF	Alarm.cnf(-)	AREP Status
ALPMR_Alarm_Notification.ind (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, User_Structure_Identifier, User_Data)	MAP_AN_IND	Alarm.ind	AREP API Alarm Priority Alarm Type Slot Number Subslot Number Alarm Specifier Module Ident Number Submodule Ident Number Alarm Item

5.5.3.3 State transition diagram

The FSPMCTL defines no state machine. Therefore no states are described.

5.5.3.4 State machine description

The FSPMCTL defines no state machine. Therefore no state machine description exists.

5.5.3.5 FSPMCTL state table

The FSPMCTL defines no state machine because all services are transferred to the underlying protocol machines.

5.5.3.6 Functions, Macros, Timers and Variables

Table 716 shows the functions, macros, timers and variables defined for FSPMCTL, which are used by services primitives issued by the FAL user.

Table 716 – Functions, Macros, Timers and Variables used by AP-Context (FAL user) to FSPMCTL

Name	Function
MAP_ALARM_RSP-	if (PNIO Status != GOOD) then CREP=AREP.CREP API Alarm_Type=Alarm Type Slot_Number=Slot Number Subslot_Number=Subslot Number Alarm_Specifier=Alarm Specifier Sequence_Number PNIO_Status=PNIO Status
MAP_ALARM_RSP+	if (PNIO Status == GOOD) CREP=AREP.CREP API Alarm_Type=Alarm Type Slot_Number=Slot Number Subslot_Number=Subslot Number Alarm_Specifier=Alarm Specifier Sequence_Number PNIO_Status=PNIO Status
MAP_AN_REQ	CREP=AREP.CREP API Alarm Priority Alarm_Type=Alarm Type Slot_Number=Slot Number Subslot_Number=Subslot Number Alarm_Specifier=Alarm Specifier Module_Ident_Number=Module Ident Number Submodule_Ident_Number=Submodule Ident Number Alarm_User_Data_Structure_Identifier=User Structure Identifier Alarm_User_Data=User Structure Identifier.User Data
MAP_AREADY_RES-	AREP=AREP Error Decode Error Code 1 Error Code 2
MAP_AREADY_RES+	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number)
MAP_CON_REQ	AREP=AREP ARBlockReq=AR Parameter Block ListOfIOCRBlockReq=List of IO CR Parameter Blocks ListOfExpectedSubmoduleBlockReq=List of Expected Submodule Blocks AlarmCRBlockReq=Alarm CR Parameter Block ParameterServerBlock=Parameter Server Block ListOfMulticastCRBlock=List of Multicast CR Block ARRPCBlock=AR RPC Block IRInfoBlock=IR Info Block SRInfoBlock=SR Info Block ARVendorBlock=AR Vendor Block ARServerBlock=AR Server Block AR FSU Block=AR FSU Block

Name	Function
MAP_EOP_REQ	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number) ControlBlockConnect.PrmEnd=TRUE
MAP_GIN_REQ	CREP=CREP
MAP_GOUTIOCS_REQ	CREP=CREP
MAP_PB_REQ	AREP=AREP ControlBlockConnect.SessionKey=Session Key ControlCommand.PrmBegin=PrmBegin ListOfSubmodules=List Of Submodules
MAP_READ_REQ	map service parameter 1:1 (=)
MAP_READ_RSP-	map service parameter 1:1 (=)
MAP_READ_RSP+	map service parameter 1:1 (=)
MAP_REL_REQ	AREP=AREP ReleaseBlock.SessionKey=SessionKey
MAP_SINIOCS_REQ	CREP=CREP Data=IOCS
MAP_SOUT_REQ	CREP=CREP Data={IOPS,Subslot Output Data}
MAP_SPS_REQ	CREP=AREP.CREP DataStatus={ProviderState=ProviderState Flag}
MAP_SRS_REQ	CREP=AREP.CREP DataStatus={Redundancy=Redundancy Flag}
MAP_SSS_REQ	CREP=AREP.CREP DataStatus={State=State Flag}
MAP_WRITE_REQ	AREP=AREP API SlotNumber=Slot Number SubslotNumber=Subslot Number Index=Index Multiple=Multiple SeqNumber=Seq Number Length=Length Data=Data
MAP_WRITE_RSP-	map service parameter 1:1 (=)
MAP_WRITE_RSP+	map service parameter 1:1 (=)

Table 717 shows the functions, macros, timers and variables defined for FSPMCTL, which are used by services primitives issued by the FSPMCTL.

Table 717 – Functions, Macros, Timers and Variables used by FSPMCTL to AP-Context (FAL user)

Name	Function
MAP_AA_CNF	AREP=CREP.AREP other ignore
MAP_AA_IND-	if (PNIO_Status != GOOD) then AREP=CREP.AREP API Alarm Type=Alarm_Type Slot Number=Slot_Number Subslot Number=Subslot_Number Alarm Specifier=Alarm_Specifier PNIO Status=PNIO_Status

Name	Function
MAP_AA_IND+	if (PNIO_Status == GOOD) then AREP=CREP.AREP API Alarm Type=Alarm_Type Slot Number=Slot_Number Subslot Number=Subslot_Number Alarm Specifier=Alarm_Specifier PNIO Status=PNIO_Status
MAP_AN_CNF	AREP=CREP.AREP Status
MAP_AN_IND	AREP=CREP.AREP API Alarm Priority Alarm Type =Alarm_Type Slot Number =Slot_Number Subslot Number =Subslot_Number Alarm Specifier=Alarm_Specifier Module Ident Number=Module_Ident_Number Submodule Ident Number=Submodule_Ident_Number Alarm Item = {User_Structure_Identifier, User_Data}
MAP_AREADY_IND	AREP=AREP Session Key=ControlBlockPlug.SessionKey^ControlConnectPlug.SessionKey Module Diff Block=ModuleDiffBlock
MAP_CON_CNF-	AREP =AREP Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2 Add Data 1=AddData1 Add Data 2=AddData2
MAP_CON_CNF+	AREP =AREP AR Response Block=ARBlockRes List of IO CR Response Blocks=ListOfIOCRBlockRes Alarm CR Response Block=AlarmCRBlockRes Module Diff Block=ModuleDiffBlock AR RPC Block=ARRPCBlock AR Vendor Block=ARVendorBlock
MAP_DIAGEVT_IND	AREP =AREP CREP=CREP Alarm Item = Diagnosis Data
MAP_EOP_CNF-	AREP =AREP Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2 Add Data 1=AddData1 Add Data 2=AddData2
MAP_EOP_CNF+	AREP=AREP Session Key= ControlBlockConnect.SessionKey^ControlBlockPlug.SessionKey Alarm Sequence Number=ControlBlockPlug.AlarmSequenceNumber
MAP_GIND_CNF-	AREP=CREP.AREP other ignore
MAP_GIND_CNF+	AREP=CREP IOPS=Data Subslot Input Data=Data New Flag=New_Flag IOCS=Data
MAP_GOUTIOCS_CNF-	AREP=CREP.AREP ERRCLS ERRCODE
MAP_GOUTIOCS_CNF+	AREP=CREP.AREP IOCS=Status RevCounter
MAP_IIOCS_CNF	AREP=CREP.AREP other ignore

Name	Function
MAP_INDATA_IND	AREP=CREP.AREP CREP=CREP Slot Number =0 Subslot Number =0 InData Flag=TRUE
MAP_NEWDATA_IND	if (Data has been changed) then AREP=AREP CREP=CREP Slot Number Subslot Number
MAP_NEWSTATUS_IND	if (APDU_Status has been changed) AREP=CREP.AREP CREP=CREP DataValid Flag=APDU_Status.DataStatus.DataValid State Flag=APDU_Status.DataStatus.State Redundancy Flag=APDU_Status.DataStatus.Redundancy ProviderState Flag=APDU_Status.DataStatus.ProviderState ProblemIndicator Flag=APDU_Status.DataStatus.ProblemIndicator
MAP_NODATA_IND	AREP=CREP.AREP CREP=CREP Slot Number =0 Subslot Number =0 Watchdog Flag=TRUE
MAP_READ_CNF-	if (ServiceReqWasRead(AREP, Seq Number) == TRUE) then AREP = AREP Seq Number = Seq Number Error Decode = ErrorDecode Error Code 1 = ErrorCode1 Error Code 2 = ErrorCode2 Add Data 1 = AddData1 Add Data 2 =AddData2
MAP_READ_CNF+	if (ServiceReqWasRead(AREP, Seq Number) == TRUE) then AREP = AREP Seq Number = Seq Number Length = Length Data = Data
MAP_READ_IND	if ((Index < =0x7FFF) OR (0xAFF0<=Index<=0xAFFF)) then AREP=AREP API=API Target ARUUID = TargetARUUID Slot Number = SlotNumber Subslot Number =SubslotNumber Index =Index Seq Number =SeqNumber Length=Length else map to special FAL user Read service primitive
MAP_REL_CNF-	AREP =AREP Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2
MAP_REL_CNF+	AREP=AREP Session Key=ReleaseBlock.SessionKey
MAP_SPS_CNF	AREP=CREP.AREP other ignore
MAP_SRS_CNF	AREP=CREP.AREP other ignore
MAP_SYNCH_IND	Slot Subslot Global Cycle Counter=CycleCounter Phase Status
MAP_SYSTI_IND	Sync Data Sync Error Status=Status

Name	Function
MAP_WRITE_CNF-	if (ServiceReqWasWrite(AREP, Seq Number) == TRUE) then AREP=AREP Multiple Seq Number Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2 Add Data 1=AddData1 Add Data 2=AddData2
MAP_WRITE_CNF+	if (ServiceReqWasWrite(AREP, Seq Number) == TRUE) then AREP=AREP Multiple Seq Number =SeqNumber
MAP_WRITE_IND	if ((Index < =0x7FFF) OR (0xAFF0<=Index<=0xAFFF)) then AREP=AREP API=API Slot Number = SlotNumber Subslot Number =SubslotNumber Index =Index Multiple = Multiple Seq Number =SeqNumber Length=Length Data=Data} else map to special FAL user Write service primitive

5.6 Application Relationship Protocol Machines

5.6.1 Alarm Protocol Machine Initiator

5.6.1.1 Primitive definitions

5.6.1.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Alarm Protocol Machine Initiator (ALPMI) are described in the service definition and shown in Table 718.

Table 718 – Remote primitives issued or received by ALPMI

Primitive	Source	Destination	Associated parameters	Functions
APMR_A_Data.ind	APMR	ALPMI	CREP, Data	—
APMS_A_Data.cnf (-)	APMS	ALPMI	CREP, ERRCLS, ERRCODE	—
APMS_A_Data.cnf (+)	APMS	ALPMI	CREP	—
ALPMI_Alarm_Notification.req	FSPMDEV FSPMCTL	ALPMI	CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number , Alarm_User_Data_Structure_Identifier, Alarm_User_Data	The service Alarm_Notification conveys alarm data.
APMS_A_Data.req	ALPMI	APMS	CREP, Data	—

Primitive	Source	Destination	Associated parameters	Functions
ALPMI_Alarm_Notification.cnf (-)	ALPMI	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	This service primitive indicates that the Alarm_Notification service failed.
ALPMI_Alarm_Notification.cnf (+)	ALPMI	FSPMDEV FSPMCTL	CREP	This service primitive indicates that the Alarm_Notification service succeeded.

5.6.1.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by ALPMI are described in the service definition and shown in Table 719.

Table 719 – Local primitives issued or received by ALPMI

Primitive	Source	Destination	Associated parameters	Functions
ALPMI_Activate_req	CMSU CTLSU	ALPMI	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	The service Activate initializes the ALPMI and requests the initialization of the APMS and APMR protocol machines.
ALPMI_Close_req	CMSU CTLSU	ALPMI	CREP	The service Close deinitializes the ALPMI and closes the APMR and APMS protocol machines.
ALPMI_Activate.cnf (-)	ALPMI	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
ALPMI_Activate.cnf (+)	ALPMI	CMSU CTLSU	CREP	—
ALPMI_Close.cnf (+)	ALPMI	CMSU CTLSU	CREP	—
ALPMI_Error_ind	ALPMI	CMSU CTLSU	CREP, ERRCLS, ERRCODE	This service primitive indicates a failure during transmission of alarm data.

5.6.1.2 State transition diagram

The state transition diagram of the ALPMI is shown in Figure 96.

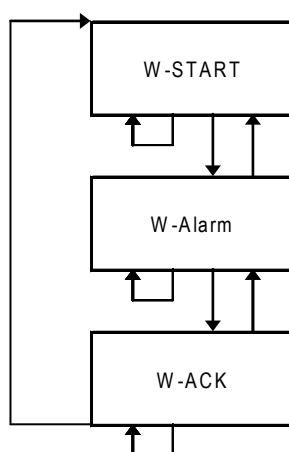


Figure 96 – State transition diagram of ALPMI

States of the ALPMI

W-START	The W-START state indicates that the initialization is needed. The Activate service sets the machine in the W-Alarm state.
W-Alarm	Indicates a successful initialization and the state machine is waiting for Alarm Notification requests
W-ACK	Wait for an Alarm Acknowledge and generate the Alarm Notification confirmation.

5.6.1.3 State machine description

This state machine handles the Alarm Notification. Each priority (High and Low) is handled independent. An application issues an Alarm.req which is conveyed by the ALPMI and the APMS to the APMR and the ALPMR. The ALPMR indicates the Alarm.ind to the remote application.

The remote application issues an Alarm.rsp to the ALPMR and the APMS which is conveyed to the APMR and the ALPMI. The ALPMI issues an Alarm.cnf to the application.

5.6.1.4 ALPMI state table

Table 720 contains the complete description of the ALPMI state machine.

Table 720 – ALPMI state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	ALPMI_Activate_req () => AlarmInstance[SrcSAP, DstSAP, Prio] := DA, RTATimeoutFactor, RTARetries, ... // NOTE Information used for ALPMI, ALPMR, APMS and APMR Use CREP to identify the AlarmInstance ALPMI_Activate_cnf (+)	W-Alarm
2	W-START	ALPMI_Close_req () => ALPMI_Close_cnf ()	W-START

#	Current State	Event /Condition =>Action	Next State
3	W-START	APMS_A_Data.cnf (+) => ignore	W-START
4	W-START	APMS_A_Data.cnf (-) => ERRCLS := ALPMI ERRCODE := Invalid ALPMI_Error_ind ()	W-START
5	W-START	ALPMI_Alarm_Notification.req () => ALPMI_Alarm_Notification.cnf (-)	W-START
6	W-START	APMR_A_Data.ind () => ignore	W-START
7	W-Alarm	ALPMI_Alarm_Notification.req () => Select the addressed / associated APMS and combine data for the alarm notification request APMS_A_Data.req ()	W-ACK
8	W-Alarm	ALPMI_Activate_req () => ERRCLS:= ALPMI ERRCODE:= WRONG-STATE ALPMI_Activate_cnf (-)	W-Alarm
9	W-Alarm	ALPMI_Close_req () => ALPMI_Close_cnf ()	W-START
10	W-Alarm	APMS_A_Data.cnf (+) => ignore	W-Alarm
11	W-Alarm	APMS_A_Data.cnf (-) => ERRCLS := ALPMI ERRCODE := Invalid ALPMI_Error_ind ()	W-Alarm
12	W-Alarm	APMR_A_Data.ind () => ignore	W-Alarm
13	W-ACK	APMR_A_Data.ind () /Data == Alarm-Ack-PDU => ALPMI_Alarm_Notification.cnf (+)	W-Alarm
14	W-ACK	ALPMI_Activate_req () => ERRCLS:= ALPMI ERRCODE:= WRONG-STATE ALPMI_Activate_cnf (-)	W-ACK
15	W-ACK	ALPMI_Alarm_Notification.req () => ALPMI_Alarm_Notification.cnf (-)	W-ACK
16	W-ACK	ALPMI_Close_req () => ALPMI_Close_cnf ()	W-START
17	W-ACK	APMS_A_Data.cnf (+) => ignore	W-ACK
18	W-ACK	APMS_A_Data.cnf (-) => ERRCLS := ALPMI ERRCODE := Invalid ALPMI_Error_ind ()	W-ACK

#	Current State	Event /Condition =>Action	Next State
19	W-ACK	APMR_A_Data.ind () /Data != Alarm-Ack-PDU => ERRCLS:= ALPMI ERRCODE:= WRONG-ACK-PDU ALPMI_Error_ind ()	W-Alarm

5.6.1.5 Functions, Macros, Timers and Variables

Table 721 contains the functions, macros, timers and variables used by the ALPMI and their arguments and their descriptions.

Table 721 – Functions, Macros, Timers and Variables used by ALPMI

Name	Type	Meaning
AlarmInstance	Variable	This AR global variable contains the required information for this protocol machine instance.
ERRCLS	Variable	This local variable contains the entity signaling the error.
ERRCODE	Variable	This local variable contains the error reason in the context of the ERRCLS.

5.6.2 Alarm Protocol Machine Responder

5.6.2.1 Primitive definitions

5.6.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Alarm Protocol Machine Responder (ALPMR) are described in the service definition and shown in Table 722.

Table 722 – Remote primitives issued or received by ALPMR

Primitive	Source	Destination	Associated parameters	Functions
APMR_A_Data.ind	APMR	ALPMR	CREP.APMR, Data	—
APMS_A_Data.cnf (-)	APMS	ALPMR	CREP.APMS, ERRCLS, ERRCODE	—
APMS_A_Data.cnf (+)	APMS	ALPMR	CREP	—
ALPMR_Alarm_Ack.req	FSPMCTL FSPMDEV	ALPMR	CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status	The service Alarm_Ack conveys the alarm acknowledgement.
ALPMR_Alarm_Ack.cnf (-)	ALPMR	FSPMCTL FSPMDEV	CREP, ERRCLS, ERRCODE	This service primitive indicates that the Alarm_Ack service failed.
ALPMR_Alarm_Ack.cnf (+)	ALPMR	FSPMCTL FSPMDEV	CREP	This service primitive indicates that the Alarm_Ack service succeeded.

Primitive	Source	Destination	Associated parameters	Functions
ALPMR_Alarm_Notification.ind	ALPMR	FSPMCTL FSPMDEV	CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, Module_Ident_Number, Submodule_Ident_Number, User_Structure_Identifier, User_Data	The service Alarm_Notification indicates alarm data.

5.6.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by ALPMR are described in the service definition and shown in Table 723.

Table 723 – Local primitives issued or received by ALPMR

Primitive	Source	Destination	Associated parameters	Functions
ALPMR_Activate_req	CMSU CTLSU	ALPMR	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	The service Activate initializes the ALPMR and requests the initialization of the APMS and APMR protocol machines.
ALPMR_Close_req	CMSU CTLSU	ALPMR	CREP	The service Close deinitializes the ALPMI and closes the APMR and APMS protocol machines.
ALPMR_Activate_cnf (-)	ALPMR	CMSU CTLSU	CREP, ERRCLS, ERRCODE	This service primitive indicates that the Activate service failed.
ALPMR_Activate_cnf (+)	ALPMR	CMSU CTLSU	CREP	This service primitive indicates that the Activate service succeeded.
ALPMR_Close_cnf (+)	ALPMR	CMSU CTLSU	CREP	This service primitive indicates that the Close service succeeded.
ALPMR_Error_ind	ALPMR	CMSU CTLSU	CREP, ERRCLS, ERRCODE	This service primitive indicates a failure during transmission of alarm data.

5.6.2.2 State transition diagram

The state transition diagram of the ALPMR is shown in Figure 97.

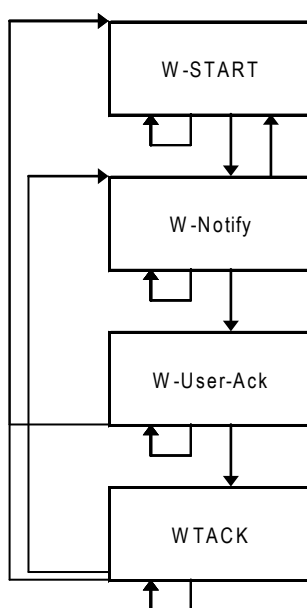


Figure 97 – State transition diagram of ALPMR

States of the ALPMR

W-START	The W-START state indicates that the initialization is needed. The Activate service sets the machine in the W-Notify state.
W-Notify	After successful initialization the state machine is waiting for Alarm Notification PDUs in the state W-Notify and enters afterwards the W-User-Ack state waiting for an Alarm Ack from the application.
W-User-Ack	Waiting for an Alarm Ack from the application.
WTACK	Wait for the confirmation of the transport acknowledge.

5.6.2.3 State machine description

This state machine handles the responder for an Alarm Notification. Each priority (High and Low) is handled independent.

A remote application issues an Alarm.req which is conveyed by the ALPMI and the APMS to the APMR and the ALPMR. The ALPMR indicates the Alarm.ind to the application.

The application issues an Alarm.rsp to the ALPMR and the APMS which is conveyed to the APMR and the ALPMI. The ALPMI issues an Alarm.cnf to the remote application.

5.6.2.4 ALPMR state table

Table 724 contains the complete description of the ALPMR state machine.

Table 724 – ALPMR state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	ALPMR_Activate_req () => AlarmInstance[SrcSAP, DstSAP, Prio] := DA, RTATimeoutFactor, RTARetries, ... // NOTE Information used for ALPMI, ALPMR, APMS and APMR Use CREP to identify the AlarmInstance ALPMR_Activate_cnf (+)	W-Notify
2	W-START	ALPMR_Close_req () => ALPMR_Close_cnf ()	W-START
3	W-START	ALPMR_Alarm_Ack.req () => ERRCLS:= ALPMR ERRCODE:= WRONG-STATE ALPMR_Alarm_Ack.cnf (-)	W-START
4	W-START	APMR_A_Data.ind () => ignore	W-START
5	W-START	APMS_A_Data.cnf () => ignore	W-START
6	W-Notify	ALPMR_Activate_req () => ERRCLS:= ALPMR ERRCODE:= WRONG-STATE ALPMR_Activate_cnf (-)	W-Notify
7	W-Notify	ALPMR_Alarm_Ack.req () => ERRCLS:= ALPMR ERRCODE:= WRONG-STATE ALPMR_Alarm_Ack.cnf (-)	W-Notify
8	W-Notify	ALPMR_Close_req () => ALPMR_Close_cnf () ALPMR_Error_ind ()	W-START
9	W-Notify	APMR_A_Data.ind () /Data != Alarm-Notification-PDU => ERRCLS:= ALPMR ERRCODE:= WRONG-NOTIFICATION-PDU ALPMR_Error_ind ()	W-Notify
10	W-Notify	APMR_A_Data.ind () /Data = Alarm-Notification-PDU => Combine data for the alarm notification indication ALPMR_Alarm_Notification.ind ()	W-User-Ack
11	W-Notify	APMS_A_Data.cnf (+) => ignore	W-Notify
12	W-Notify	APMS_A_Data.cnf (-) => ERRCLS:= ALPMR ERRCODE:= Invalid ALPMR_Error_ind ()	W-Notify
13	W-User-Ack	ALPMR_Activate_req () => ERRCLS:= ALPMR ERRCODE:= WRONG-STATE ALPMR_Activate_cnf (-)	W-User-Ack

#	Current State	Event /Condition =>Action	Next State
14	W-User-Ack	ALPMR_Alarm_Ack.req () => Select the addressed / associated APMS and combine data for the alarm ack request APMS_A_Data.req ()	WTACK
15	W-User-Ack	ALPMR_Close_req () => ALPMR_Close_cnf () ALPMR_Error_ind ()	W-START
16	W-User-Ack	APMR_A_Data.ind () => ERRCLS:= RTA_ERR_CLS_PROTOCOL ERRCODE:= AR protocol violation Combine data for the RTA error request ALPMR_Error_ind ()	W-User-Ack
17	W-User-Ack	APMS_A_Data.cnf (+) => ignore	W-User-Ack
18	W-User-Ack	APMS_A_Data.cnf (-) => ERRCLS:= ALPMR ERRCODE:= Invalid ALPMR_Error_ind ()	W-User-Ack
19	WTACK	APMS_A_Data.cnf (+) => ALPMR_Alarm_Ack.cnf (+)	W-Notify
20	WTACK	ALPMR_Activate_req () => ERRCLS:= ALPMR ERRCODE:= WRONG-STATE ALPMR_Activate_cnf (-)	WTACK
21	WTACK	ALPMR_Alarm_Ack.req () => ERRCLS:= ALPMR ERRCODE:= WRONG-STATE ALPMR_Alarm_Ack.cnf (-)	WTACK
22	WTACK	ALPMR_Close_req () => ALPMR_Close_cnf () ALPMR_Error_ind ()	W-START
23	WTACK	APMR_A_Data.ind () => ERRCLS:= RTA_ERR_CLS_PROTOCOL ERRCODE:= AR protocol violation ALPMR_Error_ind ()	WTACK
24	WTACK	APMS_A_Data.cnf (-) => ERRCLS:= ALPMR ERRCODE:= Invalid ALPMR_Error_ind ()	WTACK

5.6.2.5 Functions, Macros, Timers and Variables

Table 725 contains the functions, macros, timers and variables used by the ALPMR and their arguments and their descriptions.

Table 725 – Functions, Macros, Timers and Variables used by ALPMR

Name	Type	Meaning
AlarmInstance	Variable	This AR global variable contains the required information for this protocol machine instance.
ERRCLS	Variable	This local variable contains the entity signaling the error.
ERRCODE	Variable	This local variable contains the error reason in the context of the ERRCLS.

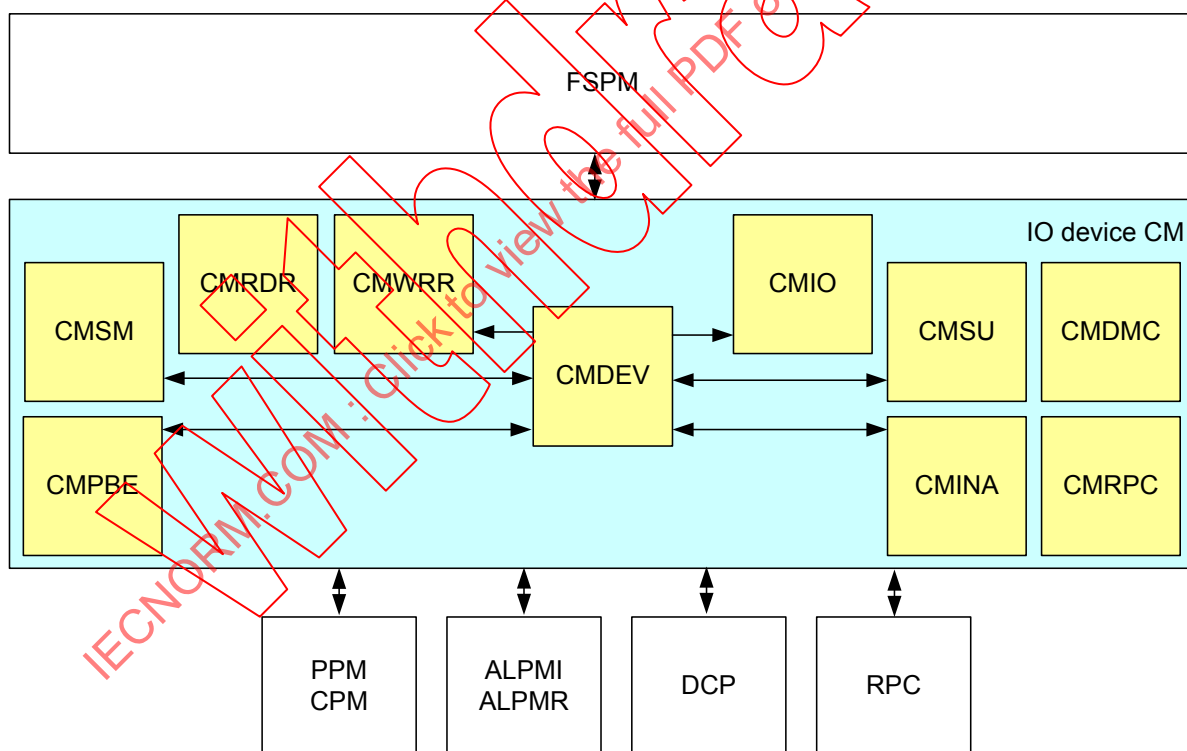
5.6.3 Device

5.6.3.1 General

5.6.3.1.1 General

The CMDEV arranges the connection establishment of an IO device. Figure 98 shows the integration of the IO device CM.

The following definition shall be used for ARBlockReq.ARProperties.StartupMode:=Advanced. For ARBlockReq.ARProperties.StartupMode:=Legacy the definitions of the version V2.2 and the Annex B applies.

**Figure 98 – Scheme of the IO device CM**

CMDEV

This state machine handles the context management of the IO device.

CMWRR

This state machine handles the responder functionality of the write service.

CMRDR

This state machine handles the responder functionality of the read service.

CMSM

This state machine handles the connection monitoring during the startup.

CMPBE

This optional state machine handles the PrmBegin, PrmEnd and ApplRdy sequence for system redundancy and configure in run.

CMSU

This state machine handles the startup of the different state machines during startup and shut down.

CMIO

This state machine handles the IO data services.

CMDMC

This state machine handles the startup of the multicast communication relations.

CMRPC

This state machine handles the translation of RPC services.

CMINA

This state machine handles IP and Name assignment.

5.6.3.1.2 State transition diagram

The state transition diagram of the IO device CM is shown in Figure 99.

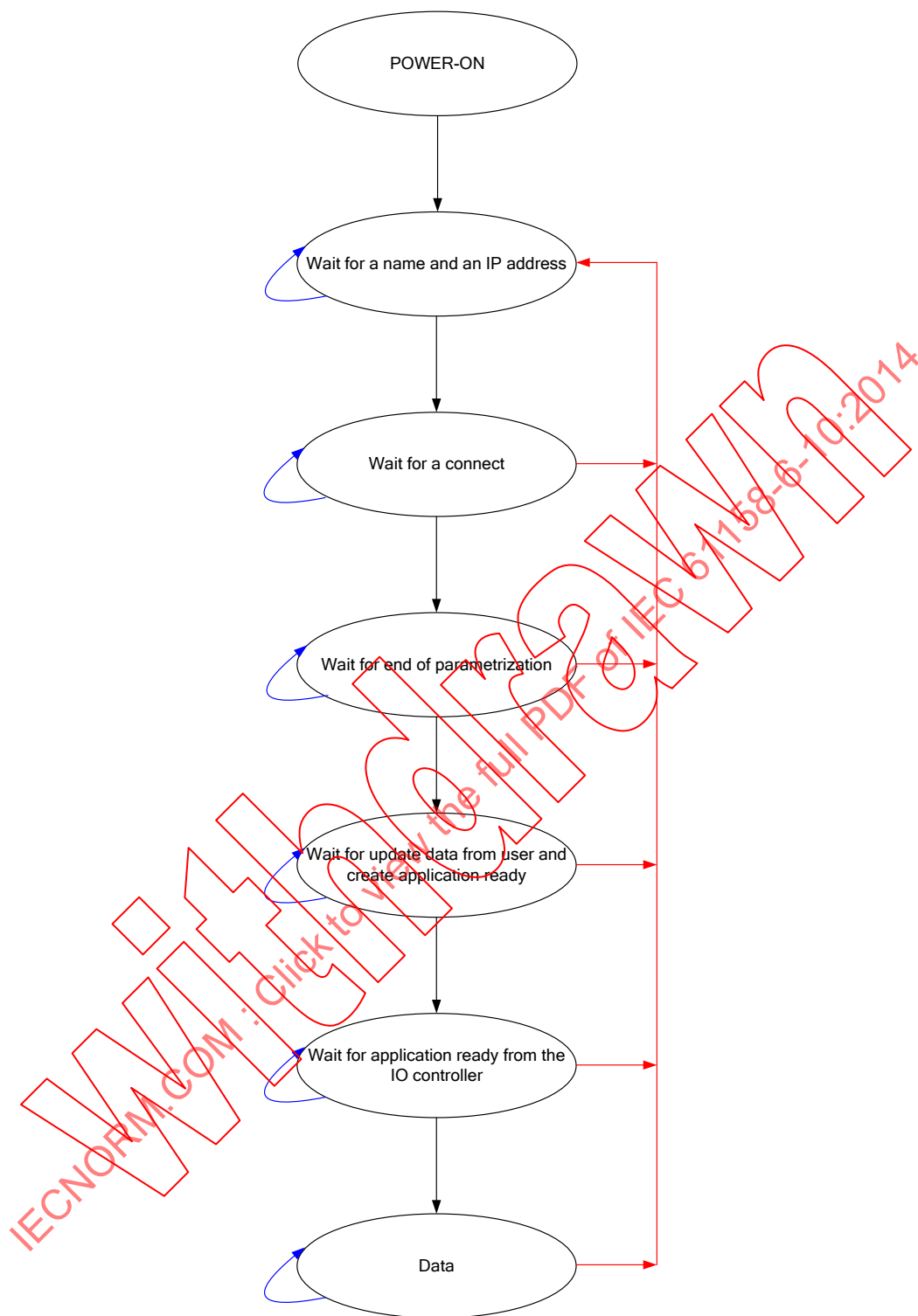


Figure 99 – State transition diagram of the IO device CM

States of the IO device CM

POWER-ON

Startup the IO device and initialize all necessary resources

Wait for a name and an IP address

A name and an IP address is needed for the communication to the IO controller

Wait for connect

The IO device is the responder for the application relation to the IO controller. The connect establishes an application relation between the IO controller and the IO device. Part of this AR are the required communication resources and the requested submodules.

Wait for end of parameterization	The IO controller parameterizes all submodules of the AR and informs the application of the IO device when the last startup record is conveyed. The application uses this signal to evaluate the parameters and executes them.
Wait for update from user and create application ready	The submodules generate interpretable data after the parameterization. This data shall be updated in the PPM buffer before the application ready is signaled to the IO controller.
Wait for application ready from IO controller	The IO controller has received the application ready from the IO device executes it locally. After that, the confirmation of the application ready shall be given to the IO device.
Data	Cyclic exchange data of the submodules between the IO controller and IO devices starts, with the first NewData_ind after the reception of the application ready confirmation from the IO controller. See Annex A for the special case isochronous application

5.6.3.2 Context Management Device

5.6.3.2.1 CMDEV state machine

5.6.3.2.1.1 Primitive definitions

5.6.3.2.1.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Device (CMDEV) are described in the service definition and shown in Table 726.

Table 726 – Remote primitives issued or received by CMDEV

Primitive	Source	Destination	Associated parameters	Functions
CM_Ccontrol.req	FSPMDEV	CMDEV	AREP, ControlBlock, ModuleDiffBlock	The service Ccontrol conveys the application ready flag.
CM_Connect.rsp (-)	FSPMDEV	CMDEV	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	This service primitive is a positive response to establish an application relationship.
CM_Connect.rsp (+)	FSPMDEV	CMDEV	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock	This service primitive is a negative response and the requested application relationship is not established.
CM_Dcontrol.rsp (-)	FSPMDEV	CMDEV	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	This service primitive is a response to end of parameter signal.
CM_Dcontrol.rsp (+)	FSPMDEV	CMDEV	AREP, ControlBlock	This service primitive is a response to end of parameter signal.
RM_Ccontrol.cnf (-)	CMRPC	CMDEV	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Ccontrol.cnf (+)	CMRPC	CMDEV	AREP, ControlBlock	—

Primitive	Source	Destination	Associated parameters	Functions
RM_Connect.ind	CMRPC	CMDEV	AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq, ListOfSubFrameBlockReq	—
RM_Dcontrol.ind	CMRPC	CMDEV	AREP, ControlBlock	—
RM_Release.ind	CMRPC	CMDEV	AREP, ControlBlock	—
CM_Ccontrol.cnf (-)	CMDEV	FSPMDEV	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	This service primitive confirms the Ccontrol service.
CM_Ccontrol.cnf (+)	CMDEV	FSPMDEV	AREP, ControlBlock	This service primitive confirms the Ccontrol service.
CM_Connect.ind	CMDEV	FSPMDEV	AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq, ListOfSubFrameBlockReq	This service primitive indicates a request to establish an application relationship.
CM_Dcontrol.ind	CMDEV	FSPMDEV	AREP, ControlBlock	This service primitive indicates the end of parameter.
CM_Release.ind	CMDEV	FSPMDEV	AREP, ReleaseBlock	This service primitive indicates that the application relationship should be released.
RM_Ccontrol.req	CMDEV	CMRPC	AREP, ControlBlock, ModuleDiffBlock	—
RM_Connect.rsp (-)	CMDEV	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2	—
RM_Connect.rsp (+)	CMDEV	CMRPC	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock	—
RM_Dcontrol.rsp (+)	CMDEV	CMRPC	AREP, ControlBlock	—
RM_Release.rsp (+)	CMDEV	CMRPC	AREP, ControlBlock	—

5.6.3.2.1.1.2 Primitives exchanged between local machines

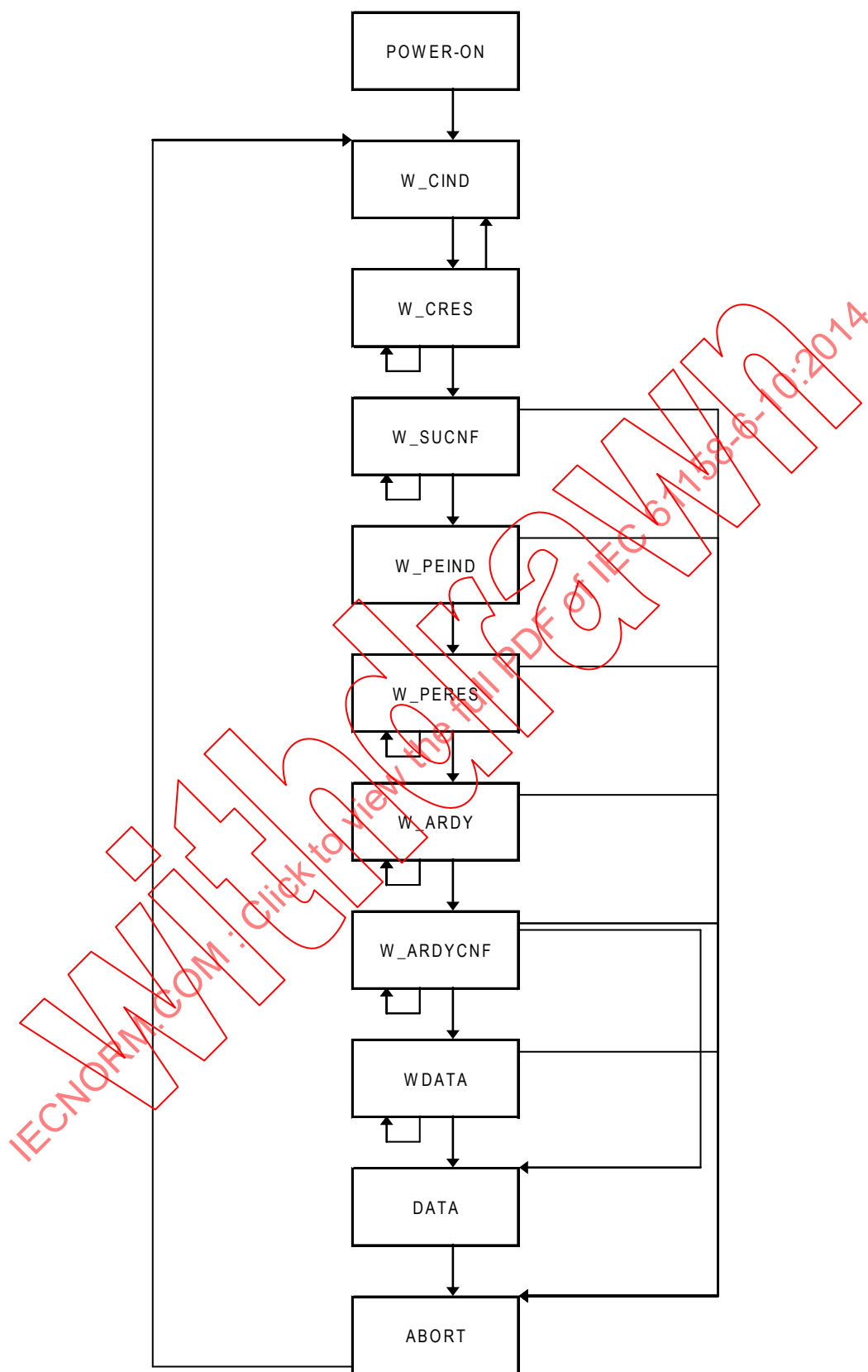
The local service primitives including their associated parameters issued or received by CMDEV are described in the service definition and shown in Table 727.

Table 727 – Local primitives issued or received by CMDEV

Primitive	Source	Destination	Associated parameters	Functions
CMIO_info_ind	CMIO	CMDEV	—	—
CMSU_start_cnf (-)	CMSU	CMDEV	ListofCREPs, ListofErrors	This service primitive confirms the start service.
CMSU_start_cnf (+)	CMSU	CMDEV	ListofCREPs	This service primitive confirms the start service.
CM_Abort_req	FSPMDEV	CMDEV	AREP	The service Abort removes the stored AREP from the list of AREPs.
CM_Init_req	FSPMDEV	CMDEV	—	—
CMSU_start_req	CMDEV	CMSU	—	This service primitive confirms the start service.
CM_Abort_cnf	CMDEV	FSPMDEV	AREP	The service Abort removes the stored AREP from the list of AREPs.
CM_Init_cnf	CMDEV	FSPMDEV	—	—
CMDEV_state_ind	CMDEV	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	state {ABORT, STARTUP, PRMEND, APPLRDY, DATA}	This service primitive controls the state of the AR establishment.

5.6.3.2.1.2 State transition diagram

The state transition diagram of the CMDEV is shown in Figure 100.

**Figure 100 – State transition diagram of CMDEV**

States of the CMDEV

POWER-ON

Data initialization

W_CIND	Wait for connect indication
W_CRES	Wait for connect response from the application and start the CMSU
W_SUCNF	Wait for CMSU confirmation and convey the connect response
W_PEIND	Wait for PrmEnd indication
W_PERES	Wait for PrmEnd response from the application
W_ARDY	Wait for ARDY request from the application
W_ARDYCNF	Wait for ARDY confirmation
WDATA	Wait for the established cyclic data exchange to start the CPM / PPM connection monitoring. The ALPMI is now able to issue an alarm.
DATA	Data exchange and connection monitoring using the CPM / PPM. The ALPMI is now able to issue an alarm.
ABORT	Abort application relation

5.6.3.2.1.3 State machine description

The CM Device protocol machine (CMDEV) is present for each AR of an IO Device.

Two kinds of Ars, selected by the attribute ARProperties.DeviceAccess, handled by CMDEV and CMDEV_DA exists.

Furthermore, if a multicast communication relation (MCR) is selected, the MCITimeoutFactor shall be used to optimize the diagnosis. If a MCR is not running and the MCITimeoutFactor is not expired, the ApplRdy.req shall be delayed until the MCITimeoutFactor expires or if earlier, all MCRs are running.

The possible RPC reruns are handled at the CMRPC.

5.6.3.2.1.4 CMDEV state table

Table 728 contains the complete description of the CMDEV state machine.

Table 728 – CMDEV state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	CM_Init_req () => CM_Init_cnf ()	W_CIND
2	W_CIND	RM_Connect.ind () /CheckAPDU () == OK => Ready4DATA:=FALSE CM_Connect.ind ()	W_CRES
3	W_CIND	RM_Connect.ind () /CheckAPDU () == ErrorDecode, ErrorCode1, ErrorCode2 => RM_Connect.rsp (-)	W_CIND
4	W_CRES	CM_Connect.rsp (+) => CMSU_start_req () //NOTE Start all required protocol machines	W_SUCNF

#	Current State	Event /Condition =>Action	Next State
5	W_CRES	CM_Connect.rsp (-) => CreateLogBookEntry () RM_Connect.rsp (-)	W_CIND
6	W_CRES	RM_Dcontrol.ind () => ignore	W_CRES
7	W_CRES	RM_Release.ind () => ignore	W_CRES
8	W_SUCNF	CMSU_start_cnf (+) => CMDEV_state_ind (STARTUP) RM_Connect.rsp (+)	W_PEIND
9	W_SUCNF	CMSU_start_cnf (-) => ErrorDecode := PNIO ErrorCode1 := CMDEV ErrorCode2 := state conflict CreateLogBookEntry () RM_Connect.rsp (-)	ABORT
10	W_SUCNF	RM_Dcontrol.ind () => ignore	W_SUCNF
11	W_SUCNF	RM_Release.ind () => ignore	W_SUCNF
12	W_PEIND	RM_Dcontrol.ind () /PRMEND => CM_Dcontrol.ind ()	W_PERES
13	W_PEIND	RM_Release.ind () => CM_Release.ind () RM_Release.rsp ()	ABORT
14	W_PERES	CM_Dcontrol.rsp (+) => CMDEV_state_ind (PRMEND) RM_Dcontrol.rsp (+)	W_ARDY
15	W_PERES	CM_Dcontrol.rsp (-) => RM_Dcontrol.rsp (-)	ABORT
16	W_PERES	RM_Dcontrol.ind () => ignore	W_PERES
17	W_PERES	RM_Release.ind () => ignore	W_PERES
18	W_ARDY	CMIO_info_ind () /state == DATA_POSSIBLE => Ready4DATA:=TRUE	W_ARDY
19	W_ARDY	CMIO_info_ind () /state == DATA_IMPOSSIBLE => Ready4DATA:=FALSE	W_ARDY
20	W_ARDY	CM_Ccontrol.req () /APPLRDY && All PPM buffers are updated and valid => CMDEV_state_ind (APPLRDY)	W_ARDYCNF

#	Current State	Event /Condition =>Action	Next State
		RM_Ccontrol.req (APPL_RDY)	
21	W_ARDY	CM_Ccontrol.req () /APPLRDY && ! All PPM buffers are updated and valid => CM_Ccontrol.cnf (-)	W_ARDY
22	W_ARDY	RM_Release.ind () => CM_Release.ind () RM_Release.rsp ()	ABORT
23	W_ARDYCNF	CMIO_info_ind () /state == DATA_POSSIBLE => Ready4DATA:=TRUE	W_ARDYCNF
24	W_ARDYCNF	CMIO_info_ind () /state == DATA_IMPOSSIBLE => Ready4DATA:=FALSE	W_ARDYCNF
25	W_ARDYCNF	RM_Ccontrol.cnf (+) /Ready4DATA == TRUE => CMDEV_state_ind (DATA) CM_Ccontrol.cnf (+) //NOTE The next NewData_ind () after the „Local AR In Data.ind“ contains the first valid data from the IO controller	DATA
26	W_ARDYCNF	RM_Ccontrol.cnf (+) /Ready4DATA == FALSE => CM_Ccontrol.cnf (-)	WDATA
27	W_ARDYCNF	RM_Ccontrol.cnf (-) => CreateLogBookEntry () CM_Ccontrol.cnf (-)	ABORT
28	W_ARDYCNF	RM_Release.ind () => CM_Release.ind () RM_Release.rsp ()	ABORT
29	WDATA	CMIO_info_ind () /state == DATA_POSSIBLE => Ready4DATA:=TRUE CMDEV_state_ind (DATA)	DATA
30	WDATA	CMIO_info_ind () /state == DATA_IMPOSSIBLE => Ready4DATA:=FALSE	WDATA
31	WDATA	RM_Release.ind () => CM_Release.ind () RM_Release.rsp ()	ABORT
32	DATA	RM_Release.ind () => CM_Release.ind () RM_Release.rsp ()	ABORT
33	ABORT	=> CMDEV_state_ind (ABORT)	W_CIND
34	ANY	CMDEV_state_ind () /state == ABORT => ignore	W_CIND

#	Current State	Event /Condition =>Action	Next State
35	ANY	CM_Abort_req () => CMDEV_state_ind (ABORT)	W_CIND

5.6.3.2.1.5 Functions, Macros, Timers and Variables

Table 729 contains the functions, macros, timers and variables used by the CMDEV and their arguments and their descriptions.

Table 729 – Functions, Macros, Timers and Variables used by CMDEV

Name	Type	Function/Meaning
CheckAPDU	Function	This local function checks the semantics of the connect.
Ready4Data	Variable	This local variable is used to store state informations during AR establishment.
All PPM buffer are updated and valid	Macro	This local macro return TRUE if the user has updated the Data_Buffer of the PPMs.

5.6.3.2.2 CMDEV Device Access state machine

5.6.3.2.2.1 General

The CMDEV_DA arranges the device access connection establishment of an IO device. Figure 101 shows the integration of the IO device CM – device access.

The following definition shall be used for ARBlockReq.ARProperties.StartupMode:=Advanced. For ARBlockReq.ARProperties.StartupMode:=Legacy the definitions of the version V2.2 and the Annex B applies.

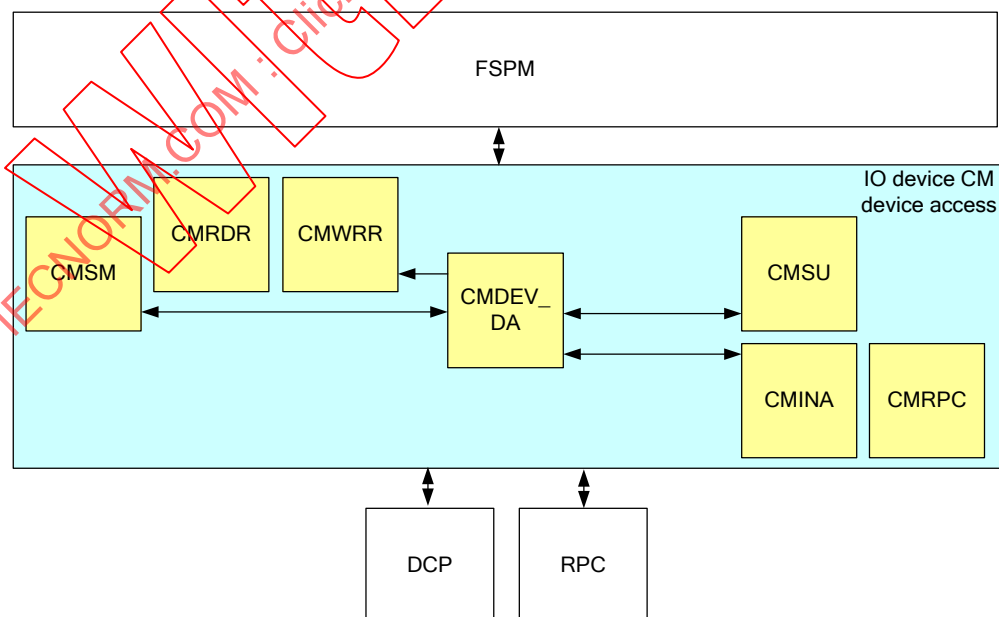


Figure 101 – Scheme of the IO device CM – device access

CMDEV

This state machine handles the context management of the IO device.

CMWRR

This state machine handles the responder functionality of the write service.

CMRDR

This state machine handles the responder functionality of the read service.

CMSM

This state machine handles the connection monitoring during the startup.

CMSU

This state machine handles the startup of the different state machines during startup and shut down.

CMRPC

This machine handles the RPC communication and therefore all issued and received RPC services.

CMINA

This machine handles the IP and Name assignment.

5.6.3.2.2 Primitive definitions

5.6.3.2.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Device Device Access (CMDEV_DA) are described in the service definition and shown in Table 730.

Table 730 – Remote primitives issued or received by CMDEV_DA

Primitive	Source	Destination	Associated parameters	Functions
CM_Connect.ind	CMDEV_DA	FSPMDEV	AREP, ARBlockReq	—
RM_Connect.rsp(-)	CMDEV_DA	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2	—
RM_Connect.rsp(+)	CMDEV_DA	CMRPC	AREP, ARBlockRes	—
RM_Release.rsp(-)	CMDEV_DA	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2	—
RM_Release.rsp(+)	CMDEV_DA	CMRPC	AREP, IODReleaseRes	—
CM_Connect.rsp (-)	FSPMDEV	CMDEV_DA	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Connect.rsp (+)	FSPMDEV	CMDEV_DA	AREP, ARBlockRes	—
RM_Connect.ind	CMRPC	CMDEV_DA	AREP, ARBlockReq	—
RM_Release.ind	CMRPC	CMDEV_DA	AREP, IODReleaseReq	—

5.6.3.2.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMDEV_DA are described in the service definition and shown in Table 731.

Table 731 – Local primitives issued or received by CMDEV_DA

Primitive	Source	Destination	Associated parameters	Functions
CMDEV_state_ind	CMDEV_DA	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	AREP, state {STARTUP, ABORT}	—
CM_Abort_req	FSPMDEV	CMDEV_DA	AREP	—
CM_Abort_cnf	CMDEV_DA	FSPMDEV	AREP	—
CM_Init_req	FSPMDEV	CMDEV_DA	AREP	—
CM_Init_cnf	CMDEV_DA	FSPMDEV	AREP	—

5.6.3.2.2.3 State transition diagram

The state transition diagram of the CMDEV_DA is shown in Figure 100.

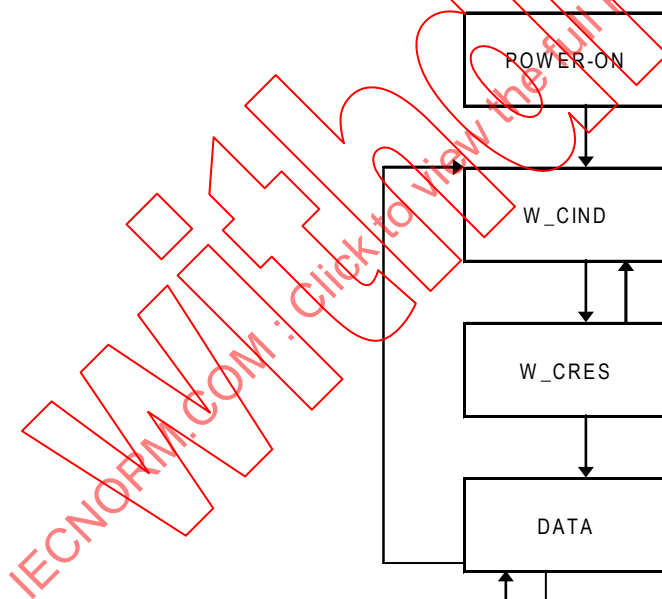


Figure 102 – State transition diagram of CMDEV_DA

States of the CMDEV_DA

POWER-ON

Data initialization

W_CIND

Wait for connect indication

W_CRES

Wait for connect response from the application

DATA

Data exchange monitored by CMSM

5.6.3.2.2.4 State machine description

The CM Device protocol machine for device access (CMDEV_DA) is present for each AR of an IO Device. The selection is done by the attribute ARProperties.DeviceAccess.

The device access is used if only a record service channel to the application is needed. The connection monitoring for this kind of an IOCAR is done by the CMSM using read and write record services.

The CMDEV_DA signals the CMSM only STARTUP when switching to DATA to keep the monitoring using read and write services active.

5.6.3.2.2.5 CMDEV_DA (device access) state table

Table 732 contains the complete description of the CMDEV_DA state machine.

Table 732 – CMDEV_DA state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	CM_Init_req () => CM_Init_cnf ()	W_CIND
2	W_CIND	RM_Connect.ind () /CheckAPDU () == OK => CM_Connect.ind ()	W_CRES
3	W_CIND	RM_Connect.ind () /CheckAPDU () == ErrorDecode, ErrorCode1, ErrorCode2 => RM_Connect.rsp (-)	W_CIND
4	W_CRES	CM_Abort_req () => RM_Connect.rsp (-) CMDEV_state_ind (ABORT) CM_Abort_cnf ()	W_CIND
5	W_CRES	CM_Connect.rsp (+) => RM_Connect.rsp (+) CMDEV_state_ind (STARTUP)	DATA
6	W_CRES	CM_Connect.rsp (-) => RM_Connect.rsp (-)	W_CIND
7	DATA	CM_Abort_req () => CMDEV_state_ind (ABORT) CM_Abort_cnf ()	W_CIND
8	DATA	RM_Release.ind () => RM_Release.rsp () CMDEV_state_ind (ABORT)	W_CIND
9	DATA	CM_Connect.rsp () => ignore	DATA

5.6.3.2.2.6 Functions, Macros, Timers and Variables

Table 733 contains the functions, macros, timers and variables used by the CMDEV(DA) and their arguments and their descriptions.

Table 733 – Functions, Macros, Timers and Variables used by CMDEV(DA)

Name	Type	Function/Meaning
CheckAPDU	Function	This local function checks the semantics of the connect.

5.6.3.3 Context Management Startup Device

5.6.3.3.1 Primitive definitions

5.6.3.3.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Startup Protocol Machine Device (CMSU) are described in the service definition and shown in Table 734.

Table 734 – Remote primitives issued or received by CMSU

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.3.3.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMSU are described in the service definition and shown in Table 735.

Table 735 – Local primitives issued or received by CMSU

Primitive	Source	Destination	Associated parameters	Functions
ALPMI_Activate_req	CMSU	ALPMI	CREP	—
ALPMR_Activate_req	CMSU	ALPMR	CREP	—
APMR_Activate_req	CMSU	APMR	CREP	—
APMS_Activate_req	CMSU	APMS	CREP	—
CMSU_start_cnf (-)	CMSU	CMDEV	ErrorList {ListOfPPM, ListOfCPM, ListOfMCR, ALPMI, ALPMR, DFP}	This service primitive starts the underlying protocol resources.
CMSU_start_cnf (+)	CMSU	CMDEV	—	—
CMDMC_Activate_req	CMSU	CMDMC	CREP.CMDMC, ProviderStationName, CR_Parameter	—
CMDMC_Close_req	CMSU	CMDMC	CREP	—
ALPMI_Activate_cnf (-)	ALPMI	CMSU	—	—
ALPMI_Error_ind	ALPMI	CMSU	CREP, ERRCLS, ERRCODE	—
ALPMR_Activate_cnf (-)	ALPMR	CMSU	—	—

Primitive	Source	Destination	Associated parameters	Functions
ALPMR_Error_ind	ALPMR	CMSU	CREP, ERRCLS, ERRCODE	—
APMR_Activate_cnf (-)	APMR	CMSU	—	—
APMR_Error_ind	APMR	CMSU	CREP, ERRCLS, ERRCODE	—
APMS_Activate_cnf (-)	APMS	CMSU	—	—
APMS_Error_ind	APMS	CMSU	CREP, ERRCLS, ERRCODE	—
CMSU_start_req	CMDEV	CMSU	AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmod uleBlockReq, ListOfMCRBlockReq, ListofSubFrameBlockR eq, ARBlockRes, ListofIOCRBlockRes, AlarmCRBlockRes	This service primitive starts the underlying protocol resources.
CMDMC_Activate_cnf (-)	CMDMC	CMSU	—	—
CMDMC_Error_ind ()	CMDMC	CMSU	ErrorCode1, ErrorCode2	—
CPM_Activate_cnf (-)	CPM	CMSU	—	—
CPM_Error_ind	CPM	CMSU	ErrorCode1, ErrorCode2	—
DFP_Activate_cnf (-)	DFP	CMSU	—	—
PPM_Activate_cnf (-)	PPM	CMSU	—	—
PPM_Error_ind	PPM	CMSU	ErrorCode1, ErrorCode2	—
XXX_Activate_cnf (+)	xxx	CMSU	CREP	—
XXX_Close_cnf (+)	xxx	CMSU	CREP	—
CPM_Activate_req	CMSU	CPM	CREP, DA, SA, FrameID, RxOption, ARProperties.StartupM ode, ...	—
CPM_Close_req	CMSU	CPM	CREP	—
DFP_Activate_req	CMSU	DFP	CREP	—
DFP_Close_req	CMSU	DFP	CREP	—
CMDEV_state_ind	CMDEV CMSU	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	State {ABORT, STARTUP, PRMEND, APPLRDY, DATA}	This service primitive controls the state of the CMSU.

Primitive	Source	Destination	Associated parameters	Functions
PPM_Activate_req	CMSU	PPM	CREP, DA, SA, FrameID, TxOption, ARProperties.StartupMode, ...	—
PPM_Close_req	CMSU	PPM	CREP	—

5.6.3.3.2 State transition diagram

The state transition diagram of the CMSU is shown in Figure 103.

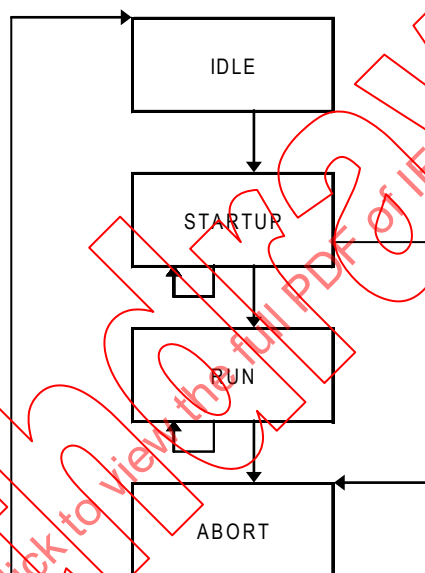


Figure 103 – State transition diagram of CMSU

States of the CMSU

IDLE	Wait for the start command and start the associated state machines.
STARTUP	Check the response of the started machines and continue in state RUN.
RUN	Startup of the state machines is done, wait for shut down or fault.
ABORT	Termination sequence

5.6.3.3.3 State machine description

The Context Management Startup Protocol Machine Device (CMSU) of an IO device arranges the start of the underlying protocol machines.

The startup of the CPM and PPM state machines shall be done between receiving the connect request service and the Application ready service. The startup shall be finish before issuing the Application ready request service.

5.6.3.3.4 CMSU state table

Table 736 contains the description of the CMSU state machine. The difference between ARProperties.StartupMode=Legacy and ARProperties.StartupMode=Advanced is handled by the CMSU, the PPM and the CPM.

Table 736 – CMSU state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CMSU_start_req () => CREP.PPM := create (PPM) CREP.CPM := create (CPM) CREP.ALPMI := create (ALPMI) CREP.ALPMR := create (ALPMR) CREP.APMS := create (APMS) CREP.APMR := create (APMR) if exist: CREP.MPPM := create (PPM) CREP.CMDMC := create (CMDMC) CREP.DFP := create (DFP) //NOTE Create high and low alarm machine instances PPM_Activate_req () CPM_Activate_req () if applicable CMDMC_Activate_req () if applicable DFP_Activate_req () ALPMI_Activate_req () ALPMR_Activate_req () APMS_Activate_req () APMR_Activate_req ()	STARTUP
2	STARTUP	XXX_Activate_cnf (+) //if last confirmation is received => CMSU_start_cnf (+)	STARTUP
3	STARTUP	PPM_Activate_cnf (-) => ErrorCode2 = AR add provider or consumer failed CMSU_start_cnf (-)	STARTUP
4	STARTUP	CPM_Activate_cnf (-) => ErrorCode2 = AR add provider or consumer failed CMSU_start_cnf (-)	STARTUP
5	STARTUP	ALPMI_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMSU_start_cnf (-)	STARTUP
6	STARTUP	ALPMR_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMSU_start_cnf (-)	STARTUP
7	STARTUP	APMS_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMSU_start_cnf (-)	STARTUP
8	STARTUP	APMR_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMSU_start_cnf (-)	STARTUP

#	Current State	Event /Condition =>Action	Next State
9	STARTUP	CMDMC_Activate_cnf (-) => ErrorCode1 := CMDEV ErrorCode2 := state conflict CMSU_start_cnf (-)	STARTUP
10	STARTUP	DFP_Activate_cnf (-) => ErrorCode2 = AR add provider or consumer failed CMSU_start_cnf (-)	STARTUP
11	STARTUP	CMSU_start_req () => ErrorCode1 := CMDEV ErrorCode2 := state conflict CMSU_start_cnf (-)	STARTUP
12	STARTUP	CMDEV_state_ind () /state != ABORT => ignore	RUN
13	STARTUP	CMDEV_state_ind () /state == ABORT => PPM_Close_req () CPM_Close_req () if applicable CMDMC_Close_req () if applicable DFP_Close_req () ALPMI_Close_req () ALPMR_Close_req () APMS_Close_req () APMR_Close_req ()	ABORT
14	RUN	CMDEV_state_ind () /state != ABORT => ignore	RUN
15	RUN	CMDEV_state_ind () /state == ABORT => PPM_Close_req () CPM_Close_req () if applicable CMDMC_Close_req () if applicable DFP_Close_req () ALPMI_Close_req () ALPMR_Close_req () APMS_Close_req () APMR_Close_req ()	ABORT
16	RUN	CMSU_start_req () => ErrorCode1 := CMDEV ErrorCode2 := state conflict CMSU_start_cnf (-)	RUN

#	Current State	Event /Condition =>Action	Next State
17	ABORT	XXX_Close_cnf (+) /if last confirmation is received => Delete (PPM) Delete (CPM) Delete (ALPMI) Delete (ALPMR) Delete (APMS) Delete (APMR) if exist: Delete (DFP) if exist: Delete (CMDMC) //NOTE Delete all created PPMs and CPMs	IDLE
18	ANY	PPM_Error_ind () => CMDEV_state_ind (ABORT)	SAME
19	ANY	CPM_Error_ind () => CMDEV_state_ind (ABORT)	SAME
20	ANY	ALPMI_Error_ind () => ErrorCode2 := AR alarm-send or ErrorCode2 := AR alarm-ack-send CMDEV_state_ind (ABORT)	SAME
21	ANY	ALPMR_Error_ind () => ErrorCode2 := AR alarm-ind CMDEV_state_ind (ABORT)	SAME
22	ANY	APMS_Error_ind () => ErrorCode2 := AR alarm-send or ErrorCode2 := AR alarm-ack-send CMDEV_state_ind (ABORT)	SAME
23	ANY	APMR_Error_ind () => ErrorCode2 := AR alarm-ind CMDEV_state_ind (ABORT)	SAME
24	ANY	CMDMC_Error_ind () => CMDEV_state_ind (ABORT)	SAME

5.6.3.3.5 Functions, Macros, Timers and Functions

Table 737 contains the functions, macros, timers and variables used by the CMSU and their arguments and their descriptions.

Table 737 – Functions, Macros, Timers and Variables used by the CMSU

Name	Type	Function/Meaning
Delete	Function	This local function frees or unbounds the machines from the CMCTL instance if the AR is aborted.
If last confirmation is received	Macro	This local macro collects all the confirmations and returns TRUE if all receive with xxx_cnf (+).

5.6.3.4 Context Management Input Output Device

5.6.3.4.1 Primitive definitions

5.6.3.4.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Input Output Protocol Machine Device (CMIO) are described in the service definition and shown in Table 738.

Table 738 – Remote primitives issued or received by CMIO

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.3.4.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by Context Management Input Output Protocol Machine Device (CMIO) are described in the service definition and shown in Table 739.

Table 739 – Local primitives issued or received by CMIO

Primitive	Source	Destination	Associated parameters	Functions
CMIO_info_ind	CMIO	CMDEV	info	
CPM_State_ind	CPM	CMIO	—	
CPM_NewData_ind	CPM	CMIO FSPMDEV CMDMC CTLIO	ARER, CREP, APDU_Status, data {Data, NoData}	This service primitive indicates an update of a receive buffer.
CMDEV_state_ind	CMDEV	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	State {ABORT, STARTUP, PRMEND, APPLRDY}	This service primitive controls the state of the CMIO.

5.6.3.4.2 State transition diagram

The state transition diagram of the CMIO is shown in Figure 104.

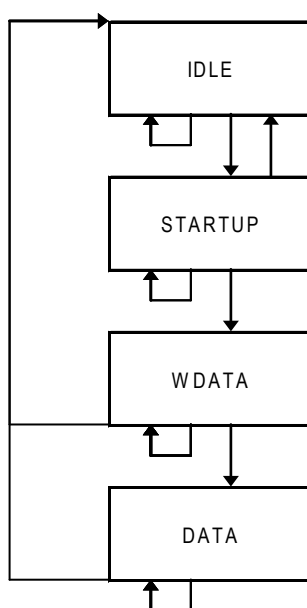


Figure 104 – State transition diagram of CMIO

States of the CMIO

IDLE

No AR context exists

STARTUP

Initial parameterization phase

WDATA

Wait until the AR is ready to switch to the DATA state due to all associated CPMs are receiving frames.

DATA

Data exchange phase. Monitoring the DataHoldTime.

5.6.3.4.3 State machine description

The CM Input Output protocol machine (CMIO) arranges the access to the DataStatus and NewData indication.

5.6.3.4.4 CMIO state table

Table 740 contains the complete description of the CMIO state machine.

Table 740 – CMIO state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CPM_NewData_ind () => ignore	IDLE
2	IDLE	CPM_State_ind () => ignore	IDLE
3	IDLE	CMDEV_state_ind () /state == ABORT => ignore	IDLE

#	Current State	Event /Condition =>Action	Next State
4	IDLE	CMDEV_state_ind () /state == STARTUP => For all CPMs CmInstance[CREP.CPM]:=Stop	STARTUP
5	STARTUP	CPM_NewData_ind () => ignore	STARTUP
6	STARTUP	CPM_State_ind () => ignore	STARTUP
7	STARTUP	CMDEV_state_ind () /state == ABORT => ignore	IDLE
8	STARTUP	CMDEV_state_ind () /state == PRMEND => StartTimer ()	WDATA
9	WDATA	CPM_NewData_ind () => CmInstance[CREP.CPM]:=Start	WDATA
10	WDATA	CPM_State_ind () /state == STOP => CmInstance[CREP.CPM]:=Stop	WDATA
11	WDATA	CPM_State_ind () /state == START => CmInstance[CREP.CPM]:=Start	WDATA
12	WDATA	TimerExpired () /All CPM of CmInstance[CREP.CPM] contain the value "Start" => StartTimer () CMIO_info_ind (DATA_POSSIBLE)	WDATA
13	WDATA	TimerExpired () /One CPM of CmInstance[CREP.CPM] contains the value "Stop" => StartTimer () CMIO_info_ind (DATA_IMPOSSIBLE)	WDATA
14	WDATA	CMDEV_state_ind () /state == ABORT => ignore	IDLE
15	WDATA	CMDEV_state_ind () /state == APPLRDY => ignore	WDATA
16	WDATA	CMDEV_state_ind () /state == DATA => StopTimer ()	DATA
17	DATA	CPM_NewData_ind () => ignore	DATA
18	DATA	CPM_State_ind () /state == STOP && CREP != CREP.MCPM => CMDEV_state_ind (ABORT)	DATA

#	Current State	Event /Condition =>Action	Next State
19	DATA	CPM_State_ind () /state == START => ignore	DATA
20	DATA	CMDEV_state_ind () /state == ABORT => ignore	IDLE

5.6.3.4.5 Functions, Macros, Timers and Variables

Table 741 contains the functions, macros, timers and variables used by the CMIO and their arguments and their descriptions.

Table 741 – Functions used by the CMIO

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or restarts a timer. The time for this timer may be 100 ms. NOTE An implementation can use "change" events instead of this timer.
StopTimer	Function	This local functions stops a timer
TimerExpired	Function	This local function is issued if a timer expires
CmInstance	Variable	This AR global variable contains the informations about the AR.

5.6.3.5 Context Management Write Record Device

5.6.3.5.1 Primitive definitions

5.6.3.5.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Write Record Responder Protocol Machine Device (CMWRR) are described in the service definition and shown in Table 742.

Table 742 – Remote primitives issued or received by CMWRR

Primitive	Source	Destination	Associated parameters	Functions
CM_Write.rsp (-)	FSPMDEV	CMWRR	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Write.rsp (+)	FSPMDEV	CMWRR	AREP, SeqNumber, AddData1, AddData2	—
RM_Write.ind	CMRPC	CMWRR	AREP, API, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data	—

Primitive	Source	Destination	Associated parameters	Functions
CM_Write.ind	CMWRR	FSPMDEV	AREP, API, SlotNumber, SubslotNumber, Index, PrmFlag, SeqNumber, Length, Data	—
RM_Write.rsp (-)	CMWRR	CMRPC	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Write.rsp (+)	CMWRR	CMRPC	AREP, SeqNumber	—

5.6.3.5.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMWRR are described in the service definition and shown in Table 743.

Table 743 – Local primitives issued or received by CMWRR

Primitive	Source	Destination	Associated parameters	Functions
CMDEV_state_ind	CMDEV	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	state	—

5.6.3.5.2 State transition diagram

The state transition diagram of the CMWRR is shown in Figure 105.

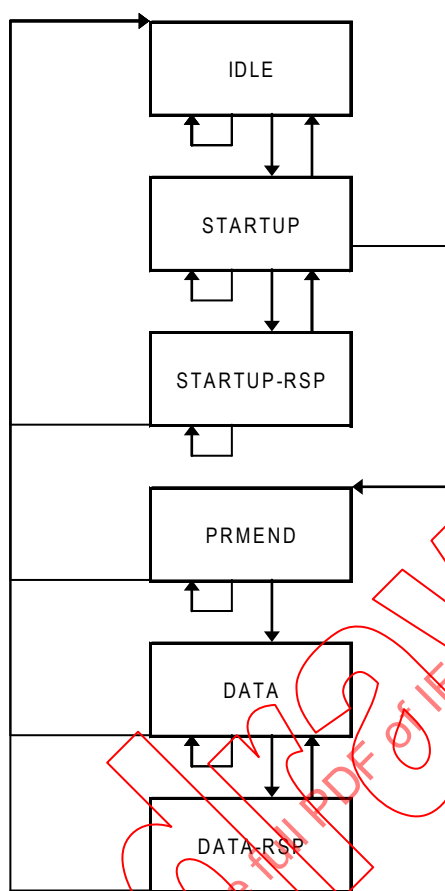


Figure 105 – State transition diagram of CMWRR

States of the CMWRR

IDLE	No AR context exists
STARTUP	The associated AR reaches the internal state STARTUP. Write services handled during this state.
STARTUP-RSP	Wait for the user response and convey it to the requester.
PRMEND	The associated AR reaches the internal state PRMEND. Write services rejected during this state.
DATA	The associated AR reaches the internal state DATA. Write services handled during this state.
DATA-RSP	Wait for the user response and convey it to the requester.

5.6.3.5.3 State machine description

The CM Write Record protocol machine (CMWRR) arranges the write record service.

5.6.3.5.4 CMWRR state table

Table 744 contains the complete description of the CMWRR state machine. The CMWRR checks whether the ARUUID exists.

Table 744 – CMWRR state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	RM_Write.ind () => ErrorDecode := PNORW ErrorCode1 := state conflict RM_Write.rsp (-)	IDLE
2	IDLE	CM_Write.rsp () => ignore	IDLE
3	IDLE	CMDEV_state_ind () /state == STARTUP => ignore	STARTUP
4	IDLE	CMDEV_state_ind () /state == ABORT => ignore	IDLE
5	STARTUP	RM_Write.ind () /Arstate != Backup => CM_Write.ind ()	STARTUP-RSP
6	STARTUP	RM_Write.ind () /Arstate == Backup => ErrorDecode := PNORW ErrorCode1 := backup RM_Write.rsp (-)	STARTUP
7	STARTUP-RSP	CM_Write.rsp (+) => RM_Write.rsp (+)	STARTUP
8	STARTUP-RSP	CM_Write.rsp (-) => RM_Write.rsp (-)	STARTUP
9	STARTUP-RSP	RM_Write.ind () => ignore	STARTUP-RSP
10	STARTUP-RSP	CMDEV_state_ind () /state == ABORT => ignore	IDLE
11	STARTUP	CMDEV_state_ind () /state == PRMEND => ignore	PRMEND
12	STARTUP	CMDEV_state_ind () /state == ABORT => ignore	IDLE
13	PRMEND	RM_Write.ind () => ErrorDecode := PNORW ErrorCode1 := state conflict RM_Write.rsp (-)	PRMEND
14	PRMEND	CMDEV_state_ind () /state == APPLRDY => ignore	DATA
15	PRMEND	CMDEV_state_ind () /state == ABORT => ignore	IDLE

#	Current State	Event /Condition =>Action	Next State
16	DATA	CMDEV_state_ind () /state == DATA => ignore	DATA
17	DATA	RM_Write.ind () /Arstate != Backup => CM_Write.ind ()	DATA-RSP
18	DATA	RM_Write.ind () /Arstate == Backup => ErrorDecode := PNORW ErrorCode1 := backup RM_Write.rsp (-)	DATA
19	DATA	CMDEV_state_ind () /state == ABORT => ignore	IDLE
20	DATA-RSP	CM_Write.rsp (+) => RM_Write.rsp (+)	DATA
21	DATA-RSP	CM_Write.rsp (-) => RM_Write.rsp (-)	DATA
22	DATA-RSP	RM_Write.ind () => ignore	DATA-RSP
23	DATA-RSP	CMDEV_state_ind () /state == DATA => ignore	DATA-RSP
24	DATA-RSP	CMDEV_state_ind () /state == ABORT => ignore	IDLE

5.6.3.5.5 Functions, Macros, Timers and Variables

Table 745 contains the functions, macros, timers and variables used by the CMWRR and their arguments and their descriptions.

Table 745 – Functions, Macros, Timers and Variables used by CMWRR

Name	Type	Function/Meaning
Arstate	Variable	This AR global variable contains the current state of the AR. Possible values are: First, Primary, Backup Initial value of this variable is Primary

5.6.3.6 Context Management Read Record Device

5.6.3.6.1 Primitive definitions

5.6.3.6.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Read Record Responder Protocol Machine Device (CMRDR) are described in the service definition and shown in Table 746.

Table 746 – Remote primitives issued or received by CMRDR

Primitive	Source	Destination	Associated parameters	Functions
CM_Read.rsp (-)	FSPMDEV	CMRDR	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Read.rsp (+)	FSPMDEV	CMRDR	AREP, SeqNumber, Length, Data	—
RM_Read.ind	CMRPC	CMRDR	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Read.ind	CMRDR	FSPMDEV	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Read.rsp (-)	CMRDR	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.rsp (+)	CMRDR	CMRPC	AREP, SeqNumber, Length, Data	—

5.6.3.6.1.2 Primitives exchanged between local machines

The local primitives including their associated parameters issued or received by CMRDR are described in the service definition and shown in Table 747.

Table 747 – Local primitives issued or received by CMRDR

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.3.6.2 State transition diagram

The state transition diagram of the CMRDR is shown in Figure 106.

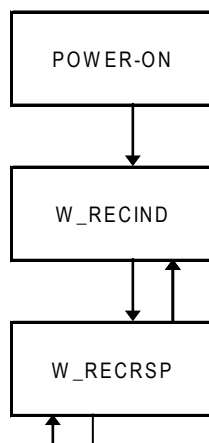


Figure 106 – State transition diagram of CMRDR

States of the CMRDR

POWER-ON

Data initialization

W_RECIND

Wait for read record indication

W_RECRSP

Wait for read record response

5.6.3.6.3 State machine description

The CM Read Record protocol machine (CMRDR) arranges the read record service.

5.6.3.6.4 CMRDR state table

Table 748 contains the complete description of the CMRDR state machine. The CMRPC checks whether the ARUUD exists.

Table 748 – CMRDR state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> Init	W_RECIND
2	W_RECIND	RM_Read.ind () => CM_Read.ind ()	W_RECRSP
3	W_RECRSP	CM_Read.rsp (+) => RM_Read.rsp (+)	W_RECIND
4	W_RECRSP	CM_Read.rsp (-) => RM_Read.rsp (-)	W_RECIND
5	W_RECRSP	RM_Read.ind () => ignore	W_RECRSP

The serialization of the “Read” RPC calls for one AR is done by the RPC layer. Thus, it is not necessary to check the serialization again in the CMRDR.

5.6.3.6.5 Functions, Macros, Timers and Variables

Table 749 contains the functions, macros, timers and variables used by the CMRDR and their arguments and their descriptions.

Table 749 – Functions, Macros, Timers and Variables used by CMRDR

Name	Type	Function/Meaning
—	—	—

5.6.3.7 Context Management Surveillance Device

5.6.3.7.1 Primitive definitions

5.6.3.7.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Surveillance Protocol Machine Device (CMSM) are described in the service definition and shown in Table 750.

Table 750 – Remote primitives issued or received by CMSM

Primitive	Source	Destination	Associated parameters	Functions
CM_Read.ind	CMRDR	CMSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Read.rsp	CMRDR	CMSM	AREP, SeqNumber	—
CM_Write.ind	CMWRR	CMSM	AREP, API, SlotNumber, SubslotNumber, Index, PrmFlag, SeqNumber, Length, Data	—
CM_Write.rsp	CMWRR	CMSM	AREP, SeqNumber	—
RM_Read.ind	CMRPC	CMSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	Trigger record indication This service primitive is the indication of the Read service.
CM_Abort.req	FSPMDEV CMSU CMIO CMSM	CMDEV	AREP	Abort current AR establishing

Primitive	Source	Destination	Associated parameters	Functions
RM_Read.rsp (+)	CMSM	CMRPC	AREP, API, TargetUUID, SeqNumber, Length, Data	Trigger record response This service primitive is the response to the Read service.

5.6.3.7.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMSM are described in the service definition and shown in Table 751.

Table 751 – Local primitives issued or received by CMSM

Primitive	Source	Destination	Associated parameters	Functions
CMDEV_state_ind	CMDEV	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	state	State "PRMEND" start CMI timeout State "DATA" start CMI timeout

5.6.3.7.2 State transition diagram

The state transition diagram of the CMSM is shown in Figure 107.

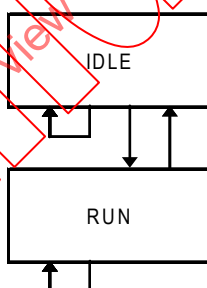


Figure 107 – State transition diagram of CMSM

States of the CMSM

IDLE	Wait for the start of the connection monitoring
RUN	Connection monitoring running

5.6.3.7.3 State machine description

The CM server protocol machine (CMSM) arranges the connection monitoring between the connect response and the start of the PPM / CPM connection monitoring.

The CMSM uses Read, Write and Dcontrol indication and response for the triggering of the connection monitoring.

5.6.3.7.4 CMSM state table

Table 752 contains the complete description of the CMSM state machine.

Table 752 – CMSM state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CMDEV_state_ind () /state == STARTUP => StartTimer (CMI, CMInitiatorActivityTimeout)	RUN
2	IDLE	CMDEV_state_ind () /state != STARTUP => ignore	IDLE
3	IDLE	TimerExpired (CMI) => ignore	IDLE
4	IDLE	RM_Read.ind () /Index == Trigger => RM_Read.rsp (+)	IDLE
5	IDLE	RM_Read.ind () /Index != Trigger => ignore	IDLE
6	IDLE	CM_Read.ind () => ignore	IDLE
7	IDLE	CM_Read.rsp () => ignore	IDLE
8	IDLE	CM_Write.ind () => ignore	IDLE
9	IDLE	CM_Write.rsp () => ignore	IDLE
10	RUN	TimerExpired (CMI) => CM_Abort.req ()	IDLE
11	RUN	CMDEV_state_ind () /state == DATA state == ABORT => StopTimer (CMI)	IDLE
12	RUN	CMDEV_state_ind () /state != DATA && state != ABORT => ignore	RUN
13	RUN	RM_Read.ind () /Index == Trigger => StopTimer (CMI) StartTimer (CMI, CMInitiatorActivityTimeout) RM_Read.rsp (+)	RUN
14	RUN	CM_Read.rsp () => StartTimer (CMI, CMInitiatorActivityTimeout)	RUN
15	RUN	CM_Write.rsp () => StartTimer (CMI, CMInitiatorActivityTimeout)	RUN

#	Current State	Event /Condition =>Action	Next State
16	RUN	CM_Read.ind () => StopTimer (CMI)	RUN
17	RUN	CM_Write.ind () => StopTimer (CMI)	RUN

5.6.3.7.5 Functions, Macros, Timers and Variables

Table 753 contains the functions, macros, timers and variables used by the CMSM and their arguments and their descriptions.

Table 753 – Functions, Macros, Timers and Variables used by the CMSM

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or restarts a timer.
StopTimer	Function	This local function stops a timer.
TimerExpired	Function	This local function is issued if a timer expires.
Trigger	Macro	This local Macro represents the Trigger record.
CMI	Timer	This local timer shall be loaded with the CMInitiatorActivityTimeout and is used for the connection monitoring during startup.

5.6.3.8 Context Management Prm Begin End Device

5.6.3.8.1 Primitive definitions

5.6.3.8.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Prm Begin End Protocol Machine Device (CMPBE) are described in the service definition and shown in Table 754.

Table 754 – Remote primitives received by CMPBE

Primitive	Source	Destination	Associated parameters	Functions
CM_Ccontrol.req	FSPMDEV	CMPBE	AREP, ControlBlock, ModuleDiffBlock	—
CM_Dcontrol.rsp	FSPMDEV	CMPBE	AREP, ControlBlock	—
RM_Ccontrol.cnf	CMRPC	CMPBE	AREP, ControlBlock	—
RM_Dcontrol.ind	CMRPC	CMPBE	AREP, ControlBlock	—
CM_Ccontrol.cnf	CMPBE	FSPMDEV	AREP, ControlBlock	—
CM_Dcontrol.ind	CMPBE	FSPMDEV	AREP, ControlBlock	—
RM_Ccontrol.req	CMPBE	CMRPC	AREP, ControlBlock, ModuleDiffBlock	—
RM_Dcontrol.rsp (-)	CMPBE	CMRPC	AREP, ErrorDecode, ErrorCode1,	—

Primitive	Source	Destination	Associated parameters	Functions
			ErrorCode2, AddData1, AddData2	
RM_Dcontrol.rsp (+)	CMPBE	CMRPC	AREP, ControlBlock	—

5.6.3.8.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMPBE are described in the service definition and shown in Table 755.

Table 755 – Local primitives issued or received by CMPBE

Primitive	Source	Destination	Associated parameters	Functions
CMDEV_state_ind	CMDEV	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	state	—
CM_Abort_req	CMPBE	FSPMDEV CMSU CMIO CMSM	AREP	—

5.6.3.8.2 State transition diagram

The state transition diagram of the CMPBE is shown in Figure 108.

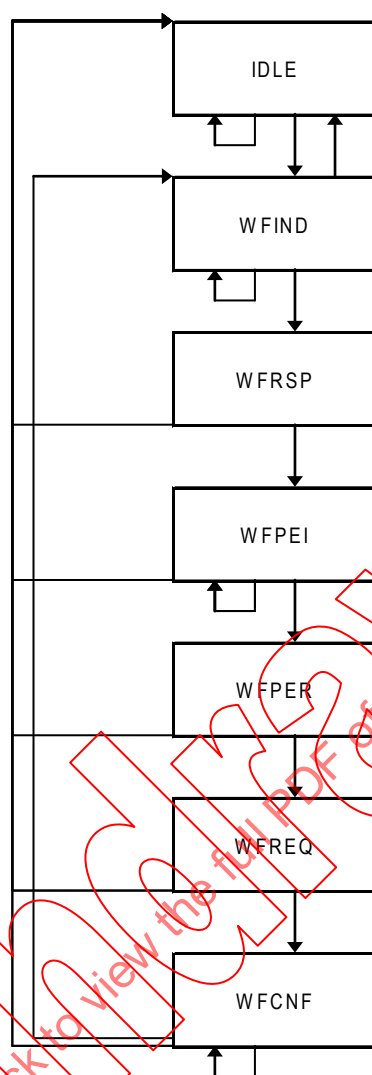


Figure 108 – State transition diagram of CMPBE

States of the CMPBE

IDLE	Wait for the start of DATA
WFIND	Wait for a PrmBegin command
WFRSP	Wait for a PrmBegin response from the application
WFPEI	Wait for the records from the IO controller and the PrmEnd request
WPPER	Wait for a PrmEnd response from the application
WFREQ	Wait for an update of the IO data of the affected submodules before issuing the ApplRdy service.
	Wait for the application ready request from the application
WFCNF	Wait for an application ready confirmation and after that, for the next CPM_NewData_ind from the affected Submodules / IOCR for valid data.

5.6.3.8.3 State machine description

The PrmBegin PrmEnd ApplRdy sequence is used to make a consistent update of the parameters of submodules using the wellknown ApplRdy model. The IO device handles this sequence by the CMPBE.

The IO device stores all Alarms from the involved submodules during the PrmBegin PrmEnd ApplRdy sequence.

Any overlapping Alarm, except the Plug or Release, shall be proceed and acknowledged by the IO controller before the IO device issues the ApplRdy. Overlapping Plug and Release Alarms shall be stored by the IO controller and processed after the PrmBegin PrmEnd ApplRdy sequence.

The IO device rates all stored Alarms from the involved submodules and removes all dispensable Alarms before issuing ApplRdy.

5.6.3.8.4 CMPBE state table

Table 756 contains the complete description of the CMPBE state machine.

Table 756 – CMPBE state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CMDEV_state_ind () /state == DATA => ignore	WFIND
2	IDLE	CMDEV_state_ind () /state != DATA => ignore	IDLE
3	WFIND	RM_Dcontrol.ind () /Stored request exists => Retrieve stored request CM_Dcontrol.ind ()	WFRSP
4	WFIND	RM_Dcontrol.ind () /ControlBlock == PrmBegin => AlarmProceedingBlocked //NOTE Do not proceed any Alarm of an involved submodule CM_Dcontrol.ind ()	WFRSP
5	WFIND	RM_Dcontrol.ind () /ControlBlock != PrmBegin => ignore	WFIND
6	WFIND	CMDEV_state_ind () /state == ABORT => ignore	IDLE
7	WFRSP	CM_Dcontrol.rsp () /ControlBlock == PrmBegin => RM_Dcontrol.rsp (+)	WFPEI
8	WFRSP	CMDEV_state_ind () /state == ABORT => ignore	IDLE
9	WFPEI	RM_Dcontrol.ind () /ControlBlock == PrmEnd => CM_Dcontrol.ind ()	WPPER
10	WFPEI	RM_Dcontrol.ind () /ControlBlock != PrmEnd =>	WFPEI

#	Current State	Event /Condition =>Action	Next State
		ignore	
11	WFPEI	CMDEV_state_ind () /state == ABORT => ignore	IDLE
12	WPPER	CM_Dcontrol.rsp () /ControlBlock == PrmEnd => RM_Dcontrol.rsp (+)	WFREQ
13	WPPER	CMDEV_state_ind () /state == ABORT => ignore	IDLE
14	WFREQ	CM_Ccontrol.req () /ControlBlock == ApplRdy && AlarmAckPending == TRUE => CM_Ccontrol.cnf (-)	WFREQ
15	WFREQ	CM_Ccontrol.req () /ControlBlock == ApplRdy && AlarmAckPending == FALSE => RM_Ccontrol.req ()	WFCNF
16	WFREQ	RM_Dcontrol.ind () /ControlBlock == PrmBegin //NOTE PrmEnd is handled by CMRPC as RPC Rerun => RM_Dcontrol.rsp (-) CMDEV_state_ind (ABORT)	IDLE
17	WFREQ	CMDEV_state_ind () /state == ABORT => ignore	IDLE
18	WFCNF	RM_Ccontrol.cnf () /ControlBlock == ApplRdy => AlarmProceedingAllowed //NOTE Continue with the proceeding of Alarms CM_Ccontrol.cnf ()	WFIND
19	WFCNF	RM_Dcontrol.ind () /ControlBlock == PrmBegin => Store request //NOTE Execute the stored request after ApplRdy.cnf to make the PBE sequence robust	WFCNF
20	WFCNF	CMDEV_state_ind () /state == ABORT => ignore	IDLE

5.6.3.8.5 Functions, Macros, Timers and Variables

Table 757 contains the functions, macros, timers and variables used by the CMPBE and their arguments and their descriptions.

Table 757 – Functions, Macros, Timers and Variables used by the CMPBE

Name	Type	Function/Meaning
AlarmAckPending	Macro	This local macro checks whether an involved submodule (list given by the PrmBegin.req) has an outstanding Alarm acknowledge for any Alarm except Plug or Release. TRUE := an outstanding Alarm acknowledge exists FALSE := no outstanding Alarm acknowledge exists
AlarmProceedingAllowed	Macro	This local macro continues the proceeding of all kind of Alarms for any involved submodule (list given by the PrmBegin).
AlarmProceedingBlocked	Macro	This local macro stops the proceeding of all kind of Alarms for any involved submodule (list given by the PrmBegin).
Retrieve stored request	Macro	This local macro retrieves the stored request.
Store request	Macro	This local macro stores the request for later use.
Stored request exists	Macro	This local macro checks whether a request is stored.

5.6.3.9 Context Management Discovery Multicast Communication

5.6.3.9.1 Primitive definitions

5.6.3.9.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Discovery Multicast Communication Protocol Machine (CMDMC) are described in the service definition and shown in Table 758.

Table 758 – Remote primitives issued or received by CMDMC

Primitive	Source	Destination	Associated parameters	Functions
DCP_Identify.cnf (-)	DCPMCS	CMDMC	CREP, ERRCLS, ERRCODE	This service primitive indicates that the address resolution of a M-Provider station name has failed.
DCP_Identify.cnf (+)	DCPMCS	CMDMC	CREP	This service primitive indicates that the address resolution of a M-Provider station name has succeeded.
DCP_Identify.req	CMDMC	DCPMCS	CREP, DA, ListOfFilter, ResponseDelay	The service Identify starts the address resolution of the station name of a M-Provider

5.6.3.9.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMDMC are described in the service definition and shown in Table 759.

Table 759 – Local primitives issued or received by CMDMC

Primitive	Source	Destination	Associated parameters	Functions
CMDMC_Activate_req	CMSU	CMDMC	CREP, ProviderStationName, CR_Parameter	The service activate initializes the CMDMC state machine.
CMDMC_Close_req	CMSU	CMDMC	CREP	The service close deinitializes the CMDMC state machine.

Primitive	Source	Destination	Associated parameters	Functions
CPM_Activate_cnf (-)	CPM	CMDMC	CREP, ERRCLS, ERRCODE	This service primitive indicates that the Activate service failed.
CPM_Activate_cnf (+)	CPM	CMDMC	CREP	This service primitive indicates that the Activate service succeeded.
CPM_Close_cnf	CPM	CMDMC	CREP	This service primitive indicates that the Close service succeeded.
CPM_NewData_ind	CPM	CMDMC FSPMDEV FSPMCTL CMIO CTLIO	AREP, CREP, APDU_Status, data {Data, NoData}	This service primitive indicates that new Multicast Consumer Data have been received.
CPM_State_ind	CPM	CMDMC FSPMDEV FSPMCTL CMIO CTLIO	CREP	This service primitive indicates that Data Hold Time has expired.
CMDMC_Error_ind ()	CMDMC	CMSU	CREP, ERRCLS, ERRCODE	This service primitive indicates that a problem exists.
CMDMC_Activate_cnf (-)	CMDMC	CMSU	CREP, ERRCLS, ERRCODE	This service primitive indicates that the Activate service failed.
CMDMC_Activate_cnf (+)	CMDMC	CMSU	CREP	This service primitive indicates that the Activate service succeeded.
CMDMC_Close_cnf	CMDMC	CMSU	CREP	This service primitive indicates that the Close service succeeded.
CPM_Close_req	CMSU	CPM	CREP	The service Activate deinitializes the CPM and stops the transmission of data. The schedule is cleared.
CPM_Activate_req	CMDMC	CPM	CREP, DA, SA, FrameId, Prio, VLAN, Exp_Lenth, DataHoldTime, Default_Value, Default_Status	The service Activate initializes the CPM and loads the schedule to the DMPM.

5.6.3.9.2 State transition diagram

The state transition diagram of the CMDMC is shown in Figure 109.

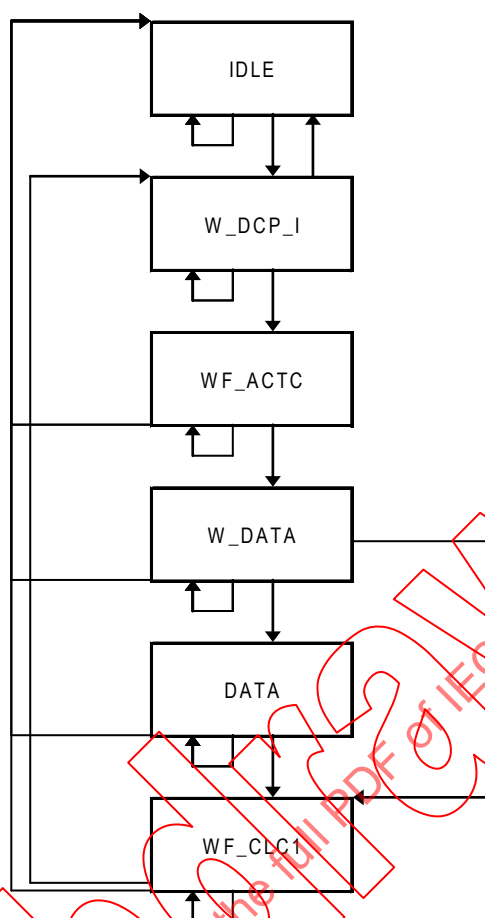


Figure 109 – State transition diagram of CMDMC

States of the CMDMC

IDLE	Wait for the start command
W_DCP_I	Wait for the DCP Identify service response
WF_ACTC	Wait for the activation of the CPM
W_DATA	Wait for the NewData_ind of the CPM
DATA	Working state
WF_CLC1	Wait for the shut down of former started state machines

5.6.3.9.3 State machine description

The Discovery Multicast Communication Protocol Machine (CMDMC) arranges the multicast communication between IO devices. This is done by discovering the communication partners and starting of the associated CPMs. The discovery uses the DCP Identify service.

5.6.3.9.4 CMDMC state table

Table 760 contains the complete description of the CMDMC state machine.

Table 760 – CMDMC state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CMDMC_Activate_req () => MCName := ProviderStationName DA := DCPMC_add ListofFilter := DeviceProperties, NameOfStation, MCName ResponseDelay := 1 For all Submodules within M-Consumer-CRs do: Add entry to the diagnosis ASE: Submodule ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out CMDMC_Activate_cnf (+) DCP_Identify.req ()	W_DCP_I
2	IDLE	CMDMC_Close_req () => CMDMC_Close_cnf ()	IDLE
3	IDLE	OTHERS => ignore	IDLE
4	W_DCP_I	DCP_Identify.cnf () /No device found => DA := DCPMC_add ListofFilter := DeviceProperties, NameOfStation, MCName ResponseDelay := 1 For all Submodules within M-Consumer-CRs do: Add entry to the diagnosis ASE: ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := AddressResolutionFailed DCP_Identify.req () DiagnosisEvent ()	W_DCP_I
5	W_DCP_I	DCP_Identify.cnf () /Device found => CREP := create(CPM) MC_InData(CREP) := FALSE CPM_Activate_req ()	WF_ACTC
6	W_DCP_I	CMDMC_Close_req () => CMDMC_Close_cnf ()	IDLE
7	W_DCP_I	OTHERS => ignore	W_DCP_I
8	WF_ACTC	CPM_Activate_cnf (+) => ignore	W_DATA
9	WF_ACTC	CPM_Activate_cnf (-) => CMDMC_Error_ind () CPM_Close_req ()	IDLE
10	WF_ACTC	CMDMC_Close_req () => CMDMC_Close_cnf ()	IDLE

#	Current State	Event /Condition =>Action	Next State
11	WF_ACTC	OTHERS => ignore	WF_ACTC
12	W_DATA	CPM_NewData_ind () /CMDEV_ApplRdy == FALSE => MC_InData(CREP) := TRUE For all Submodule within M-Consumer-CRs do: Delete M-Consumer "Multicast CR Mismatch" from the diagnosis ASE	W_DATA
13	W_DATA	CPM_NewData_ind () /CMDEV_ApplRdy == TRUE => MC_InData(CREP) := TRUE For all Submodule within M-Consumer-CRs do: Delete M-Consumer "Multicast CR Mismatch" from the diagnosis ASE	DATA
14	W_DATA	CPM_State_ind () /state==stop => MC_InData(CREP) := FALSE For all Submodules within M-Consumer-CRs do: Add entry to the diagnosis ASE: Submodule ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out DiagnosisEvent () CPM_Close_req ()	WF_CLC1
15	W_DATA	CPM_State_ind () /state!=stop => ignore	W_DATA
16	W_DATA	CMDMC_Close_req () => CMDMC_Close_cnf ()	IDLE
17	W_DATA	OTHERS => ignore	W_DATA
18	DATA	CPM_State_ind () /state==stop => MC_InData(CREP) := FALSE For all Submodules within M-Consumer-CRs do: Add entry to the diagnosis ASE: Submodule ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out DiagnosisEvent () CPM_Close_req ()	WF_CLC1
19	DATA	CPM_State_ind () /state!=stop => ignore	DATA
20	DATA	CMDMC_Close_req () => CMDMC_Close_cnf ()	IDLE
21	DATA	CPM_NewData_ind () => ignore	DATA

#	Current State	Event /Condition =>Action	Next State
22	DATA	OTHERS => ignore	DATA
23	WF_CLC1	CPM_Close_cnf () => DCP_Identify.req ()	W_DCP_I
24	WF_CLC1	CMDMC_Close_req () => CMDMC_Close_cnf ()	IDLE
25	WF_CLC1	OTHERS => ignore	WF_CLC1

5.6.3.9.5 Functions, Macros, Timers and Variables

Table 761 contains the functions, macros, timers and variables used by the CMDMC and their arguments and their descriptions.

Table 761 – Functions, Macros, Timers and Variables used by the CMDMC

Name	Type	Function/Meaning
CMDEV_ApplRdy	Variable	This local variable shows whether the AR is in state DATA or not
MC_InData	Variable	This local variable stores the information whether the CPM is in state RUN or not
DiagnosisEvent	Function	This function issues an entry to the diagnosis ASE
OTHERS	Macro	This local macro contains all other possible events, not shown explicitly, in this state

5.6.3.10 Context Management IP and Name Assignment

5.6.3.10.1 Primitive definitions

5.6.3.10.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management IP and Name Assignment Protocol Machine (CMINA) are described in the service definition and shown in Table 772.

Table 762 – Remote primitives issued or received by CMINA

Primitive	Source	Destination	Associated parameters	Functions
DCP_Set.ind	DCP	CMINA	CREP, SA, ListOfData, ListOfControlCommands	—
DHCPOFFER.ind	DHCP	CMINA	IP_Para, ListOfData	—
DCP_Set.rsp (-)	CMINA	DCP	CREP, DA, ERRCLS, ERRCODE	—
DCP_Set.rsp (+)	CMINA	DCP	CREP, DA, ListOfResponse	—

Primitive	Source	Destination	Associated parameters	Functions
DCP_HELLO.req	CMINA	DCPHMCS	CREP, ListOfData	—
DCP_HELLO.cnf ()	DCP	CMINA	—	—
DHCPDISCOVER.req	CMINA	DHCP	Options	—

5.6.3.10.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMINA are described in the service definition and shown in Table 763.

Table 763 – Local primitives issued or received by CMINA

Primitive	Source	Destination	Associated parameters	Functions
CMDEV_state_ind	CMINA	CM*	—	Cancellation information for the AR to release resources

5.6.3.10.2 State transition diagram

The state transition diagram of the CMINA is shown in Figure 110.

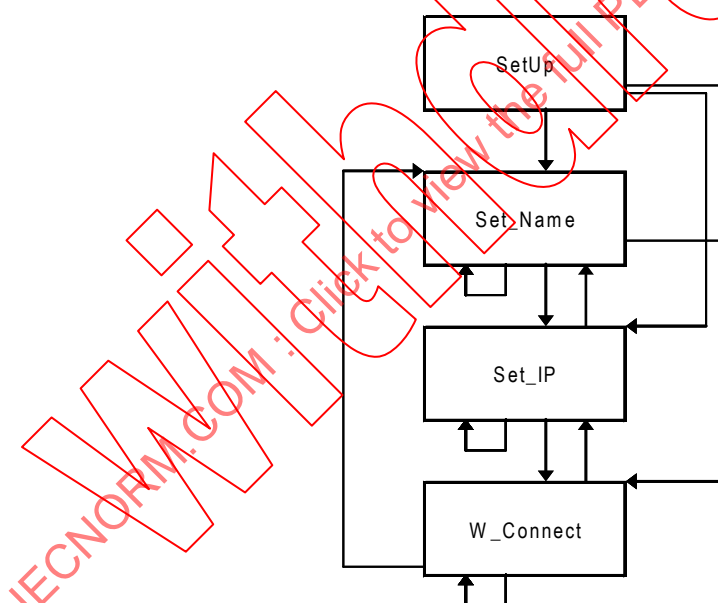


Figure 110 – State transition diagram of CMINA

States of the CMINA

SetUp	Wait for NameOfStation and IP-suite and start the discovery according to the stored values of the DCP ASE
Set_Name	Wait for a NameOfStation
Set_IP	Wait for a IP-suite
W_Connect	Wait for the AR establishment

5.6.3.10.3 State machine description

The Context Management IP and Name Assignment Protocol Machine (CMINA) arranges the IP and station name assignment and handles the DCP Set, DHCP OFFER, DCP HELLO and the DHCP DISCOVER services.

The FSHelloDelay shall be handled independent from this state machine.

5.6.3.10.4 CMINA state table

Table 764 contains the complete description of the CMINA state machine.

Table 764 – CMINA state table

#	Current State	Event /Condition =>Action	Next State
1	SetUp	/LocalNameOfStation == NIL && LocalDHCPenable => StartTimer (DHCP_RetryTime) DHCPDISCOVER.req ()	Set_Name
2	SetUp	/LocalNameOfStation == NIL && !LocalDHCPenable => ignore	Set_Name
3	SetUp	/LocalNameOfStation != NIL && LocalIPsuite == NIL && LocalDHCPenable => StartTimer (DHCP_RetryTime) DHCPDISCOVER.req ()	Set_IP
4	SetUp	/LocalNameOfStation != NIL && LocalIPsuite == NIL && LocalHELLOenable => StartTimer (HelloIntervalTime) HelloCount := FSHelloRetry DCP_HELLO.req ()	Set_IP
5	SetUp	/LocalNameOfStation != NIL && LocalIPsuite != NIL && LocalHELLOenable => StartIP () StartTimer (HelloIntervalTime) HelloCount := FSHelloRetry DCP_HELLO.req ()	W_Connect
6	SetUp	/LocalNameOfStation != NIL && LocalIPsuite == NIL &&	Set_IP

#	Current State	Event /Condition =>Action	Next State
		!LocalDHCPenable => ignore	
7	SetUp	/LocalNameOfStation != NIL && LocalIPsuite != NIL && !LocalHELLOenable => StartIP ()	W_Connect
8	Set_Name	DHCPOFFER.ind () /CheckAPDU () == Valid, Name, IP => StopTimer (DHCP_RetryTime) Store Data in DCP ASE StartIP ()	W_Connect
9	Set_Name	DHCPOFFER.ind () /CheckAPDU () == Valid => Store Data in DCP ASE	Set_Name
10	Set_Name	DHCPOFFER.ind () /CheckAPDU () == Invalid => ignore	Set_Name
11	Set_Name	DCP_Set.ind () /CheckAPDU () == Valid, Name, IP => StopTimer (DHCP_RetryTime) Store Data in DCP ASE StartIP () DCP_Set.rsp (+)	W_Connect
12	Set_Name	DCP_Set.ind () /CheckAPDU () == Valid, Name, DHCP => StartTimer (DHCP_RetryTime) Store Data in DCP ASE Reset IP suite in DCP ASE DCP_Set.rsp (+) DHCPDISCOVER.req ()	Set_IP
13	Set_Name	DCP_Set.ind () /CheckAPDU () == Valid, Name && LocalIPsuite == NIL => Store Data in DCP ASE DCP_Set.rsp (+)	Set_IP
14	Set_Name	DCP_Set.ind () /CheckAPDU () == Valid, Name && LocalIPsuite != NIL => Store Data in DCP ASE DCP_Set.rsp (+)	W_Connect
15	Set_Name	DCP_Set.ind () /CheckAPDU () == Valid CheckAPDU () == Valid, ResetIP => Store Data in DCP ASE DCP_Set.rsp (+)	Set_Name
16	Set_Name	DCP_Set.ind ()	Set_Name

#	Current State	Event /Condition =>Action	Next State
		/CheckAPDU () == Valid, Unknown => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Set.rsp (-)	
17	Set_Name	DCP_Set.ind () /CheckAPDU () == InValid => ignore	Set_Name
18	Set_Name	TimerExpired (DHCP_RetryTime) => ExponentialBackOff (DHCP_RetryTime) StartTimer (DHCP_RetryTime) DHCPDISCOVER.req ()	Set_Name
19	Set_Name	DCP_Hello.cnf () => ignore	Set_Name
20	Set_IP	DHCP OFFER.ind () /CheckAPDU () == Valid, IP => StopTimer (DHCP_RetryTime) Store Data in DCP ASE StartIP ()	W_Connect
21	Set_IP	DHCP OFFER.ind () /CheckAPDU () == Valid CheckAPDU () == Valid, ResetIP => Store Data in DCP ASE	Set_IP
22	Set_IP	DHCP OFFER.ind () /CheckAPDU () == InValid => ignore	Set_IP
23	Set_IP	DCP_Set.ind () /CheckAPDU () == Valid, IP => StopTimer (DHCP_RetryTime) Store Data in DCP ASE StartIP () DCP_Set.rsp (+)	W_Connect
24	Set_IP	DCP_Set.ind () /CheckAPDU () == Valid, ResetName CheckAPDU () == Valid, ResetToFactory => Reset assigned Data in DCP ASE DCP_Set.rsp (+)	Set_Name
25	Set_IP	DCP_Set.ind () /CheckAPDU () == Valid => Store Data in DCP ASE DCP_Set.rsp (+)	Set_IP
26	Set_IP	DCP_Set.ind () /CheckAPDU () == Valid, Unknown => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Set.rsp (-)	Set_IP
27	Set_IP	DCP_Set.ind () /CheckAPDU () == InValid =>	Set_IP

#	Current State	Event /Condition =>Action	Next State
		ignore	
28	Set_IP	TimerExpired (DHCP_RetryTime) => ExponentialBackOff (DHCP_RetryTime) StartTimer (DHCP_RetryTime) DHCPDISCOVER.req ()	Set_IP
29	Set_IP	TimerExpired (HelloIntervalTime) /HelloCount != 0 => StartTimer (HelloIntervalTime) HelloCount := HelloCount – 1 DCP_HELLO.req ()	Set_IP
30	Set_IP	TimerExpired (HelloIntervalTime) /HelloCount == 0 => ignore	Set_IP
31	Set_IP	DCP_Hello.cnf () => ignore	Set_IP
32	W_Connect	DHCPOFFER.ind () /CheckAPDU () == Valid, ChangeIP && CheckARActive () == None => StopIP () Store Data in DCP ASE StartIP ()	W_Connect
33	W_Connect	DHCPOFFER.ind () /CheckAPDU () == Valid, ChangeIP && CheckARActive () == Aractive => ignore	W_Connect
34	W_Connect	DCP_Set.ind () /CheckAPDU () == Valid => Store Data in DCP ASE DCP_Set.rsp (+)	W_Connect
35	W_Connect	DCP_Set.ind () /(CheckAPDU () == Valid, ResetName CheckAPDU () == Valid, ResetToFactory) && CheckARActive () == None => StopIP () Reset assigned Data in DCP ASE DCP_Set.rsp (+)	Set_Name
36	W_Connect	DCP_Set.ind () /(CheckAPDU () == Valid, ResetName CheckAPDU () == Valid, ResetToFactory) && CheckARActive () == Aractive => StopIP () Reset assigned Data in DCP ASE DCP_Set.rsp (+) For all active Ars CMDEV_state_ind (ABORT)	Set_Name
37	W_Connect	DCP_Set.ind () /(	Set_IP

#	Current State	Event /Condition =>Action	Next State
		CheckAPDU () == Valid, ChangeName CheckAPDU () == Valid, ResetIP) && CheckARActive () == None ==> StopIP () Store Data in DCP ASE Reset IP suite in DCP ASE DCP_Set.rsp (+)	
38	W_Connect	DCP_Set.ind () /(CheckAPDU () == Valid, ChangeName CheckAPDU () == Valid, ResetIP) && CheckARActive () == Aractive ==> StopIP () Store Data in DCP ASE Reset IP suite in DCP ASE DCP_Set.rsp (+) For all active Ars CMDEV_state_ind (ABORT)	Set_IP
39	W_Connect	DCP_Set.ind () /CheckAPDU () == Valid, ChangelP && CheckARActive () == None ==> StopIP () Store Data in DCP ASE StartIP () DCP_Set.rsp (+)	W_Connect
40	W_Connect	DCP_Set.ind () /CheckAPDU () == Valid, ChangelP && CheckARActive () == Aractive ==> ERRCLS := Invalid_Parameter ERRCODE := Invalid_State DCP_Set.rsp (-)	W_Connect
41	W_Connect	DCP_Set.ind () /CheckAPDU () == Valid, Unknown ==> ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Set.rsp (-)	W_Connect
42	W_Connect	DCP_Set.ind () /CheckAPDU () == InValid ==> ignore	W_Connect
43	W_Connect	TimerExpired (HelloIntervalTime) /HelloCount != 0 ==> StartTimer (HelloIntervalTime) HelloCount := HelloCount – 1 DCP_HELLO.req ()	W_Connect
44	W_Connect	TimerExpired (HelloIntervalTime) /HelloCount == 0 ==> ignore	W_Connect
45	W_Connect	DCP_Hello.cnf () ==> ignore	W_Connect

5.6.3.10.5 Functions, Macros, Timers and Variables

Table 765 and Table 766 contains the functions, macros, timers and variables used by the CMINA and their arguments and their descriptions.

Table 765 – Functions, Macros, Timers and Variables used by the CMINA

Name	Type	Function/Meaning
CheckAPDU	Function	This local function checks the block structure and the parameters according to the coding and checking rules.
CheckARactive	Function	This local function checks whether an AR is active.
ExponentialBackOff	Function	This local function changes the DHCP_RetryTime according to the DHCP specification.
StartIP	Function	This local function assigns an IP suite to the IP stack.
StartTimer	Function	This local function starts or restarts a timer.
StopIP	Function	This local function withdraws the IP suite from the IP stack.
StopTimer	Function	This local function stops a timer.
TimerExpired	Function	This local function is executed if a timer expires.
LocalHELLOenable	Macro	This local macro request the HELLO state from the DCP ASE and returns NIL if it not exists.
LocalIPsuite	Macro	This local macro request the IP suite from the DCP ASE and returns NIL if it not exists.
LocalNameOfStation	Macro	This local macro request the NameOfStation from the DCP ASE and returns NIL if it not exists.
DHCP_RetryTime	Timer	This timer controls the interval between two DHCP requests.
HelloIntervalTime	Timer	This timer controls the interval between two HELLO requests given by the FSHelloInterval.
ERRCLS	Variable	This local variable stores the error class for the error PDU.
ERRCODE	Variable	This local variable stores the error code for the error PDU.
HelloCount	Variable	This local variable count down the conveyed Hello request from the initial value FSHelloRetry.

Table 766 – Return values of CheckAPDU

Name	Function/Meaning
ChangeIP	ChangeIP states that the frame contains an IP-suite and an IP-suite is already stored.
ChangeName	ChangeName states that the frame contains a NameOfStation and a NameOfStation is already stored.
DHCP	DHCP states that the frame contains DHCP enable.
InValid	InValid states that the frame is malformed according this standard.
IP	IP states that the frame contains an IP-suite.
Name	Name states that the frame contains the NameOfStation.
ResetIP	ResetIP states that the frame contains a delete IP-suite command.
ResetName	ResetName states that the frame contains a delete NameOfStation command.
ResetToFactory	ResetToFactory states that the frame contains a delete NameOfStation command.
Unknown	Unknown states that the frame contains unknown options or suboptions.
Valid	Valid states that the frame is correct according this standard.

5.6.3.11 Context Management RPC Device

5.6.3.11.1 Primitive definitions

5.6.3.11.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management RPC Device Protocol Machine (CMRPC) are described in the service definition and shown in Table 767.

Table 767 – Remote primitives issued or received by CMRPC

Primitive	Source	Destination	Associated parameters	Functions
RM_Ccontrol.req	CMDEV	CMRPC	AREP, ControlBlock, ModuleDiffBlock	
RM_Connect.rsp (-)	CMDEV	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Connect.rsp (+)	CMDEV	CMRPC	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock	—
RM_Dcontrol.rsp (-)	CMDEV	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Dcontrol.rsp (+)	CMDEV	CMRPC	AREP, ControlBlock	—
RM_Read.rsp (-)	CMDEV	CMRPC	AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.rsp (+)	CMDEV	CMRPC	AREP, SeqNumber, AddData1, AddData2	—
RM_Release.rsp (-)	CMDEV	CMRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Release.rsp (+)	CMDEV	CMRPC	AREP, ControlBlock	—
RM_Write.rsp (-)	CMDEV	CMRPC	AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—

Primitive	Source	Destination	Associated parameters	Functions
RM_Write.rsp (+)	CMDEV	CMRPC	AREP, SeqNumber, AddData1, AddData2	—
RM_Ccontrol.cnf (-)	CMRPC	CMDEV	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Ccontrol.cnf (+)	CMRPC	CMDEV, CMPBE	AREP, ControlBlock	—
RM_Connect.ind	CMRPC	CMDEV, CMPBE	AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq	—
RM_Dcontrol.ind	CMRPC	CMDEV, CMDEV_DA	AREP, ControlBlock	—
RM_Read.ind	CMRPC	CMDEV, CMRDR, CMSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Release.ind	CMRPC	CMDEV, CMDEV_DA	AREP, ControlBlock	—
RM_Write.ind	CMRPC	CMDEV, CMWRR	AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data	—
RPC_Ccontrol.req	CMRPC	RPC	Arg	—
RPC_Connect.rsp (-)	CMRPC	RPC	Arg	—
RPC_Connect.rsp (+)	CMRPC	RPC	Arg	—
RPC_Dcontrol.rsp (-)	CMRPC	RPC	Arg	—
RPC_Dcontrol.rsp (+)	CMRPC	RPC	Arg	—
RPC_Read.rsp (-)	CMRPC	RPC	Arg	—
RPC_Read.rsp (+)	CMRPC	RPC	Arg	—
RPC_Release.rsp (-)	CMRPC	RPC	Arg	—
RPC_Release.rsp (+)	CMRPC	RPC	Arg	—
RPC_Write.rsp (-)	CMRPC	RPC	Arg	—
RPC_Write.rsp (+)	CMRPC	RPC	Arg	—
RPC_Ccontrol.cnf (-)	RPC	CMRPC	Arg	—
RPC_Ccontrol.cnf (+)	RPC	CMRPC	Arg	—
RPC_Connect.ind	RPC	CMRPC	Arg	—
RPC_Dcontrol.ind	RPC	CMRPC	Arg	—
RPC_Read.ind	RPC	CMRPC	Arg	—
RPC_Release.ind	RPC	CMRPC	Arg	—
RPC_Write.ind	RPC	CMRPC	Arg	—

5.6.3.11.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMRPC are described in the service definition and shown in Table 768.

Table 768 – Local primitives issued or received by CMRPC

Primitive	Source	Destination	Associated parameters	Functions
CMDEV_state_ind	CMDEV	FSPMDEV CMSU CMIO CMWRR CMSM CMPBE CMRPC	AREP, state	Cancellation information for the AR to release resources

5.6.3.11.2 State transition diagram

The state transition diagram of the CMRPC is shown in Figure 111.

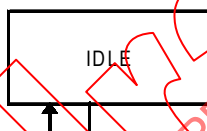


Figure 111 – State transition diagram of CMRPC

States of the CMRPC

IDLE Wait for a RPC call, check its integrity, the resources and for idempotent rerun

5.6.3.11.3 State machine description

The Context Management RPC Device Protocol Machine (CMRPC) arranges the RPC communication and handles all issued and received RPC services.

5.6.3.11.4 CMRPC state table

Table 769 contains the complete description of the CMRPC state machine.

Table 769 – CMRPC state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	RM_Ccontrol.req () => encode RPC PDU RPC_Ccontrol.req ()	IDLE
2	IDLE	RPC_Ccontrol.cnf (-) /CheckRPC () == Valid => AREP := LocateAR () RM_Ccontrol.cnf (-)	IDLE

#	Current State	Event /Condition =>Action	Next State
3	IDLE	RPC_Ccontrol.cnf (+) /CheckRPC () == Valid => AREP := LocateAR () RM_Ccontrol.cnf (+)	IDLE
4	IDLE	RPC_Ccontrol.cnf () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => AREP := LocateAR () RM_Ccontrol.cnf (-)	IDLE
5	IDLE	RPC_Dcontrol.ind () /CheckRPC () == Rerun => Retrieve stored response encode RPC PDU RPC_Dcontrol.rsp ()	IDLE
6	IDLE	RPC_Dcontrol.ind () /CheckRPC () == Valid, ExplicitAR => AREP := LocateAR () RM_Dcontrol.ind ()	IDLE
7	IDLE	RPC_Dcontrol.ind () /CheckRPC () == Valid, UnknownAR => ErrorCode2 := ARUUID_Unknown encode RPC Error PDU RPC_Dcontrol.rsp (-)	IDLE
8	IDLE	RPC_Dcontrol.ind () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode RPC Error PDU RPC_Dcontrol.rsp (-)	IDLE
9	IDLE	RM_Dcontrol.rsp (+) => encode RPC PDU Store response RPC_Dcontrol.rsp (+)	IDLE
10	IDLE	RM_Dcontrol.rsp (-) => encode Error PDU Store response RPC_Dcontrol.rsp (-)	IDLE
11	IDLE	RPC_Write.ind () /CheckRPC () == Valid, ExplicitAR => AREP := LocateAR () RM_Write.ind ()	IDLE
12	IDLE	RPC_Write.ind () /CheckRPC () == Valid, UnknownAR => ErrorCode2 := ARUUID_Unknown encode Error PDU RPC_Write.rsp (-)	IDLE
13	IDLE	RPC_Write.ind () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode RPC Error PDU RPC_Write.rsp (-)	IDLE
14	IDLE	RM_Write.rsp (+) => encode RPC PDU RPC_Write.rsp (+)	IDLE

#	Current State	Event /Condition =>Action	Next State
15	IDLE	RM_Write.rsp (-) => encode Error PDU RPC_Write.rsp (-)	IDLE
16	IDLE	RPC_Read.ind () /CheckRPC () == Valid, ImplicitAR => AREP := NIL RM_Read.ind ()	IDLE
17	IDLE	RPC_Read.ind () /CheckRPC () == Valid, ExplicitAR => AREP := LocateAR () RM_Read.ind ()	IDLE
18	IDLE	RPC_Read.ind () /CheckRPC () == Valid, UnknownAR => ErrorCode2 := ARUUID_Unknown encode Error PDU RPC_Read.rsp (-)	IDLE
19	IDLE	RPC_Read.ind () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode RPC Error PDU RPC_Read.rsp (-)	IDLE
20	IDLE	RM_Read.rsp (+) => encode RPC PDU RPC_Read.rsp (+)	IDLE
21	IDLE	RM_Read.rsp (-) => encode Error PDU RPC_Read.rsp (-)	IDLE
22	IDLE	RPC_Connect.ind () /CheckRPC () == Valid, UnknownAR && CheckResource () == Unavailable => ErrorCode2 := NO_AR_RESOURCE encode RPC Error PDU RPC_Connect.rsp (-)	IDLE
23	IDLE	RPC_Connect.ind () /CheckRPC () == Valid, UnknownAR, UnknownARset && CheckResource () == Available => AllocateResource () RM_Connect.ind ()	IDLE
24	IDLE	RPC_Connect.ind () /CheckRPC () == Valid, UnknownAR, ExplicitARset //NOTE Preallocated resources by the first AR of a AR set => RM_Connect.ind ()	IDLE
25	IDLE	RPC_Connect.ind () /CheckRPC () == Valid, UnknownAR, NoARset && CheckResource () == Available => AllocateResource () RM_Connect.ind ()	IDLE

#	Current State	Event /Condition =>Action	Next State
26	IDLE	RPC_Connect.ind () /CheckRPC () == Valid, ExplicitAR => ErrorCode2 := StateConflict encode RPC Error PDU RPC_Connect.rsp (-) CMDEV_state_ind (ABORT)	IDLE
27	IDLE	RPC_Connect.ind () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode RPC Error PDU RPC_Connect.rsp (-)	IDLE
28	IDLE	RM_Connect.rsp (+) => encode RPC PDU RPC_Connect.rsp (+)	IDLE
29	IDLE	RM_Connect.rsp (-) => FreeResource () encode Error PDU RPC_Connect.rsp (-)	IDLE
30	IDLE	RPC_Release.ind () /CheckRPC () == Valid, ExplicitAR => AREP := LocateAR () RM_Release.ind ()	IDLE
31	IDLE	RPC_Release.ind () /CheckRPC () == Valid, UnknownAR => ErrorCode2 := ARUID_Unknown encode RPC Error PDU RPC_Release.rsp (-)	IDLE
32	IDLE	RPC_Release.ind () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode RPC Error PDU RPC_Release.rsp (-)	IDLE
33	IDLE	RM_Release.rsp (+) => FreeResource () encode RPC PDU RPC_Release.rsp (+)	IDLE
34	IDLE	RM_Release.rsp (-) => FreeResource () encode Error PDU RPC_Release.rsp (-)	IDLE
35	IDLE	CMDEV_state_ind () /state == ABORT && CheckAREP () == Known => FreeResource ()	IDLE
36	IDLE	CMDEV_state_ind () /state == ABORT && CheckAREP () == Unknown => ignore	IDLE
37	IDLE	CMDEV_state_ind () /state != ABORT => ignore	IDLE

5.6.3.11.5 Functions, Macros, Timers and Variables

Table 770 and Table 771 contains the functions, macros, timers and variables used by the CMRPC and their arguments and their descriptions.

Table 770 – Functions, Macros, Timers and Variables used by the CMRPC

Name	Type	Function/Meaning
AllocateResource	Function	This local function allocates the necessary resources for the AR using ARType, number of IOCRs and amount of IO data. An AR of the ARType IOCARSR allocates the resources for the complete AR set.
CheckAREP	Function	This local function checks whether the AREP exists.
CheckResource	Function	This local function checks the availability of the necessary resources for the connect indication using ARType, number of IOCRs and amount of IO data. Special case "Out of ARset resources" For the first application relation having ARType IOCARSR and belonging to a new ARSet (as indicated by the ARUUIID) the resources of the complete ARSet are allocated, i.e. for all ARs of the complete ARSet. The resources of an application relation belonging to the same ARSet established at some later point are assigned from this allocated memory. The resources of the ARSet are freed when the last application relation of the ARSet is terminated. Special case "ARType specific resource handling" The resource checking shall cover the resource model shown in the GSD which optimizes the usage of the existing resources of a IO device.
CheckRPC	Function	This local functions checks the block structure and parameters against the definitions of this standard. Special case "Shared IO device" The first established AR defines the SendClockFactor of the physical device. If a succeeding AR owns an unreachable SendClockFactor, this function returns InValid and a parameter error for the SendClockFactor or the ReductionRatio shall be created in the connect response negative. Unreachable may be a different SendClockFactor or a not representable combination of SendClockFactor and ReductionRatio. Special case "Shared IO device with RT_CLASS_3" The first established AR defines the IRDataUUIID of the physical device. If a succeeding AR owns a different IRDataUUIID, this function returns InValid and a parameter error for the IRDataUUIID shall be created in the connect response negative. Special case "Startup of the ARs of an ARset" If the CMDEV already establishes one AR of the set, then the CMRPC rejects a new AR connection request until ApplReady.cnf is reached. Also an AR connection request of a new AR of the set is rejected if the last AR of set is shutting down. Used ErrorCode2 := "ARset – State conflict during connection establishment"
FreeResource	Function	This local function frees the allocated resources of the AR.
Encode Error PDU	Macro	This local macro generates the appropriate PDU with the detected errors from the PDU checking rules.
Encode RPC Error PDU	Macro	This local macro generates the appropriate PDU with the detected errors from the PDU checking rules. A RPC error PDU consists only out of the RPC NDR header.
Encode RPC PDU	Macro	This local macro generates the appropriate PDU.
LocateAR	Macro	This local macro retrieves the associated AREP
Retrieve stored response	Macro	This local macro retrieves the previously stored parameter in case of a detected RPC rerun.

Name	Type	Function/Meaning
Store response	Macro	This local macro stores the parameters of the RPC response to be retrievable in case of a RPC rerun
AREP	Variable	This local variable stores the application relation end point for further use

Table 771 – Return values of CheckRPC

Name	Function/Meaning
Valid	Valid states that the frame is correct according this standard.
InValid	InValid states that the frame is incorrect.
ExplicitAR	This AR, identified by the ARUUID, already exists.
UnknownAR	This AR, identified by the ARUUID, doesn't exist.
ImplicitAR	This services uses the implicit access to the device.
UnknownARset	This AR, identified by the ARUUID, is part of an ARset which doesn't exist.
ExplicitARset	This AR, identified by the ARUUID, is part of an ARset which already exists.
NoARset	This AR, identified by the ARUUID, is not part of an ARset.

5.6.4 Controller

5.6.4.1 General

5.6.4.1.1 General

The CMCTL arrange the connection establishment of an IO controller. Figure 112 shows the integration of the IO controller CM.

The following definition shall be used for ARBlockReq.ARProperties.StartupMode:=Advanced. For ARBlockReq.ARProperties.StartupMode:=Legacy the definitions of the version V2.2 and the Annex B applies.

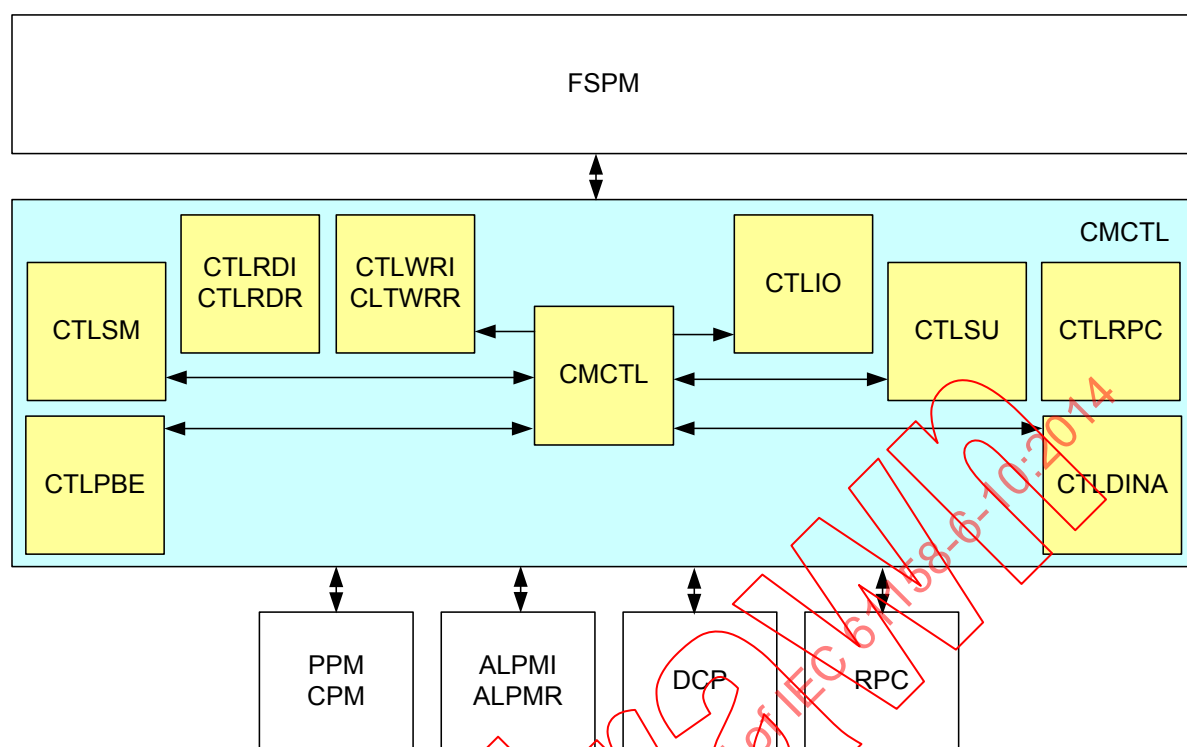


Figure 112 – Scheme of the IO controller CM

CMCTL

This state machine handles the context management of the IO controller.

CTLWRI

This state machine handles the initiator functionality of the write service.

CTLWRR

This optional state machine handles the responder functionality of the write service.

CTLRDI

This state machine handles the initiator functionality of the read service.

CTLRDR

This optional state machine handles the responder functionality of the read service.

CTLSM

This state machine handles the connection monitoring during the startup.

CTLPBE

This optional state machine handles the PrmBegin, PrmEnd and ApplRdy sequence for system redundancy and configure in run.

CTLSU

This state machine handles the startup of the different state machines during startup and shut down.

CTLRPC

This state machine handles the translation of RPC services.

CTLIO

This state machine handles the IO data services.

CTLDINA

This machine handles the discovery, IP and station name assignment.

5.6.4.1.2 State transition diagram

The state transition diagram of the CMCTL is shown in Figure 113.

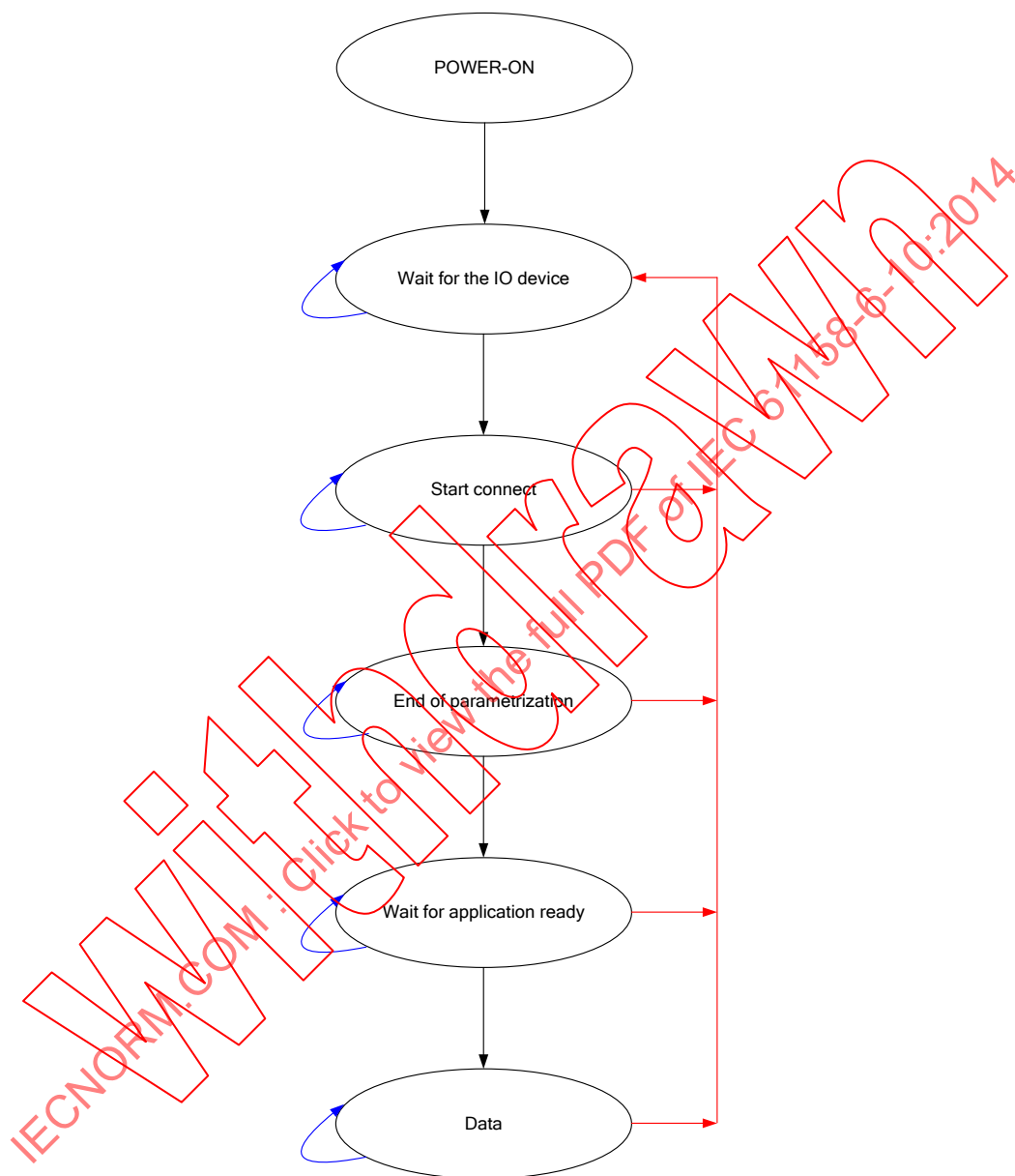


Figure 113 – State transition diagram of the IO controller CM

States of the IO controller CM

POWER-ON	Startup the IO controller and initialize all necessary resources
Wait for the IO device	Search for the expected IO device and prepare it, if necessary of the connection establishing
Start connect	Connect to the IO device by exchanging the communication parameters and the expected submodules to be exchanged in Data
End of parametrization	Convey the parameters for the expected submodules to be exchanged in Data and inform the IO device of the end of the startup parameterization
Wait for application ready	Give the IO device and the parameterized submodules time to adapt to the

Data previous conveyed parameters
Cyclic exchange data of the submodules between the IO controller and IO devices

5.6.4.2 Context Management Controller

5.6.4.2.1 Primitive definitions

5.6.4.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by CMCTL are described in the service definition and shown in Table 772.

Table 772 – Remote primitives issued or received by CMCTL

Primitive	Source	Destination	Associated parameters	Functions
RM_Ccontrol.ind	CTLRPC	CMCTL	AREP, ControlBlock, ModuleDiffBlock	—
RM_Connect.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Connect.cnf (+)	CTLRPC	CMCTL	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock, ARRPCBlockRes	—
RM_Dcontrol.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Dcontrol.cnf (+)	CTLRPC	CMCTL	AREP, ControlBlock	—
RM_Release.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Release.cnf (+)	CTLRPC	CMCTL	AREP, ControlBlock	—
CM_Connect.req	FSPMCTL	CMCTL	AREP, ARBlockReq, ListOfIOCRBlockReq, ListOfExpectedSubmodul eBlockReq, AlarmCRBlockReq, ParameterServerBlock, ListOfMulticastCRBlock, ARRPCBlock, IRInfosBlock, SRInfoBlock, ARVendorBlock	The service Connect opens an application relationship.
RM_Ccontrol.rsp (+)	CMCTL	CTLRPC	AREP, ControlBlock	—

Primitive	Source	Destination	Associated parameters	Functions
RM_Ccontrol.rsp(-)	CMCTL	CTLRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Connect.req	CMCTL	CTLRPC	AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ARRPCBlockReq	—
RM_Dcontrol.req	CMCTL	CTLRPC	AREP, ControlBlock	—
RM_Release.req	CMCTL	CTLRPC	AREP, ControlBlock	—
CM_Connect.cnf (-)	CMCTL	FSPMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	This service primitive is a negative confirmation and the requested application relationship is not established.
CM_Connect.cnf (+)	CMCTL	FSPMCTL	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock, ARRPCBlock	This service primitive is a positive confirmation to establish an application relationship.

5.6.4.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CMCTL are described in the service definition and shown in Table 773.

Table 773 – Local primitives issued or received by CMCTL

Primitive	Source	Destination	Associated parameters	Functions
CTLDINA_Discover.cnf(-)	CTLDINA	CMCTL	ErrorClass, ErrorCode	—
CTLDINA_Discover.cnf (+)	CTLDINA	CMCTL	—	—
CTLIO_info_ind	CTLIO	CMCTL	AREP	—
CMCTL_start.cnf (-)	CTLSU	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CMCTL_start.cnf (+)	CTLSU	CMCTL	AREP	—
CMCTL_PrmDone_ind	CTLWRI	CMCTL	AREP	—
CM_Abort_req	FSPMCTL	CMCTL	AREP	The service Abort terminates the AR.
CTLDINA_Discover_req ()	CMCTL	CTLDINA	AddressResolution, StationName, Dev_IP_Parameter, DCP_Parameter	—
CTLSM_start_req	CMCTL	CTLSM	AREP	—
CMCTL_start_req	CMCTL	CTLSU	AREP	—

Primitive	Source	Destination	Associated parameters	Functions
CM_Abort_cnf	CMCTL	FSPMCTL	AREP	—
CMCTL_state_ind	CMCTL	FSPMCTL CTLSU CTLWRI CTLWRR CTLPBE CTLIO CTLRDI	AREP, state {STARTUP, APPLRDY, WDATA, DATA, ABORT}	—

5.6.4.2.2 State transition diagram

The state transition diagram of the CMCTL is shown in Figure 114.

Withdrawing
IECNORM.COM : Click to view the full PDF of IEC 61158-6-10:2014

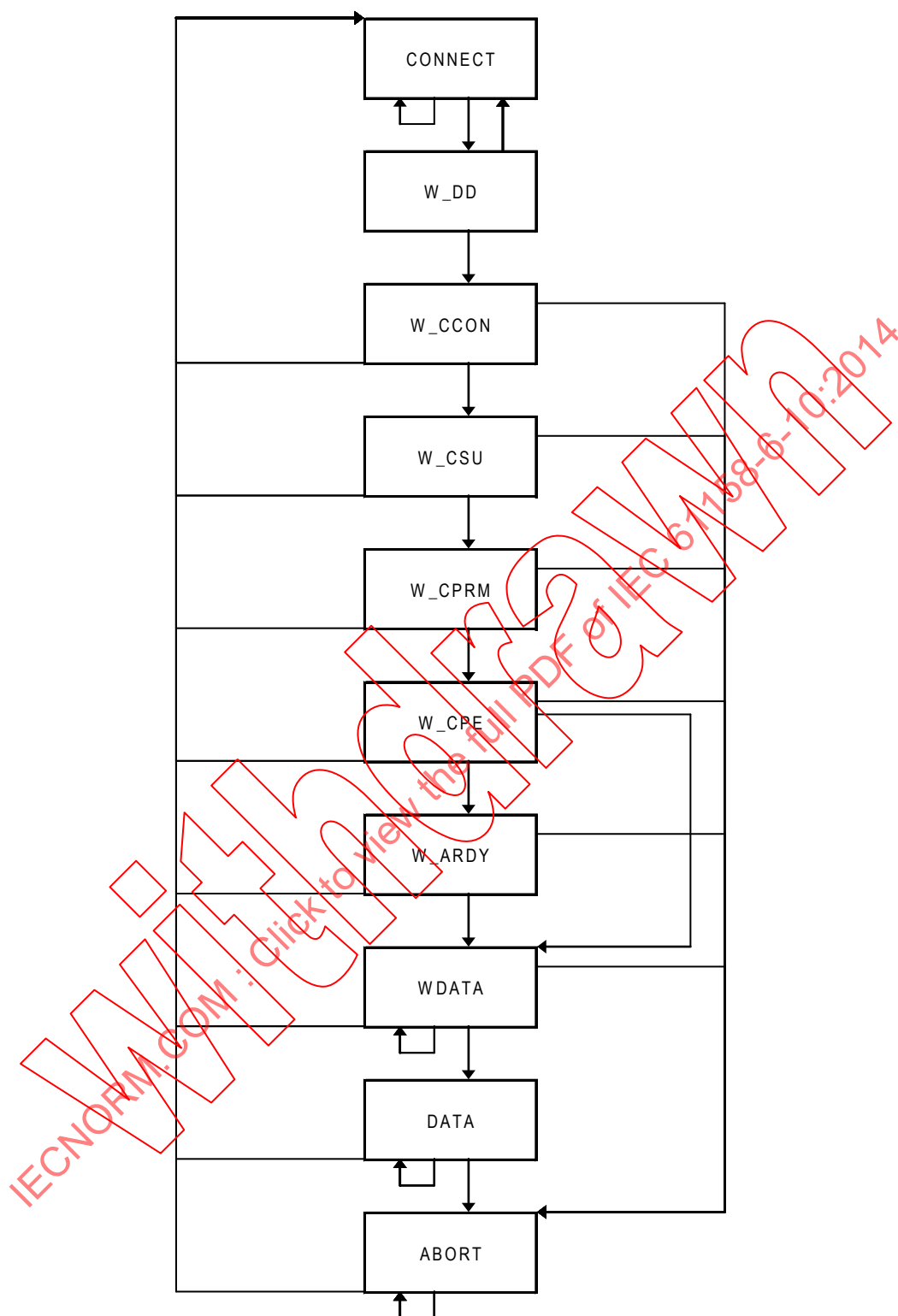


Figure 114 – State transition diagram of CMCTL

States of the CMCTL

CONNECT

Wait for a connect service from the FAL

W_DD

Wait for successful device discovery and start the AR

W_CCON

Wait for the confirmation of the connect request and start the necessary machines

W_CSU	Wait for the confirmation of the start of the necessary machines
W_CPRM	Wait for the end of the startup parameterization to issue the PrmEnd command
W_CPE	Wait for the confirmation of the PrmEnd command and change to the next state
W_ARDY	Wait for the ApplRdy command from the IO device
WDATA	Wait for data exchange using CPM and PPM and the termination of the RPC connect monitoring. The ALPMI is now able to issue an alarm.
DATA	Data exchange and connection monitoring using the CPM / PPM. The ALPMI is now able to issue an alarm.
ABORT	Do the shutdown in case of application command. An erroneous shutdown is handled by all machines concurrently initiated by the CMCTL_state_ind (ABORT).

5.6.4.2.3 State machine description

The CM Controller protocol machine exists for every AR of an IO controller. The CM Connect request service primitive may be used to address either the endpoint mapper port or the Responder RPC Server port directly. It is necessary to get the port number of the Responder RPC Server port by local means or by querying the responders endpoint mapper.

An implementation of the CMCTL shall handle the outstanding confirmations in case of an ABORT itself before a new connect service is issued.

NOTE The direct addressing of the Responder Server RPC Port is used to avoid personal firewall conflicts on PC based systems. The direct addressing of the Initiator Server RPC Port is used to support multiple engineering tools hosted on one PC.

5.6.4.2.4 CMCTL state table

Table 774 contains the complete description of the CMCTL state machine.

Table 774 – CMCTL state table

#	Current State	Event /Condition =>Action	Next State
1	CONNECT	CM_Connect.req () => Store Args CTLDINA_Discover_req ()	W_DD
2	CONNECT	CM_Abort_req () => CM_Abort_cnf ()	CONNECT
3	W_DD	CTLDINA_Discover_cnf (+) => Use stored Args RM_Connect.req ()	W_CCON
4	W_DD	CTLDINA_Discover_cnf (-) => CM_Connect.cnf (-)	CONNECT
5	W_DD	CM_Abort_req () => CM_Abort_cnf ()	CONNECT
6	W_DD	CMCTL_state_ind () /state == ABORT => ignore	CONNECT

#	Current State	Event /Condition =>Action	Next State
7	W_CCON	RM_Connect.cnf (+) => Store Result CMCTL_start_req () CTLSM_start_req ()	W_CSU
8	W_CCON	RM_Connect.cnf (-) => CM_Connect.cnf (-)	CONNECT
9	W_CCON	CM_Abort_req () => CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
10	W_CCON	CMCTL_state_ind () /state == ABORT => ignore	CONNECT
11	W_CSU	CMCTL_start.cnf (+) => LocalState := STARTUP CMCTL_state_ind (STARTUP)	W_CPRM
12	W_CSU	CMCTL_start.cnf (-) => CMCTL_state_ind (ABORT)	CONNECT
13	W_CSU	CM_Abort_req () => CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
14	W_CSU	CMCTL_state_ind () /state == ABORT => ignore	CONNECT
15	W_CPRM	CMCTL_PrmDone_ind () => Create PDU ControlBlockConnect.PrmEnd := TRUE RM_Dcontrol.req ()	W_CPE
16	W_CPRM	CM_Abort_req () => CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
17	W_CPRM	CMCTL_state_ind () /state == ABORT => ignore	CONNECT
18	W_CPE	RM_Dcontrol.cnf (+) /ControlBlockConnect.PrmEnd == TRUE => LocalState := APPLRDY StartTimer (RemoteApplicationReadyTimeout) CMCTL_state_ind (APPLRDY)	W_ARDY
19	W_CPE	RM_Dcontrol.cnf (-) /ControlBlockConnect.PrmEnd == TRUE => CMCTL_state_ind (ABORT)	CONNECT
20	W_CPE	CM_Abort_req () => CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
21	W_CPE	RM_Ccontrol.ind () /ControlCommand.ApplicationReady == TRUE => Store Result LocalState := WDATA	WDATA

#	Current State	Event /Condition =>Action	Next State
		CMCTL_state_ind (WDATA) RM_Ccontrol.rsp (+)	
22	W_CPE	CMCTL_state_ind () /state == ABORT => ignore	CONNECT
23	W_ARDY	TimerExpired (RemoteApplicationReadyTimeout) => CMCTL_state_ind (ABORT)	CONNECT
24	W_ARDY	RM_Ccontrol.ind () /ControlCommand.ApplicationReady == TRUE => Store Result LocalState := WDATA StopTimer (RemoteApplicationReadyTimeout) CMCTL_state_ind (WDATA) RM_Ccontrol.rsp (+)	WDATA
25	W_ARDY	CM_Abort_req () => StopTimer (RemoteApplicationReadyTimeout) CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
26	W_ARDY	CMCTL_state_ind () /state == ABORT => StopTimer (RemoteApplicationReadyTimeout)	CONNECT
27	WDATA	RM_Dcontrol.cnf (+) /ControlBlockConnect.PrmEnd == TRUE => ignore	WDATA
28	WDATA	RM_Dcontrol.cnf (-) /ControlBlockConnect.PrmEnd == TRUE => CMCTL_state_ind (ABORT)	CONNECT
29	WDATA	CM_Abort_req () => CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
30	WDATA	CTLIO_info_ind () /state == DATA_IMPOSSIBLE => ignore	WDATA
31	WDATA	CTLIO_info_ind () /state == DATA_POSSIBLE => LocalState := DATA CMCTL_state_ind (DATA) CM_Connect.cnf (+)	DATA
32	WDATA	CMCTL_state_ind () /state == ABORT => ignore	CONNECT
33	DATA	CMCTL_state_ind () /state == ABORT => ignore	CONNECT
34	DATA	CM_Abort_req () => CMCTL_state_ind (ABORT) RM_Release.req ()	ABORT
35	DATA	CTLIO_info_ind () => ignore	DATA

#	Current State	Event /Condition =>Action	Next State
36	DATA	RM_Connect.cnf () => ignore	DATA
37	DATA	RM_Dcontrol.cnf () => ignore	DATA
38	DATA	RM_Ccontrol.ind () => ignore	DATA
39	ABORT	RM_Release.cnf () => CM_Abort_cnf ()	CONNECT
40	ABORT	CM_Abort_req () => CM_Abort_cnf ()	ABORT
41	ABORT	RM_Connect.cnf () => ignore	ABORT
42	ABORT	RM_Dcontrol.cnf () => ignore	ABORT
43	ABORT	RM_Ccontrol.ind () => ignore	ABORT
44	ABORT	CMCTL_state_ind () => ignore	ABORT

5.6.4.2.5 Functions, Macros, Timers and Variables

Table 775 contains the functions, macros, timers and variables used by the CMCTL and their arguments and their descriptions.

Table 775 – Functions, Macros, Timers and Variables used by the CMCTL

Name	Type	Function/Meaning
StartTimer	Function	This local function is used to start or restart a timer
StopTimer	Function	This local function is used to stop a timer
TimerExpired	Function	This local function is issued if a timer expires
Create PDU	Macro	This local macro creates the corresponding PDU
Store Args	Macro	This local macros stores the parameters of the request for further use
Store Results	Macro	This local macro stores the result for a possible RPC rerun
Use stored Args	Macro	This local macro retrieve the stored parameters
RAR	Timer	This local timer is loaded with the RemoteApplicationReadyTimeout and used for the monitoring of the time between PrmEnd and ApplicationReady.
LocalState	Variable	This local variable contains the current state

5.6.4.3 Context Management Surveillance Controller

5.6.4.3.1 Primitive definitions

5.6.4.3.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by CTLSM are described in the service definition and shown in Table 776.

Table 776 – Remote primitives issued or received by CTLSM

Primitive	Source	Destination	Associated parameters	Functions
RM_Read.req	CTLSM	CTLRPC	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Read.cnf	CTLRDI	CTLSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Read.cnf (-)	CTLRPC	CTLSM	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.cnf (+)	CTLRPC	CTLSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Write.cnf	CTLWRI	CTLSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Read.req	FSPMCTL	CTLSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Write.req	FSPMCTL	CTLSM	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—

5.6.4.3.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLSM are described in the service definition and shown in Table 777.

Table 777 – Local primitives issued or received by CTLSM

Primitive	Source	Destination	Associated parameters	Functions
CTLSM_Start_req	CMCTL	CTLSM	Timeout	—
CTLIO_info_ind	CTLIO	CTLSM	info {DATA_POSSIBLE}	—
CTLSM_Start_cnf	CTLSM	CMCTL	—	—
CMCTL_state_ind	CTLSM	CMCTL	state {ABORT}	—

5.6.4.3.2 State transition diagram

The state transition diagram of the CTLSM is shown in Figure 115.

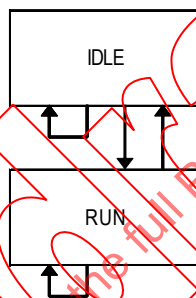


Figure 115 – State transition diagram of CTLSM

States of the CTLSM

IDLE	Wait for the start of the connection monitoring
RUN	Connection monitoring running

5.6.4.3.3 State machine description

The CM client protocol machine (CTLSM) arranges the condition monitoring between PRM_END and the PPM/CPM connection monitoring.

5.6.4.3.4 CTLSM state table

Table 778 contains the complete description of the CTLSM state machine.

Table 778 – CTLSM state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CTLSM_Start_req (CMInitiatorTriggerTimeout) => StartTimer (CMI, CMInitiatorTriggerTimeout) CTLSM_Start_cnf ()	RUN
2	IDLE	CTLIO_info_ind () => ignore	IDLE

#	Current State	Event /Condition =>Action	Next State
3	IDLE	CMCTL_state_ind () => ignore	IDLE
4	IDLE	TimeExpired (CMI) => ignore	IDLE
5	IDLE	RM_Read.cnf () /Index == Trigger => ignore	IDLE
6	RUN	TimerExpired (CMI) => RM_Read.req ()	RUN
7	RUN	CTLSM_Start_req (CMInitiatorTriggerTimeout) => ignore	RUN
8	RUN	CTLIO_info_ind () /info == DATA_POSSIBLE => StopTimer (CMI)	IDLE
9	RUN	CMCTL_state_ind () /state == ABORT => StopTimer (CMI)	IDLE
10	RUN	RM_Read.cnf (+) /Index == Trigger => StartTimer (CMI, CMInitiatorTriggerTimeout)	RUN
11	RUN	RM_Read.cnf (-) /Index == Trigger => StopTimer (CMI) CMCTL_state_ind (ABORT)	IDLE
12	RUN	CM_Read.req => StopTimer (CMI)	RUN
13	RUN	CM_Write.req => StopTimer (CMI)	RUN
14	RUN	CM_Read.cnf => StartTimer (CMI, CMInitiatorTriggerTimeout)	RUN
15	RUN	CM_Write.cnf => StartTimer (CMI, CMInitiatorTriggerTimeout)	RUN

5.6.4.3.5 Functions, Macros, Timers and Variables

Table 779 contains the functions, macros, timers and variables used by the CTLSM and their arguments and their descriptions.

Table 779 – Functions, Macros, Timers and Variables used by the CTLSM

Name	Type	Function/Meaning
TimerExpired	Function	This local function is issued if a timer expires.
StartTimer	Function	This local function starts or restarts a timer.
StopTimer	Function	This local function stops a timer.

Name	Type	Function/Meaning
CMI	Timer	This local timer is loaded with the CMInitiatorTriggerTimeout and used for the connection monitoring before the CPM / PPM monitoring is activated.

5.6.4.4 Context Management Input Output Controller

5.6.4.4.1 Primitive definitions

5.6.4.4.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by CTLIO are described in the service definition and shown in Table 780.

Table 780 – Remote primitives issued or received by CTLIO

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.4.4.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLIO are described in the service definition and shown in Table 781.

Table 781 – Local primitives issued or received by CTLIO

Primitive	Source	Destination	Associated parameters	Functions
CMCTL_state_ind	CMCTL CTLIO	CTLIO CMCTL	state {ABORT, ...}	—
CPM_NewData_ind	CPM	CMIO FSPMDEV CMDMC CTLIO	AREP, CREP, APDU_Status, data {Data, NoData}	—
CPM_State_ind	CPM	CTLIO	CREP	—
CTLIO_info_ind	CTLIO	CMCTL	info {DATA_POSSIBLE, DATA_IMPOSSIBLE}	—

5.6.4.4.2 State transition diagram

The state transition diagram of the CTLIO is shown in Figure 116.

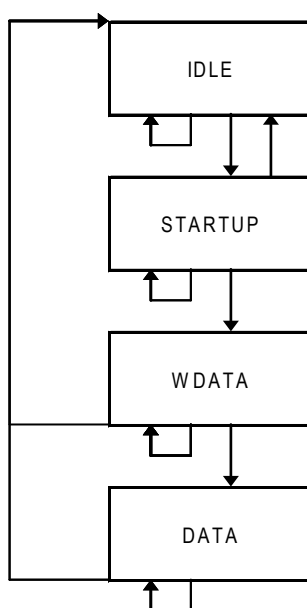


Figure 116 – State transition diagram of CTLIO

States of the CTLIO

IDLE	Wait for the start of a connection to a IO device
STARTUP	Wait for the CMCTL state PRMEND
WDATA	Wait for the established PPM / CPM data exchange
DATA	Monitor the PPM / CPM data exchange

5.6.4.4.3 State machine description

The Context Management Input Output Protocol Machine Controller (CTLIO) arranges the access to the DataStatus and NewData indication. It also handles the switchover between RPC connection monitoring and CPM / PPM connection monitoring.

5.6.4.4.4 CTLIO state table

Table 782 contains the complete description of the CTLIO state machine.

Table 782 – CTLIO state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CPM_NewData_ind () => ignore	IDLE
2	IDLE	CPM_State_ind () => ignore	IDLE
3	IDLE	CMCTL_state_ind () /state == ABORT => ignore	IDLE

#	Current State	Event /Condition =>Action	Next State
4	IDLE	CMCTL_state_ind () /state == STARTUP => For all CPMs CmInstance[CREP.CPM]:=Stop	STARTUP
5	STARTUP	CPM_NewData_ind () => ignore	STARTUP
6	STARTUP	CPM_State_ind () => ignore	STARTUP
7	STARTUP	CMCTL_state_ind () /state == ABORT => ignore	IDLE
8	STARTUP	CMCTL_state_ind () /state == APPLRDY => StartTimer ()	WDATA
9	WDATA	CPM_NewData_ind () => CmInstance[CREP.CPM]:=Start	WDATA
10	WDATA	CPM_State_ind () /state == STOP => CmInstance[CREP.CPM]:=Stop	WDATA
11	WDATA	CPM_State_ind () /state == START => CmInstance[CREP.CPM]:=Start	WDATA
12	WDATA	TimerExpired () /All CPM of CmInstance[CREP.CPM] contain the value "Start" => StartTimer () CTLIO_info_ind (DATA_POSSIBLE)	WDATA
13	WDATA	TimerExpired () /One CPM of CmInstance[CREP.CPM] contains the value "Stop" => StartTimer () CTLIO_info_ind (DATA_IMPOSSIBLE)	WDATA
14	WDATA	CMCTL_state_ind () /state == ABORT => ignore	IDLE
15	WDATA	CMCTL_state_ind () /state == WDATA => ignore	WDATA
16	WDATA	CMCTL_state_ind () /state == DATA => StopTimer ()	DATA
17	DATA	CPM_NewData_ind () => ignore	DATA
18	DATA	CPM_State_ind () /state == STOP && CREP != CREP.MCPM => CMCTL_state_ind (ABORT)	IDLE

#	Current State	Event /Condition =>Action	Next State
19	DATA	CPM_State_ind () /state == START => ignore	DATA
20	DATA	CMCTL_state_ind () /state == ABORT => ignore	IDLE

5.6.4.4.5 Functions, Macros, Timers and Variables

Table 783 contains the functions, macros, timers and variables used by the CTLIO and their arguments and their descriptions.

Table 783 – Functions, Macros, Timers and Variables used by the CTLIO

Name	Type	Function/Meaning
TimerExpired	Function	This local function is issued if a timer expires.
StartTimer	Function	This local function is used to start or restart a timer.
StopTimer	Function	This local function is used to stop a timer.
CmInstance	Variable	This AR global variable contains the information about the used CRs and their states.

5.6.4.5 Context Management Read Record Initiator Controller

5.6.4.5.1 Primitive definitions

5.6.4.5.1.1 Primitives exchanged between remote machines

Table 784 shows the service primitives including their associated parameters received by Context Management Read Record Initiator Protocol Machine Controller (CTLRDI).

Table 784 – Remote primitives received by CTLRDI

Primitive	Source	Destination	Associated parameters	Functions
CM_Read.cnf (-)	CTLRDI	FSPMCTL	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Read.cnf (+)	CTLRDI	FSPMCTL	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data	—
RM_Read.cnf (-)	CTLRPC	CTLRDI	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.cnf (+)	CTLRPC	CTLRDI	AREP,	—

Primitive	Source	Destination	Associated parameters	Functions
			SeqNumber, Length, Data	
CM_Read.req	FSPMCTL	CTLRDI	AREP, API, TargetARUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Read.req	CTLRDI	CTLRPC	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—

5.6.4.5.1.2 Primitives exchanged between local machines

Table 785 shows the service primitives including their associated parameters issued or received by the CTLRDI.

Table 785 – Local primitives issued or received by CTLRDI

Primitive	Source	Destination	Associated parameters	Functions
CMCTL_state_ind	CMCTL	FSPMCTL CTLSU CTLWRI CTLWRR CTLPBE CTLIO CTLRDI	state {ABORT, ...}	—

5.6.4.5.2 State transition diagram

The state transition diagram of the CTLRDI is shown in Figure 117.

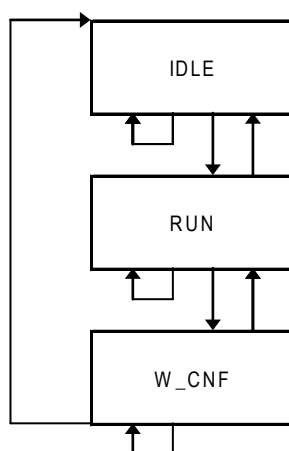


Figure 117 – State transition diagram of CTLRDI

IDLE	Idle state. Read explicit not supported.
RUN	Active state, handle read service
W_CNF	Wait for read confirmation

5.6.4.5.3 State machine description

The Context Management Read Record Initiator Protocol Machine Controller (CTLRDI) arranges the initiator side of the read record service.

5.6.4.5.4 CTLRDI state table

Table 786 contains the complete description of the CTLRDI state machine.

Table 786 – CTLRDI state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CM_Read.req () => ErrorCode:= wrong state CM_Read.cnf (-)	IDLE
2	IDLE	RM_Read.cnf () => ignore	IDLE
3	IDLE	CMCTL_state_ind () /state != ABORT => ignore	RUN
4	IDLE	CMCTL_state_ind () /state == ABORT => ignore	IDLE
5	RUN	CM_Read.req () => RM_Read.req ()	W_CNF
6	RUN	CMCTL_state_ind () /state == ABORT => ignore	IDLE
7	RUN	CMCTL_state_ind () /state != ABORT => ignore	RUN
8	W_CNF	RM_Read.cnf (+) => CM_Read.cnf (+)	RUN
9	W_CNF	RM_Read.cnf (-) => CM_Read.cnf (-)	RUN
10	W_CNF	CMCTL_state_ind () /state == ABORT => ignore	IDLE
11	W_CNF	CMCTL_state_ind () /state != ABORT => ignore	W_CNF

5.6.4.5.5 Functions, Macros, Timers and Variables

Table 787 contains the functions, macros, timers and variables used by the CTLRDI and their arguments and their descriptions.

Table 787 – Functions, Macros, Timers and Variables used by CTLRDI

Name	Type	Function/Meaning
—	—	—

5.6.4.6 Context Management Read Record Responder Controller

5.6.4.6.1 Primitive definitions

5.6.4.6.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Read Record Responder Protocol Machine Controller (CTLRDR) are described in the service definition and shown in Table 788.

Table 788 – Remote Primitives received by CTLRDR

Primitive	Source	Destination	Associated parameters	Functions
CM_Read.ind	CTLRDR	FSPMCTL	AREP, API, TargetARUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
CM_Read.rsp (-)	FSPMCTL	CTLRDR	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Read.rsp (+)	FSPMCTL	CTLRDR	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data	—
RM_Read.ind	CTLRPC	CTLRDR	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Read.rsp (-)	CTLRDR	CTLRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.rsp (+)	CTLRDR	CTLRPC	AREP, SeqNumber, Length, Data	—

5.6.4.6.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLRDR are described in the service definition and shown in Table 789.

Table 789 – Local primitives issued or received by CTLRDR

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.4.6.2 State transition diagram

The state transition diagram of the CTLRDR is shown in Figure 118.

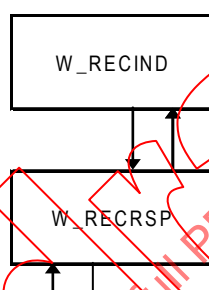


Figure 118 – State transition diagram of CTLRDR

States of the CTLRDR

W_RECIND	Wait for read record indication
W_RECRSP	Wait for read record response

5.6.4.6.3 State machine description

The CM Read Record protocol machine (CTLRDR) arranges the read record service.

5.6.4.6.4 CTLRDR state table

Table 790 contains the complete description of the CTLRDR state machine.

Table 790 – CTLRDR state table

#	Current State	Event /Condition =>Action	Next State
1	W_RECIND	RM_Read.ind () => CM_Read.ind ()	W_RECRSP
2	W_RECRSP	CM_Read.rsp (+) => RM_Read.rsp (+)	W_RECIND
3	W_RECRSP	CM_Read.rsp (-) => RM_Read.rsp (-)	W_RECIND

#	Current State	Event /Condition =>Action	Next State
4	W_RECRSP	RM_Read.ind () => ignore	W_RECRSP

5.6.4.6.5 Functions, Macros, Timers and Variables

Table 791 contains the functions, macros, timers and variables used by the CTLRDR and their arguments and their descriptions.

Table 791 – Functions, Macros, Timers and Variables used by CTLRDR

Name	Type	Function/Meaning
—	—	—

5.6.4.7 Context Management RPC Controller

5.6.4.7.1 Primitive definitions

5.6.4.7.1.1 Primitives exchanged between remote machines

Table 792 shows the service primitives including their associated parameters received by Context Management RPC Protocol Machine Controller (CTLRPC).

Table 792 – Remote primitives received by CTLRPC

Primitive	Source	Destination	Associated parameters	Functions
RM_Ccontrol.rsp (-)	CMCTL	CTLRPC	AREP, ControlBlock	—
RM_Ccontrol.rsp (+)	CMCTL	CTLRPC	AREP, ControlBlock, ModuleDiffBlock	—
RM_Connect.req	CMCTL	CTLRPC	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock	—
RM_Dcontrol.req	CMCTL	CTLRPC	AREP, ControlBlock	—
RM_Read.req	CMCTL	CTLRPC	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Read.rsp (-)	CMCTL	CTLRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.rsp (+)	CMCTL	CTLRPC	AREP, SeqNumber, Length, Data	—
RM_Release.req	CMCTL	CTLRPC	AREP, ControlBlock	—

Primitive	Source	Destination	Associated parameters	Functions
RM_Release.rsp (-)	CMCTL	CTLRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Release.rsp (+)	CMCTL	CTLRPC	AREP, ControlBlock	—
RM_Write.req	CMCTL	CTLRPC	AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data	—
RM_Write.rsp (-)	CMCTL	CTLRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Write.rsp (+)	CMCTL	CTLRPC	AREP, SeqNumber	—
RM_Ccontrol.ind	CTLRPC	CMCTL	AREP, ControlBlock, ModuleDiffBlock	—
RM_Connect.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Connect.cnf (+)	CTLRPC	CMCTL	AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock, ARRPCBlockRes	—
RM_Dcontrol.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Dcontrol.cnf (+)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Read.cnf (+)	CTLRPC	CMCTL	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—

Primitive	Source	Destination	Associated parameters	Functions
RM_Read.ind	CTLRPC	CMCTL	AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length	—
RM_Release.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Release.cnf (+)	CTLRPC	CMCTL	AREP, ControlBlock	—
RM_Release.ind	CTLRPC	CMCTL	AREP, ControlBlock	—
RM_Write.cnf (-)	CTLRPC	CMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Write.cnf (+)	CTLRPC	CMCTL	AREP, SeqNumber	—
RM_Write.ind	CTLRPC	CMCTL	AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data	—
RPC_Ccontrol.rsp (-)	CTLRPC	RPC	Arg	—
RPC_Ccontrol.rsp (+)	CTLRPC	RPC	Arg	—
RPC_Connect.req	CTLRPC	RPC	Arg	—
RPC_Dcontrol.req	CTLRPC	RPC	Arg	—
RPC_Read.req	CTLRPC	RPC	Arg	—
RPC_Read.rsp (-)	CTLRPC	RPC	Arg	—
RPC_Read.rsp (+)	CTLRPC	RPC	Arg	—
RPC_Release.req	CTLRPC	RPC	Arg	—
RPC_Release.rsp (-)	CTLRPC	RPC	Arg	—
RPC_Release.rsp (+)	CTLRPC	RPC	Arg	—
RPC_Write.req	CTLRPC	RPC	Arg	—
RPC_Write.rsp (-)	CTLRPC	RPC	Arg	—
RPC_Write.rsp (+)	CTLRPC	RPC	Arg	—
RPC_Ccontrol.ind	RPC	CTLRPC	Arg	—
RPC_Connect.cnf (-)	RPC	CTLRPC	Arg	—
RPC_Connect.cnf (+)	RPC	CTLRPC	Arg	—
RPC_Dcontrol.cnf (-)	RPC	CTLRPC	Arg	—
RPC_Dcontrol.cnf (+)	RPC	CTLRPC	Arg	—
RPC_Read.cnf (-)	RPC	CTLRPC	Arg	—
RPC_Read.cnf (+)	RPC	CTLRPC	Arg	—

Primitive	Source	Destination	Associated parameters	Functions
RPC_Read.ind	RPC	CTLRPC	Arg	—
RPC_Release.cnf (-)	RPC	CTLRPC	Arg	—
RPC_Release.cnf (+)	RPC	CTLRPC	Arg	—
RPC_Release.ind	RPC	CTLRPC	Arg	—
RPC_Write.cnf (-)	RPC	CTLRPC	Arg	—
RPC_Write.cnf (+)	RPC	CTLRPC	Arg	—
RPC_Write.ind	RPC	CTLRPC	Arg	—

5.6.4.7.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLRPC are described in the service definition and shown in Table 793.

Table 793 – Local primitives issued or received by CTLRPC

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.4.7.2 State transition diagram

The state transition diagram of the CTLRPC is shown in Figure 119.

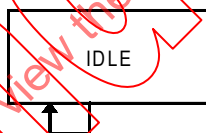


Figure 119 – State transition diagram of CTLRPC

States of the CTLRPC

IDLE Wait for an RPC or an RM call, check its integrity and for idempotent rerun

5.6.4.7.3 State machine description

The Context Management RPC Controller Protocol Machine (CTLRPC) arranges the RPC communication and handles all issued and received RPC services.

5.6.4.7.4 CTLRPC state table

Table 794 contains the complete description of the CTLRPC state machine.

Table 794 – CTRLRPC state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	RPC_Ccontrol.ind () /CheckRPC () == Rerun => Retrieve stored response encode RPC PDU RPC_Ccontrol.rsp ()	IDLE
2	IDLE	RPC_Ccontrol.ind () /CheckRPC () == Valid => AREP := LocateAR () RM_Ccontrol.ind ()	IDLE
3	IDLE	RPC_Ccontrol.ind () /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode Error PDU RPC_Ccontrol.rsp (-)	IDLE
4	IDLE	RM_Ccontrol.rsp (-) => encode Error PDU Store response RPC_Ccontrol.rsp (-)	IDLE
5	IDLE	RM_Ccontrol.rsp (+) => encode RPC PDU Store response RPC_Ccontrol.rsp (+)	IDLE
6	IDLE	RM_Dcontrol.req () => encode RPC PDU RPC_Dcontrol.req ()	IDLE
7	IDLE	RPC_Dcontrol.cnf (-) => RM_Dcontrol.cnf (-)	IDLE
8	IDLE	RPC_Dcontrol.cnf (+) /CheckRPC () == Valid => AREP := LocateAR () RM_Dcontrol.cnf (+)	IDLE
9	IDLE	RPC_Dcontrol.cnf (+) /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode Error PDU RM_Dcontrol.cnf (-)	IDLE
10	IDLE	RM_Write.rsp (-) => encode Error PDU RPC_Write.rsp (-)	IDLE
11	IDLE	RM_Write.rsp (+) => encode RPC PDU RPC_Write.rsp (+)	IDLE
12	IDLE	RM_Write.req () => encode RPC PDU RPC_Write.req ()	IDLE
13	IDLE	RPC_Write.ind () => AREP := LocateAR () RM_Write.ind ()	IDLE

#	Current State	Event /Condition =>Action	Next State
14	IDLE	RPC_Write.cnf (-) => RM_Write.cnf (-)	IDLE
15	IDLE	RPC_Write.cnf (+) /CheckRPC () == Valid => AREP := LocateAR () RM_Write.cnf (+)	IDLE
16	IDLE	RPC_Write.cnf (+) /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode Error PDU RM_Write.cnf (-)	IDLE
17	IDLE	RM_Read.rsp (-) => encode Error PDU RPC_Read.rsp (-)	IDLE
18	IDLE	RM_Read.rsp (+) => encode RPC PDU RPC_Read.rsp (+)	IDLE
19	IDLE	RM_Read.req () => encode RPC PDU RPC_Read.req ()	IDLE
20	IDLE	RPC_Read.ind () => AREP := LocateAR () RM_Read.ind ()	IDLE
21	IDLE	RPC_Read.cnf (-) => RM_Read.cnf (-)	IDLE
22	IDLE	RPC_Read.cnf (+) /CheckRPC () == Valid => AREP := LocateAR () RM_Read.cnf (+)	IDLE
23	IDLE	RPC_Read.cnf (+) /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode Error PDU RM_Read.cnf (-)	IDLE
24	IDLE	RM_Connect.req () => encode RPC PDU RPC_Connect.req ()	IDLE
25	IDLE	RPC_Connect.cnf (-) => RM_Connect.cnf (-)	IDLE
26	IDLE	RPC_Connect.cnf (+) /CheckRPC () == Valid //NOTE The CMResponderName shall be checked to avoid wrong connections => AREP := LocateAR () RM_Connect.cnf (+)	IDLE
27	IDLE	RPC_Connect.cnf (+) /CheckRPC () == InValid, ErrorDecode, ErrorCode1, ErrorCode2 => encode Error PDU RM_Connect.cnf (-)	IDLE

#	Current State	Event /Condition =>Action	Next State
28	IDLE	RM_Release.req () => encode RPC PDU RPC_Release.req ()	IDLE
29	IDLE	RPC_Release.cnf () => RM_Release.cnf ()	IDLE
30	IDLE	RPC_Release.ind () => encode Error PDU RPC_Release.rsp (-)	IDLE
31	IDLE	RPC_Connect.ind () => encode Error PDU RPC_Connect.rsp (-)	IDLE

5.6.4.7.5 Functions, Macros, Timers and Variables

Table 795 contains the functions, macros, timers and variables used by the CTLRPC and their arguments and their descriptions.

Table 795 – Functions, Macros, Timers and Variables used by the CTLRPC

Name	Type	Function/Meaning
CheckRPC	Function	This local functions checks the block structure and parameters against the definitions of this standard.
Encode Error PDU	Macro	This local macro generates the appropriate PDU with the detected errors from the PDU checking rules. An error PDU consists only out of the RPC NDR header.
Encode RPC PDU	Macro	This local macro generates the appropriate PDU.
LocateAR	Macro	This local macro retrieves the associated AREP.
Retrieve stored response	Macro	This local macro retrieves the previously stored parameter in case of a detected RPC rerun.
Store response	Macro	This local macro stores the parameters of the RPC response to be retrievable in case of a RPC rerun
AREP	Variable	This local variable stores the application relation end point for further use

5.6.4.8 Context Management Startup Controller

5.6.4.8.1 Primitive definitions

5.6.4.8.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Startup Protocol Machine Controller (CTLSU) are described in the service definition and shown in Table 796.

Table 796 – Remote primitives issued or received by CTLSU

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

5.6.4.8.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLSU are described in the service definition and shown in Table 797.

Table 797 – Local Primitives issued or received by CTLSU

Primitive	Source	Destination	Associated parameters	Functions
ALPMI_Activate_req	CTLSU	ALPMI	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	—
ALPMI_Close_req	CTLSU	ALPMI	CREP	—
APMR_Activate_req	CTLSU	APMR	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	—
APMR_Close_req	CTLSU	APMR	CREP	—
APMS_Activate_req	CTLSU	APMS	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	—
APMS_Close_req	CTLSU	APMS	CREP	—
CMCTL_start_cnf (-)	CTLSU	CMCTL	—	—
CMCTL_start_cnf (+)	CTLSU	CMCTL	—	—
CPM_Activate_req	CTLSU	CPM	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	—
CPM_Close_req	CTLSU	CPM	CREP	—
ALPMI_Activate_cnf (-)	ALPMI	CTLSU	—	—
ALPMI_Error_ind	ALPMI	CTLSU	ErrorCode1, ErrorCode2	—
ALPMR_Activate_cnf (-)	ALPMR	CTLSU	—	—
ALPMR_Error_ind	ALPMR	CTLSU	—	—
APMR_Error_ind	APMR	CTLSU	ErrorCode1, ErrorCode2	—
APMS_Error_ind	APMS	CTLSU	ErrorCode1, ErrorCode2	—
CMCTL_start_req	CMCTL	CTLSU	AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListofMCRBlockReq, ListofSubFrameBlockReq, ARBlockRes, ListofIOCRBlockRes, AlarmCRBlockRes	—
CPM_Activate_cnf (-)	CPM	CTLSU	—	—
CPM_Error_ind	CPM	CTLSU	ErrorCode1, ErrorCode2	—

Primitive	Source	Destination	Associated parameters	Functions
DFP_Activate_cnf (-)	DFP	CTLSU	—	—
PPM_Activate_cnf (-)	PPM	CTLSU	—	—
PPM_Error_ind	PPM	CTLSU	ErrorCode1, ErrorCode2	—
XXX_Activate_cnf (+)	xxx	CTLSU	CREP	—
XXX_Close_cnf (+)	xxx	CTLSU	CREP	—
CMCTL_state_ind	CMCTL	FSPMCTL CTLSU CTLWRI CTLWRR CTLPBE CTLIO CTLRDI	state {ABORT}	—
PPM_Activate_req	CTLSU	PPM	CREP, DA, SA, VLAN, RTATimeoutFactor, RTARetries	—
PPM_Close_req	CTLSU	PPM	CREP	—

5.6.4.8.2 State transition diagram

The state transition diagram of the CTLSU is shown in Figure 120.

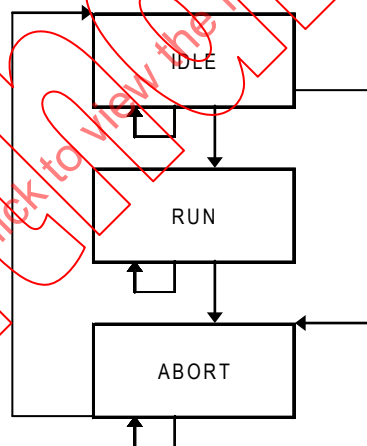


Figure 120 – State transition diagram of CTLSU

States of the CTLSU

IDLE

No AR context exists

RUN

The lower machines are started or bound to the AR. This state is kept during startup and data exchange.

ABORT

Termination sequence

5.6.4.8.3 State machine description

The IO controller startup protocol machine (CTLSU) arranges the start of the underlying protocol machines.

The startup of the CPM and PPM state machines shall be done between receiving the connect confirmation service and the Parameterization end confirmation service. The startup shall be finish before issuing the Application ready response service.

5.6.4.8.4 CTLSU state table

Table 798 contains the complete description of the CTLSU state machine.

Table 798 – CTLSU state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CMCTL_start_req () => CREP.PPM := create (PPM) CREP.CPM := create (CPM) CREP.ALPMI := create (ALPMI) CREP.ALPMR := create (ALPMR) CREP.APMS := create (APMS) CREP.APMR := create (APMR) //NOTE Create high and low instance of the Alarm machines PPM_Activate_req () CPM_Activate_req () ALPMI_Activate_req () ALPMR_Activate_req () APMS_Activate_req () APMR_Activate_req ()	IDLE
2	IDLE	XXX_Activate_cnf (+) /if last confirmation is received => CMCTL_start_cnf (+)	IDLE
3	IDLE	PPM_Activate_cnf (-) => ErrorCode2 = AR add-provider or consumer failed CMCTL_start_cnf (-)	IDLE
4	IDLE	CPM_Activate_cnf (-) => ErrorCode2 = AR add-provider or consumer failed CMCTL_start_cnf (-)	IDLE
5	IDLE	ALPMI_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMCTL_start_cnf (-)	IDLE
6	IDLE	ALPMR_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMCTL_start_cnf (-)	IDLE
7	IDLE	APMS_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMCTL_start_cnf (-)	IDLE
8	IDLE	APMR_Activate_cnf (-) => ErrorCode2 = AR alarm-open failed CMCTL_start_cnf (-)	IDLE
9	IDLE	CMCTL_state_ind () /state == !ABORT => ignore	RUN

#	Current State	Event /Condition =>Action	Next State
10	IDLE	CMCTL_state_ind () /state == ABORT => PPM_Close_req () CPM_Close_req () ALPMI_Close_req () ALPMR_Close_req () APMS_Close_req () APMR_Close_req ()	ABORT
11	IDLE	PPM_Error_ind () => CMCTL_state_ind (ABORT)	IDLE
12	IDLE	CPM_Error_ind () => CMCTL_state_ind (ABORT)	IDLE
13	IDLE	ALPMI_Error_ind () => CMCTL_state_ind (ABORT)	IDLE
14	IDLE	ALPMR_Error_ind () => CMCTL_state_ind (ABORT)	IDLE
15	IDLE	APMS_Error_ind () => CMCTL_state_ind (ABORT)	IDLE
16	IDLE	APMR_Error_ind () => CMCTL_state_ind (ABORT)	IDLE
17	RUN	CMCTL_state_ind () /state == !ABORT => ignore	RUN
18	RUN	CMCTL_state_ind () /state == ABORT => PPM_Close_req () CPM_Close_req () ALPMI_Close_req () ALPMR_Close_req () APMS_Close_req () APMR_Close_req ()	ABORT
19	RUN	CMCTL_start_req () => ErrorCode1 := CMCTL ErrorCode2 := state conflict CMCTL_start_cnf (-)	RUN
20	RUN	PPM_Error_ind () => CMCTL_state_ind (ABORT)	RUN
21	RUN	CPM_Error_ind () => CMCTL_state_ind (ABORT)	RUN
22	RUN	ALPMI_Error_ind () => CMCTL_state_ind (ABORT)	RUN
23	RUN	ALPMR_Error_ind () => CMCTL_state_ind (ABORT)	RUN

#	Current State	Event /Condition =>Action	Next State
24	RUN	APMS_Error_ind () => CMCTL_state_ind (ABORT)	RUN
25	RUN	APMR_Error_ind () => CMCTL_state_ind (ABORT)	RUN
26	ABORT	XXX_Close_cnf (+) /if last confirmation is received => Delete (PPM) Delete (CPM) Delete (ALPMI) Delete (ALPMR) Delete (APMS) Delete (APMR)	IDLE
27	ABORT	PPM_Error_ind () => ignore	ABORT
28	ABORT	CPM_Error_ind () => ignore	ABORT
29	ABORT	ALPMI_Error_ind () => ignore	ABORT
30	ABORT	ALPMR_Error_ind () => ignore	ABORT
31	ABORT	APMS_Error_ind () => ignore	ABORT
32	ABORT	APMR_Error_ind () => ignore	ABORT

5.6.4.8.5 Functions, Macros, Timers and Variables

Table 799 contains the functions, macros, timers and variables used by the CTLSU and their arguments and their descriptions.

Table 799 – Functions, Macros, Timers and Variables used by the CTLSU

Name	Type	Function/Meaning
Delete	Function	This local function frees or unbounds the machines from the CMCTL instance if the AR is aborted.
If last confirmation is received	Macro	This local macro collects all the confirmations and returns TRUE if all receive with xxx_cnf (+).

5.6.4.9 Context Management Write Record Initiator Controller

5.6.4.9.1 Primitive definitions

5.6.4.9.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Write Record Initiator Protocol Machine Controller (CTLWRI) are described in the service definition and shown in Table 800.

Table 800 – Remote primitives issued or received by CTLWRI

Primitive	Source	Destination	Associated parameters	Functions
CM_Write.cnf (-)	CTLWRI	FSPMCTL	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Write.cnf (+)	CTLWRI	FSPMCTL	AREP, SeqNumber	—
RM_Write.req	CTLWRI	CTLRPC	AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data	—
CM_Write.req	FSPMCTL	CTLWRI	AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data	—
RM_Write.cnf (-)	CTLRPC	CTLWRI	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Write.cnf (+)	CTLRPC	CTLWRI	AREP, SeqNumber	—

5.6.4.9.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLWRI are described in the service definition and shown in Table 801.

Table 801 – Local primitives issued or received by CTLWRI

Primitive	Source	Destination	Associated parameters	Functions
CMCTL_state_ind	CMCTL	CTLWRI	state {ABORT, ...}	—
CMCTL_PrmDone_ind	CTLWRI	CMCTL	—	—

5.6.4.9.2 State transition diagram

The state transition diagram of the CTLWRI is shown in Figure 121.

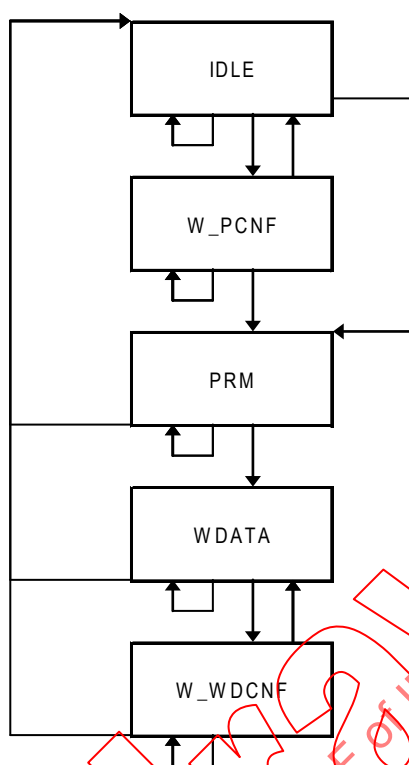


Figure 121 – State transition diagram of CTLWRI

States of the CTLWRI

IDLE	No AR context exists
PRM	Do the startup parameterization.
W_PCNF	Wait for the write record confirmation during startup parameterization.
WDATA	An application process may issue a write record request after the CMCTL enters the WDATA state. This is monitored by the CTLWRI.
W_WDCNF	After a write record is conveyed, the CTLWRI waits for the confirmation.

5.6.4.9.3 State machine description

The Context Management Write Record Initiator Protocol Machine Controller (CTLWRI) arranges initiator side of the read record service.

5.6.4.9.4 CTLWRI state table

Table 802 contains the complete description of the CTLWRI state machine.

Table 802 – CTLWRI state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CM_Write.req () => ErrorCode:= wrong state CM_Write.cnf (-)	IDLE
2	IDLE	RM_Write.cnf () => ignore	IDLE
3	IDLE	CMCTL_state_ind () /state == STARTUP && List of Startup Records is empty => ignore CMCTL_PrmDone_ind ()	PRM
4	IDLE	CMCTL_state_ind () /state == STARTUP && ! List of Startup Records is empty => Get first entry RM_Write.req ()	W_PCNF
5	W_PCNF	RM_Write.cnf () /! List of Startup Records is empty => Get next entry RM_Write.req ()	W_PCNF
6	W_PCNF	RM_Write.cnf () /List of Startup Records is empty => CMCTL_PrmDone_ind ()	PRM
7	W_PCNF	CM_Write.req () => ErrorCode:= wrong state CM_Write.cnf (-)	W_PCNF
8	W_PCNF	CMCTL_state_ind () /state == ABORT => ignore	IDLE
9	W_PCNF	CMCTL_state_ind () /state != ABORT => ignore	W_PCNF
10	PRM	CM_Write.req () => ErrorCode:= wrong state CM_Write.cnf (-)	PRM
11	PRM	CMCTL_state_ind () /state == ABORT => ignore	IDLE
12	PRM	CMCTL_state_ind () /state == APPLRDY state == STARTUP => ignore	PRM
13	PRM	CMCTL_state_ind () /state == WDATA state == DATA => ignore	WDATA

#	Current State	Event /Condition =>Action	Next State
14	WDATA	CM_Write.req () => RM_Write.req ()	W_WDCNF
15	WDATA	CMCTL_state_ind () /state == ABORT => ignore	IDLE
16	WDATA	CMCTL_state_ind () /state != ABORT => ignore	WDATA
17	W_WDCNF	RM_Write.cnf (+) => CM_Write.cnf (+)	WDATA
18	W_WDCNF	RM_Write.cnf (-) => CM_Write.cnf (-)	WDATA
19	W_WDCNF	CM_Write.req () => ErrorCode:= wrong state CM_Write.cnf (-)	W_WDCNF
20	W_WDCNF	CMCTL_state_ind () /state == ABORT => ignore	IDLE
21	W_WDCNF	CMCTL_state_ind () /state != ABORT => ignore	W_WDCNF

5.6.4.9.5 Functions, Macros, Timers and Variables

Table 803 contains the functions, macros, timers and variables used by the CTLWRI and their arguments and their descriptions.

Table 803 – Functions, Macros, Timers and Variables used by CTLWRI

Name	Type	Function/Meaning
—	—	—

5.6.4.10 Context Management Write Record Responder Controller

5.6.4.10.1 Primitive definitions

5.6.4.10.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Write Record Responder Protocol Machine Controller (CTLWRR) are described in the service definition and shown in Table 804.

Table 804 – Remote primitives issued or received by CTLWRR

Primitive	Source	Destination	Associated parameters	Functions
CM_Write.ind	CTLWRR	FSPMCTL	AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data	—
RM_Write.rsp (-)	CTLWRR	CTLRPC	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
RM_Write.rsp (+)	CTLWRR	CTLRPC	AREP, SeqNumber	—
CM_Write.rsp (-)	FSPMCTL	CTLWRR	AREP, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	—
CM_Write.rsp (+)	FSPMCTL	CTLWRR	AREP, SeqNumber	—
RM_Write.ind	CTLRPC	CTLWRR	AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data	—

5.6.4.10.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLWRR are described in the service definition and shown in Table 805.

Table 805 – Local primitives issued or received by CTLWRR

Primitive	Source	Destination	Associated parameters	Functions
CMCTL_state_ind	CMCTL	CTLWRR	state {ABORT, ...}	—

5.6.4.10.2 State transition diagram

The state transition diagram of the CTLWRR is shown in Figure 122.

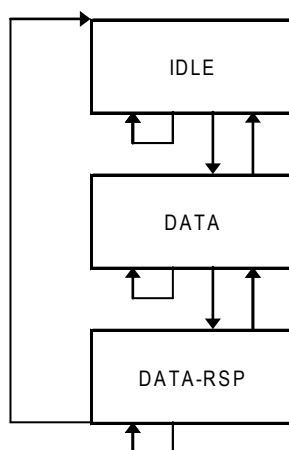


Figure 122 – State transition diagram of CTLWRR

States of the CTLWRR

IDLE	No AR context exists
DATA	The associated AR reaches an internal state in which write services are handled.
DATA-RSP	Wait for the user response and convey it to the requester.

5.6.4.10.3 State machine description

The Context Management Write Record Responder Protocol Machine Controller (CTLWRR) arranges the read record service.

5.6.4.10.4 CTLWRR state table

Table 806 contains the complete description of the CTLWRR state machine.

Table 806 – CTLWRR state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	RM_Write.ind () => ErrorDecode := PNORW ErrorCode1 := state conflict RM_Write.rsp (-)	IDLE
2	IDLE	CM_Write.rsp () => ignore	IDLE
3	IDLE	CMCTL_state_ind () /state == STARTUP =>	DATA
4	IDLE	CMCTL_state_ind () /state != STARTUP =>	IDLE
5	DATA	CMCTL_state_ind () /state != ABORT => ignore	DATA

#	Current State	Event /Condition =>Action	Next State
6	DATA	CMCTL_state_ind () /state == ABORT => ignore	IDLE
7	DATA	RM_Write.ind () => CM_Write.ind ()	DATA-RSP
8	DATA-RSP	CM_Write.rsp (+) => RM_Write.rsp (+)	DATA
9	DATA-RSP	CM_Write.rsp (-) => RM_Write.rsp (-)	DATA
10	DATA-RSP	RM_Write.ind () => ignore	DATA-RSP
11	DATA-RSP	CMCTL_state_ind () /state != ABORT => ignore	DATA-RSP
12	DATA-RSP	CMCTL_state_ind () /state == ABORT => ignore	IDLE

5.6.4.10.5 Functions, Macros, Timers and Variables

Table 807 contains the functions, macros, timers and variables used by the CTLWRR and their arguments and their descriptions.

Table 807 – Functions, Macros, Timers and Variables used by CTLWRR

Name	Type	Function/Meaning
—	—	—

5.6.4.11 Context Management Prm Begin End Controller

5.6.4.11.1 Primitive definitions

5.6.4.11.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Prm Begin End Controller (CTLPBE) are described in the service definition and shown in Table 808.

Table 808 – Remote primitives issued or received by CTLPBE

Primitive	Source	Destination	Associated parameters	Functions
RM_Ccontrol.ind	CTLRPC	CTLPBE	AREP, ControlBlock, ModuleDiffBlock	ApplRdy
RM_Dcontrol.cnf (-)	CTLRPC	CTLPBE	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	PrmBegin or PrmEnd

Primitive	Source	Destination	Associated parameters	Functions
RM_Dcontrol.cnf (+)	CTLRPC	CTLPBE	AREP, ControlBlock	PrmBegin or PrmEnd
CM_Ccontrol.rsp (-)	FSPMCTL	CTLPBE	AREP, ControlBlock	ApplRdy
CM_Ccontrol.rsp (+)	FSPMCTL	CTLPBE	AREP, ControlBlock	ApplRdy
CM_Dcontrol.req	FSPMCTL	CTLPBE	AREP, ControlBlock, ListOfSubmodules	PrmBegin or PrmEnd
RM_Ccontrol.rsp (-)	CTLBE	CTLRPC	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	ApplRdy
RM_Ccontrol.rsp (+)	CTLBE	CTLRPC	AREP, ControlBlock	ApplRdy
RM_Dcontrol.req	CTLBE	CTLRPC	AREP, ControlBlock	PrmBegin or PrmEnd
CM_Ccontrol.ind	CTLBE	FSPMCTL	AREP, ControlBlock, ModuleDiffBlock	ApplRdy
CM_Dcontrol.cnf (-)	CTLBE	FSPMCTL	AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2	PrmBegin or PrmEnd
CM_Dcontrol.cnf (+)	CTLBE	FSPMCTL	AREP, ControlBlock, ModuleDiffBlock	PrmBegin or PrmEnd

5.6.4.11.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLPBE are described in the service definition and shown in Table 809.

Table 809 – Local primitives issued or received by CTLPBE

Primitive	Source	Destination	Associated parameters	Functions
CMCTL_state.ind	CMCTL CTLPBE	CTLPBE CMCTL	AREP, state {STARTUP, APPLRDY, WDATA, DATA, ABORT}	—

5.6.4.11.2 State transition diagram

The state transition diagram of the CTLPBE is shown in Figure 123.

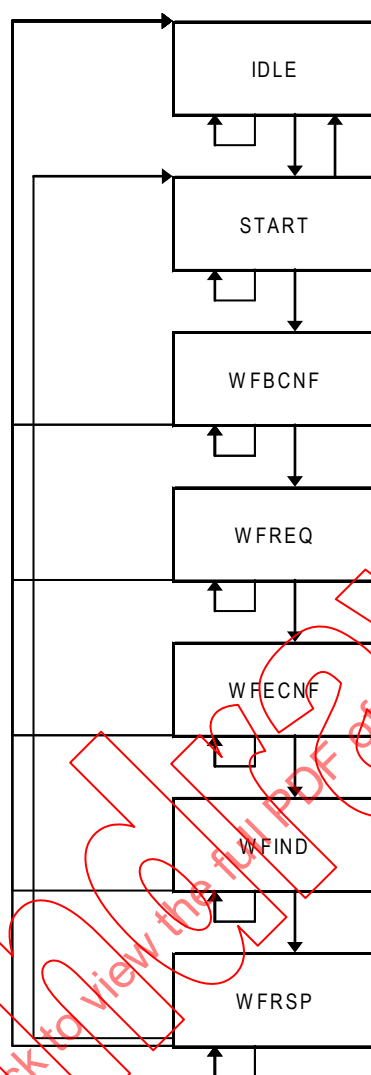


Figure 123 – State transition diagram of CTLPBE

States of the CTLPBE

IDLE	Wait for the start of DATA
START	Wait for a PrmBegin command
WFBCNF	Wait for a PrmBegin confirmation from the IO device
WFREQ	The application shall convey the records to the IO device using the CTLWRI. This machine waits for the PrmEnd command to continue.
WFECNF	Wait for a PrmEnd confirmation from the IO device
WFIND	Wait for an application ready indication from the IO device
WFRSP	Wait for an application ready confirmation from the application and convey it to the IO device. A possible RPC rerun is handled by the CTLRPC.

5.6.4.11.3 State machine description

The CTLPBE arranges a consistent parameterization for a given amount of submodules and their startup parameters in state DATA.

The IO controller shall proceed all Alarms from the involved submodules during the PrmBegin PrmEnd ApplRdy sequence, except the Plug or Release. Overlapping Plug and Release

Alarms shall be stored by the IO controller and processed after the PrmBegin PrmEnd ApplRdy sequence.

5.6.4.11.4 CTLPBE state table

Table 810 contains the complete description of the CTLPBE state machine.

Table 810 – CTLPBE state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	CMCTL_state_ind () /state == DATA => ignore	START
2	IDLE	CMCTL_state_ind () /state != DATA => ignore	IDLE
3	START	CM_Dcontrol.req () /ControlBlock == PrmBegin => RM_Dcontrol.req ()	WFBCNF
4	START	CM_Dcontrol.req () /ControlBlock != PrmBegin => ignore	START
5	START	CMCTL_state_ind () /state == ABORT => ignore	IDLE
6	WFBCNF	RM_Dcontrol.cnf (+) /ControlBlock == PrmBegin => CM_Dcontrol.cnf (+)	WFREQ
7	WFBCNF	RM_Dcontrol.cnf (-) /ControlBlock == PrmBegin => CM_Dcontrol.cnf (-) CMCTL_state_ind (ABORT)	IDLE
8	WFBCNF	CM_Dcontrol.req () => ignore CM_Dcontrol.cnf (-)	WFBCNF
9	WFBCNF	CMCTL_state_ind () /state == ABORT => ignore	IDLE
10	WFBCNF	CMCTL_state_ind () /state != ABORT => ignore	WFBCNF
11	WFREQ	CM_Dcontrol.req () /ControlBlock == PrmEnd => RM_Dcontrol.req ()	WFECNF
12	WFREQ	CM_Dcontrol.req () /ControlBlock != PrmEnd => ignore	IDLE

#	Current State	Event /Condition =>Action	Next State
13	WFREQ	CMCTL_state_ind () /state == ABORT => ignore	IDLE
14	WFREQ	CMCTL_state_ind () /state != ABORT => ignore	WFREQ
15	WFECNF	RM_Dcontrol.cnf (+) /ControlBlock == PrmEnd => CM_Dcontrol.cnf (+)	WFIND
16	WFECNF	RM_Dcontrol.cnf (-) /ControlBlock == PrmEnd => CM_Dcontrol.cnf (-) CMCTL_state_ind (ABORT)	IDLE
17	WFECNF	CM_Dcontrol.req () => ignore CM_Dcontrol.cnf (-)	WFECNF
18	WFECNF	CMCTL_state_ind () /state == ABORT => ignore	IDLE
19	WFECNF	CMCTL_state_ind () /state != ABORT => ignore	WFECNF
20	WFIND	RM_Ccontrol.ind () /ControlBlock == ApplRdy => CM_Ccontrol.ind ()	WFRSP
21	WFIND	CM_Dcontrol.req () => ignore CM_Dcontrol.cnf (-)	WFIND
22	WFIND	CMCTL_state_ind () /state == ABORT => ignore	IDLE
23	WFIND	CMCTL_state_ind () /state != ABORT => ignore	WFIND
24	WFRSP	CM_Ccontrol.rsp (+) /ControlBlock == ApplRdy => RM_Ccontrol.rsp (+)	START
25	WFRSP	CM_Ccontrol.rsp (-) /ControlBlock == ApplRdy => RM_Ccontrol.rsp (-)	IDLE
26	WFRSP	CM_Dcontrol.req () => ignore CM_Dcontrol.cnf (-)	WFRSP
27	WFRSP	CMCTL_state_ind () /state == ABORT => ignore	IDLE

#	Current State	Event /Condition =>Action	Next State
28	WFRSP	CMCTL_state_ind () /state != ABORT => ignore	WFRSP

5.6.4.11.5 Functions, Macros, Timers and Variables

Table 811 contains the functions, macros, timers and variables used by the CTLPBE and their arguments and their descriptions.

Table 811 – Functions, Macros, Timers and Variables used by CTLPBE

Name	Type	Function/Meaning
—	—	—

5.6.4.12 Context Management Discovery, IP and Name Assignment

5.6.4.12.1 Primitive definitions

5.6.4.12.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Context Management Discovery, IP and Name Assignment (CTLDINA) are described in the service definition and shown in Table 812.

Table 812 – Remote primitives issued or received by CTLDINA

Primitive	Source	Destination	Associated parameters	Functions
ARP_getMACQ.ind	ARP	CTLDINA	MAC_Address, IP_Address	—
ARP_getMAC.req	CTLDINA	ARP	IP_Address	—
DCPHMCR_Activate.req	CTLDINA	DCPHMCR	CREP, VLAN-Prio, VLAN-ID, DCPHMC_Timeout	—
DCPMCS_Activate.req	CTLDINA	DCPMCS	CREP, VLAN-Prio, VLAN-ID, DCPMC_Timeout	—
DCP_Identify.req	CTLDINA	DCPMCS	CREP, ListofFilter, ResponseDelay, DCP_Para.isASU	—
DCP_Set.req	CTLDINA	DCPUCS	CREP, DA, DataSet, Dev_IP_Para, SetCmd	—
DCPUCS_Activate.req	CTLDINA	DCPUCS	CREP, VLAN-Prio, VLAN-ID, DCPUC_Timeout	—
DNS_QueryName.req	CTLDINA	DNS	StationName.RealStationName	—
DCP_Hello.ind	DCPHMCR	CTLDINA	CREP, ListOfData	—

Primitive	Source	Destination	Associated parameters	Functions
DCPHMCR_Activate.cnf (-)	DCPHMCR	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCPHMCR_Activate.cnf (+)	DCPHMCR	CTLDINA	CREP	—
DCPHMCR_Close.cnf (-)	DCPHMCR	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCPHMCR_Close.cnf (+)	DCPHMCR	CTLDINA	CREP	—
DCP_Identify.cnf (-)	DCPMCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCP_Identify.cnf (+)	DCPMCS	CTLDINA	CREP, ListOfData	—
DCPMCS_Activate.cnf (-)	DCPMCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCPMCS_Activate.cnf (+)	DCPMCS	CTLDINA	CREP	—
DCPMCS_Close.cnf (-)	DCPMCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCPMCS_Close.cnf (+)	DCPMCS	CTLDINA	CREP	—
DCP_Set.cnf (-)	DCPUCS	CTLDINA	CREP, DA, ERRCLS, ERRCODE	—
DCP_Set.cnf (+)	DCPUCS	CTLDINA	CREP, DA, ListOfResponse	—
DCPUCS_Activate.cnf (-)	DCPUCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCPUCS_Activate.cnf (+)	DCPUCS	CTLDINA	CREP	—
DCPUCS_Close.cnf (-)	DCPUCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCPUCS_Close.cnf (+)	DCPUCS	CTLDINA	CREP	—
DNS_QueryName.cnf (-)	DNS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DNS_QueryName.cnf (+)	DNS	CTLDINA	IP_Add	—

5.6.4.12.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLDINA are described in the service definition and shown in Table 813.

Table 813 – Local primitives issued or received by CTLDINA

Primitive	Source	Destination	Associated parameters	Functions
CTLDINA_Discover_req	CMCTL	CTLDINA	Add_resolution, StationName, Dev_IP_Para, DCP_Para	This service starts the device discovery and name resolution process.

Primitive	Source	Destination	Associated parameters	Functions
CTLDINA_Discover_cnf (-)	CTLDINA	CMCTL	ErrorClass, ErrorCode	This service primitive indicates that the address and name resolution service failed. The error parameter contains the reason for the failure (e.g. no device, multiple devices)
CTLDINA_Discover_cnf (+)	CTLDINA	CMCTL	Dev_MAC_Addr	This service primitive indicates that the Activate service succeeded and delivers the MAC Address of the IO device.
DCP_Identify_ind	DCP	CTLDINA	—	—
CMCTL_state_ind	CTLDINA	CMCTL	state {ABORT}	This service shut down the AR connection establishment.

5.6.4.12.2 State transition diagram

The state transition diagram of the CTLDINA is shown in Figure 124.

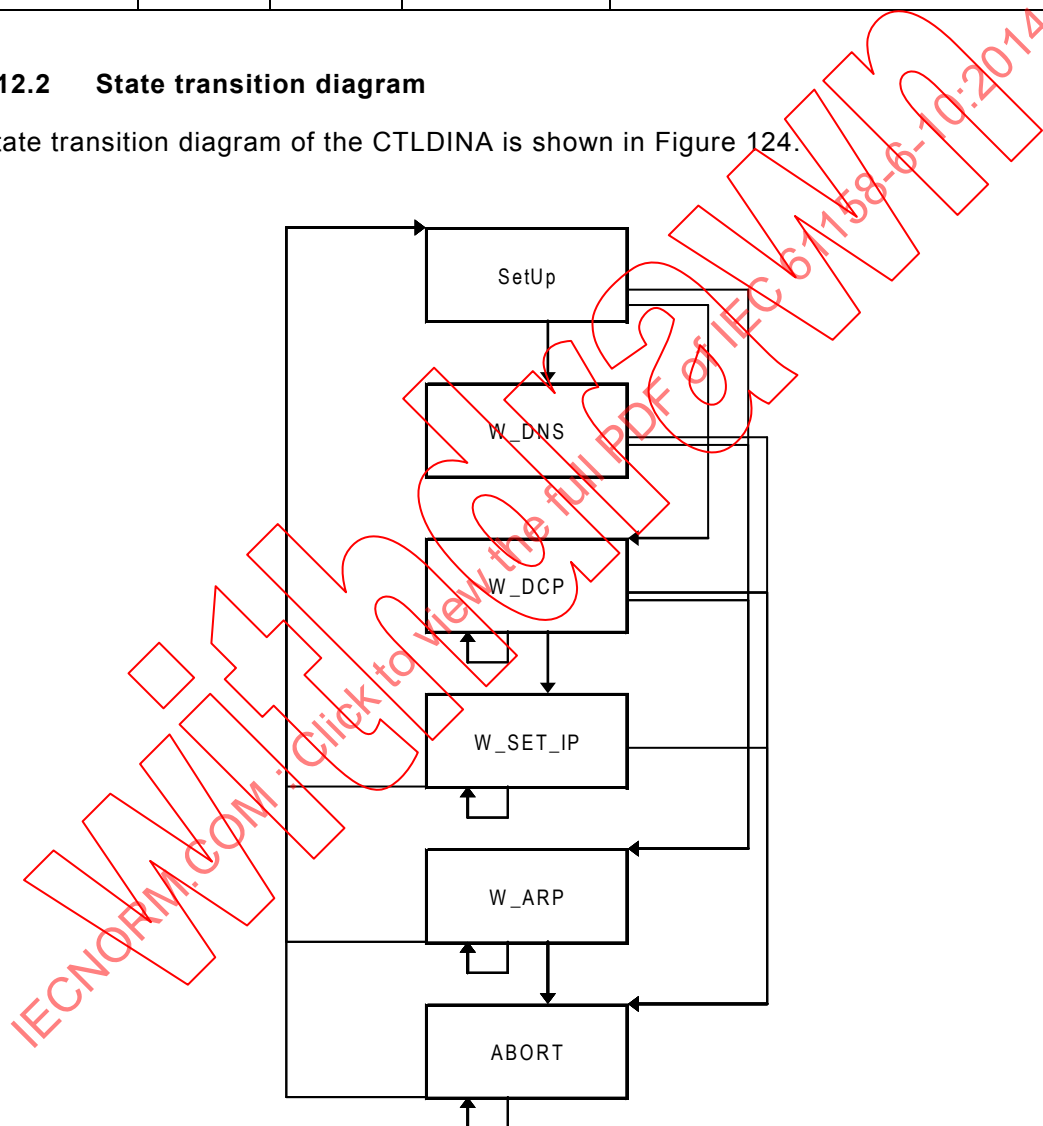


Figure 124 – State transition diagram of CTLDINA

States of the CTLDINA

SetUp	Start the appropriate discovery process and change to the correct next state
W_DNS	Wait for the response of the DNS service and evaluate the result
W_DCP	Wait for the response of the DCP service and evaluate the result
W_SET_IP	Wait for the response of the DCP set service and evaluate the result

W_ARP	Wait for the response of the ARP service and evaluate the result
ABORT	A fault or a late error is detected and signaled

5.6.4.12.3 State machine description

The Context Management Discovery, IP and Name Assignment (CTLDINA) arranges the discovery, IP and station name assignment.

5.6.4.12.4 CTLDINA state table

Table 814 contains the complete description of the CTLDINA state machine.

Table 814 – CTLDINA state table

#	Current State	Event /Condition =>Action	Next State
1	SetUp	CTLDINA_Discover_req () /Add_resolution == DNS => Query := StationName.RealStationName DNS_QueryName.req ()	W_DNS
2	SetUp	CTLDINA_Discover_req () /Add_resolution == DCP => LateError := FALSE Listoffilter := Filter with StationName.RealStationName ResponseDelay := 1 DCP_Identify.req ()	W_DCP
3	SetUp	CTLDINA_Discover_req () /Add_resolution == none => NumberOfARP:=0 IP_Addr := Dev_IP_Para.IP StartTimer (ARP_Timer) ARP_getMAC.req ()	W_ARP
4	W_DNS	DNS_QueryName.cnf (+) => NumberOfARP:=0 IP_Addr := Query_IP_Add StartTimer (ARP_Timer) ARP_getMAC.req ()	W_ARP
5	W_DNS	DNS_QueryName.cnf (-) / Reason == Time-Out => ErrorClass := CTLDINA ErrorCode := Unknown_RealStationName	ABORT
6	W_DCP	DCP_Identify_ind () /Length(ListOfFoundDevices) == 1 && ListOfFoundDevices.IP-Suite == Dev_IP_Para.IP-Suite => LateError:=TRUE NumberOfARP:=0 IP_Addr := ListOfFoundDevices.IP MAC_Addr := ListOfFoundDevices.SA StartTimer (ARP_Timer) //NOTE Late error handling ARP_getMAC.req () CMDINA_Init_cnf (+)	W_ARP
7	W_DCP	DCP_Identify.cnf (+) /Length(ListOfFoundDevices) == 1 && ListOfFoundDevices.IP-Suite == Dev_IP_Para.IP-Suite	W_DCP

#	Current State	Event /Condition =>Action	Next State
		=> ignore	
8	W_DCP	DCP_Identify.cnf (+) /Length(ListOfFoundDevices) == 1 && ! (ListOfFoundDevices.IP-Suite == Dev_IP_Para.IP-Suite) => IP_Addr := Dev_IP_Para.IP MAC_Addr := ListOfFoundDevices.SA SetCmd=IP-Suite DCP_Set.req ()	W_SET_IP
9	W_DCP	DCP_Identify.cnf (+) /Length(ListOfFoundDevices) > 1 => ErrorClass := CTLDINA ErrorCode := Multiple_RealStationName	ABORT
10	W_DCP	DCP_Identify.cnf (+) /Length(ListOfFoundDevices) == 0 => ErrorClass := CTLDINA ErrorCode := No_RealStationName	ABORT
11	W_DCP	DCP_Identify.cnf (-) => ErrorClass := CTLDINA ErrorCode := No_StationName	ABORT
12	W_DCP	DCP_Hello.ind () /ListOfFoundDevices.StationName == StationName.RealStationName && ListOfFoundDevices.IP == VALID => LateError:=TRUE NumberOfARP:=0 IP_Addr := ListOfFoundDevices.IP MAC_Addr := ListOfFoundDevices.SA StartTimer (ARP_Timer) //NOTE Late error handling ARP_getMAC.req () CMDINA_Init.cnf (+)	W_ARP
13	W_DCP	DCP_Hello.ind () /ListOfFoundDevices.StationName != StationName.RealStationName ListOfFoundDevices.IP != VALID => ignore	W_DCP
14	W_SET_IP	DCP_Set.cnf (+) => Dev_MAC_Addr := MAC_Addr CTLDINA_Discover.cnf (+)	SetUp
15	W_SET_IP	DCP_Set.cnf (-) => ErrorClass := CTLDINA ErrorCode := DCP_Error	ABORT
16	W_SET_IP	DCP_Hello.ind () => ignore	W_SET_IP
17	W_SET_IP	DCP_Identify.cnf () => ignore	W_SET_IP
18	W_SET_IP	DCP_Identify_ind () => ignore	W_SET_IP

#	Current State	Event /Condition =>Action	Next State
19	W_ARP	ARP_getMACQ.ind () /IP_Address == Dev_IP_Para.IP && DifferentMacAddress => NumberOfARPs:= NumberOfARPs + 1 Dev_MAC_Addr := MAC_Address	W_ARP
20	W_ARP	ARP_getMACQ.ind () /IP_Address != Dev_IP_Para.IP (IP_Address == Dev_IP_Para.IP && !DifferentMacAddress) => ignore	W_ARP
21	W_ARP	TimerExpired (ARP_Timer) /NumberOfARPs == 0 => ErrorClass := CTLDINA ErrorCode := No_IP_Addr	ABORT
22	W_ARP	TimerExpired (ARP_Timer) /NumberOfARPs > 1 => ErrorClass := CTLDINA ErrorCode := Multiple_IP_Addr	ABORT
23	W_ARP	TimerExpired (ARP_Timer) /NumberOfARPs == 1 && LateError == FALSE => CTLDINA_Discover_cnf (+)	SetUp
24	W_ARP	TimerExpired (ARP_Timer) /NumberOfARPs == 1 && LateError == TRUE => ignore	SetUp
25	W_ARP	DCP_Identify.cnf () /LateError == FALSE => ignore	W_ARP
26	W_ARP	DCP_Identify.cnf (+) /LateError == TRUE && Length(ListOfFoundDevices) > 1 => ErrorClass := CTLDINA ErrorCode := Multiple_RealStationName	ABORT
27	W_ARP	DCP_Identify.cnf (+) /LateError == TRUE && Length(ListOfFoundDevices) == 0 => ErrorClass := CTLDINA ErrorCode := No_RealStationName	ABORT
28	W_ARP	DCP_Identify.cnf (+) /LateError == TRUE && Length(ListOfFoundDevices) == 1 => ignore	W_ARP
29	W_ARP	DCP_Hello.ind () => ignore	W_ARP
30	W_ARP	DCP_Identify_ind () => ignore	W_ARP
31	ABORT		SetUp

#	Current State	Event /Condition =>Action	Next State
		/LateError == FALSE => Close all open underlying state machines CTLDINA_Discover_cnf (-)	
32	ABORT	/LateError == TRUE => Close all open underlying state machines CMCTL_state_ind (ABORT)	SetUp
33	ABORT	DCP_Hello.ind () => ignore	ABORT
34	ABORT	DCP_Identify.cnf () => ignore	ABORT
35	ABORT	DCP_Identify_ind () => ignore	ABORT

5.6.4.12.5 Functions, Macros, Timers and Variables

Table 815 contains the functions, macros, timers and variables used by the CTLDINA and their arguments and their descriptions.

Table 815 – Functions, Macros, Timers and Variables used by the CTLDINA

Name	Type	Function/Meaning
StartTimer	Function	This local function starts or retriggers a timer
StopTimer	Function	This local function stops a timer
TimerExpired	Function	This local function is executed if a timer expires
DifferentMacAddress	Macro	This local macro checks whether the MAC address of the sender of the ARP response is equal or different to the already stored MAC address. The first received ARP is always different.
Length (ListOfFoundDevices)	Macro	This local macro calculates the number of elements of the ListOfFoundDevices list
Store parameters	Macro	This local macro stores parameters in the context of the CTLDINA
ARP_Timer	Timer	This timer is used to monitor the response for the ARP service
ErrorClass	Variable	This local variable contains the error location
ErrorCode	Variable	This local variable contains the error code in the context of the ErrorClass
LateError	Variable	This local variable is used for the late error handling to speed up the startup during normal operation.
ListOfFilters	Variable	This local variable contains the list of DCP options/suboptions which are used for the DCP_Identify service
ListOfFoundDevices	Variable	This local variable contains the list of DCP options/suboptions which are delivered by the DCP_Identify service
NumberOfARP	Variable	This local variable contains amount of received ARP responses
Query	Variable	This local variable contains the list of parameters which are used for the DNS service

5.6.4.12.6 Automatic NameOfStation assignment

The CTLDINA should be extended for automatic NameOfStation assignment. The principle state transition diagram is shown in Figure 125.

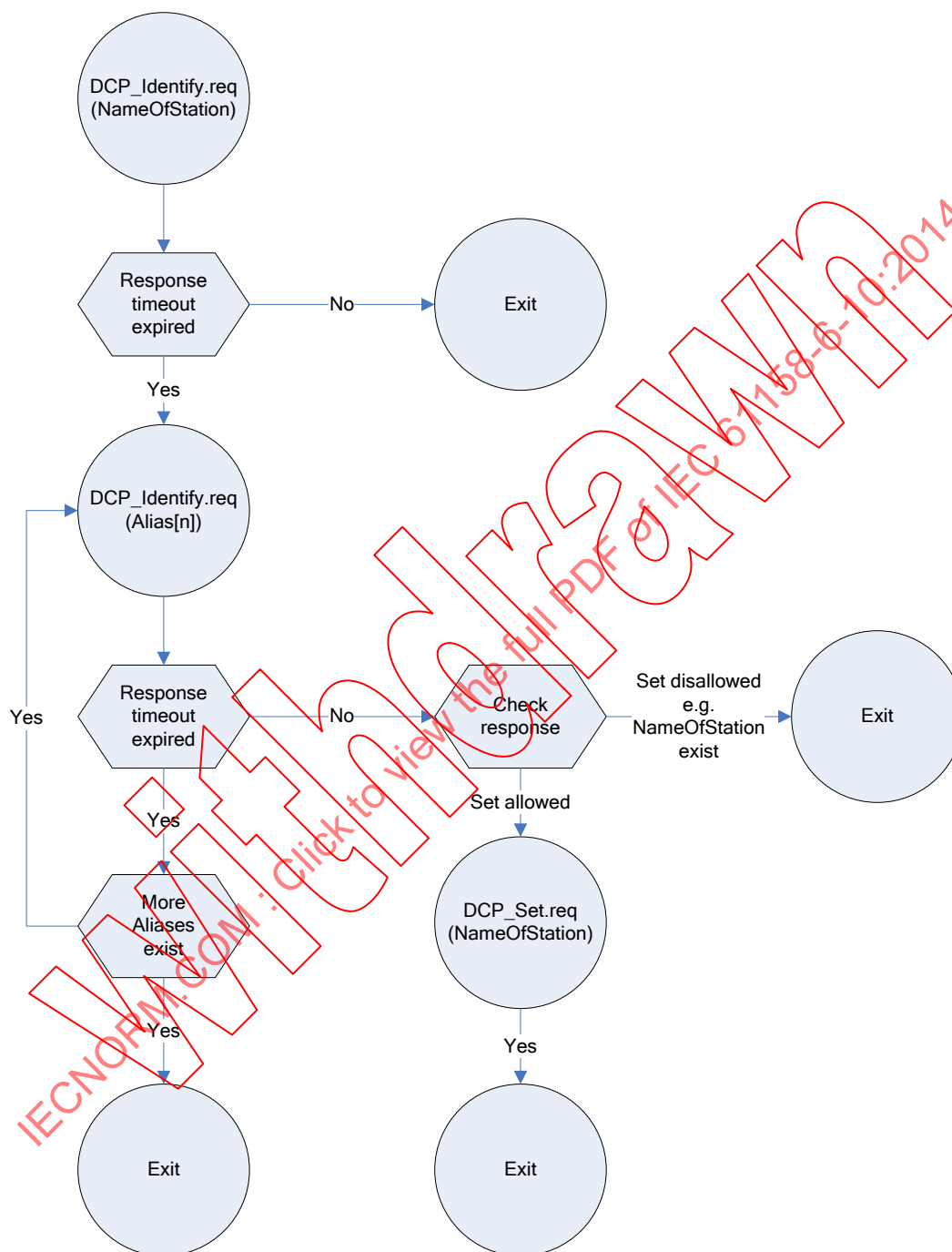


Figure 125 – Automatic NameOfStation assignment

5.7 DLL Mapping Protocol Machines

The DLL Mapping Protocol Machines (DMPM) mapping shall be according to 4.18.2.

Annex A (normative)

Unified establishing of an AR for all RT classes

This standard offers a unified AR establishment for all kinds of CRs. For this reason, the software effort may be reduced.

The AR establishing is parted into period “A” between Connect and PrmEnd and period “B” after PrmEnd.

During “A” the connection monitoring is done locally by the IOD using CMiniatorTimeout and by the IOC using the given RPC monitoring. Additionally a defined “Application timeout” of 300 s is used by the IOC.

During “B” the connection monitoring is switched from the model in period “A” to a connection monitoring using the WDT / DHT of the RT_CLASS_1 / RT_CLASS_2 (GREEN) or RT_CLASS_3 (RED) IOCRs.

NOTE The IOD (acting as RPC server) needs no additional timeout, because it activates the CMiniatorTimeout again, after the RPC response was sent.

Table A.1 shows the attributes of the examples.

Table A.1 – Examples for the AR establishing

Figure	Traffic mode	Meaning
Figure A.1, Figure A.2, Figure A.3, Figure A.4 and Figure A.5	GREEN or RED	All kinds of RT and IRT communication The connection monitoring is done in parallel with RPC and RT. As long as the GREEN / RED monitoring isn't established, the monitoring is done by RPC. This figure shows the RPC communication thread which is synchronized with the RT communication thread. For interoperability with previous versions see Table B.1.

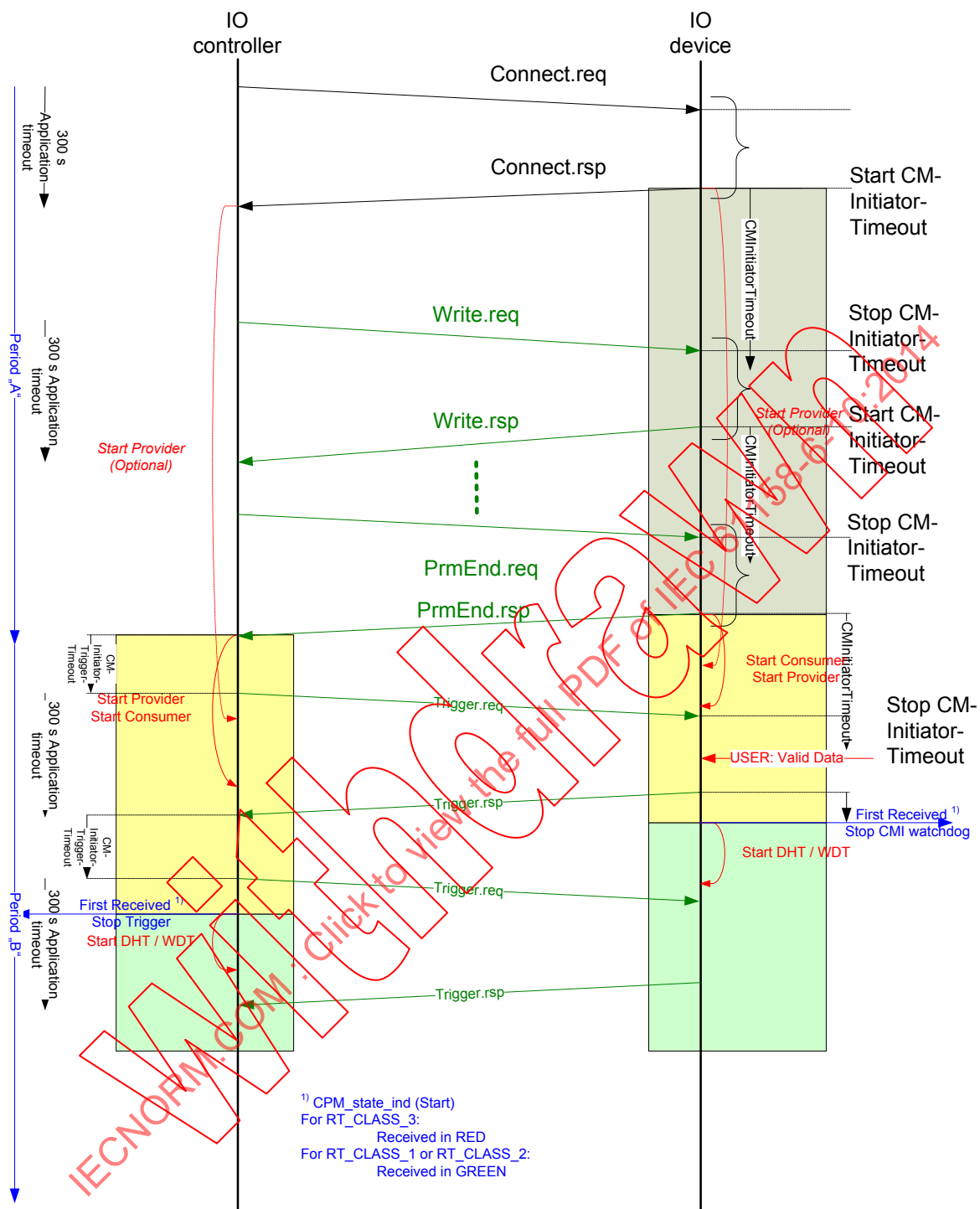


Figure A.1 – Establishing of an AR using RT_CLASS_1, RT_CLASS_2 or RT_CLASS_3 (Initial connection monitoring w/o RT)

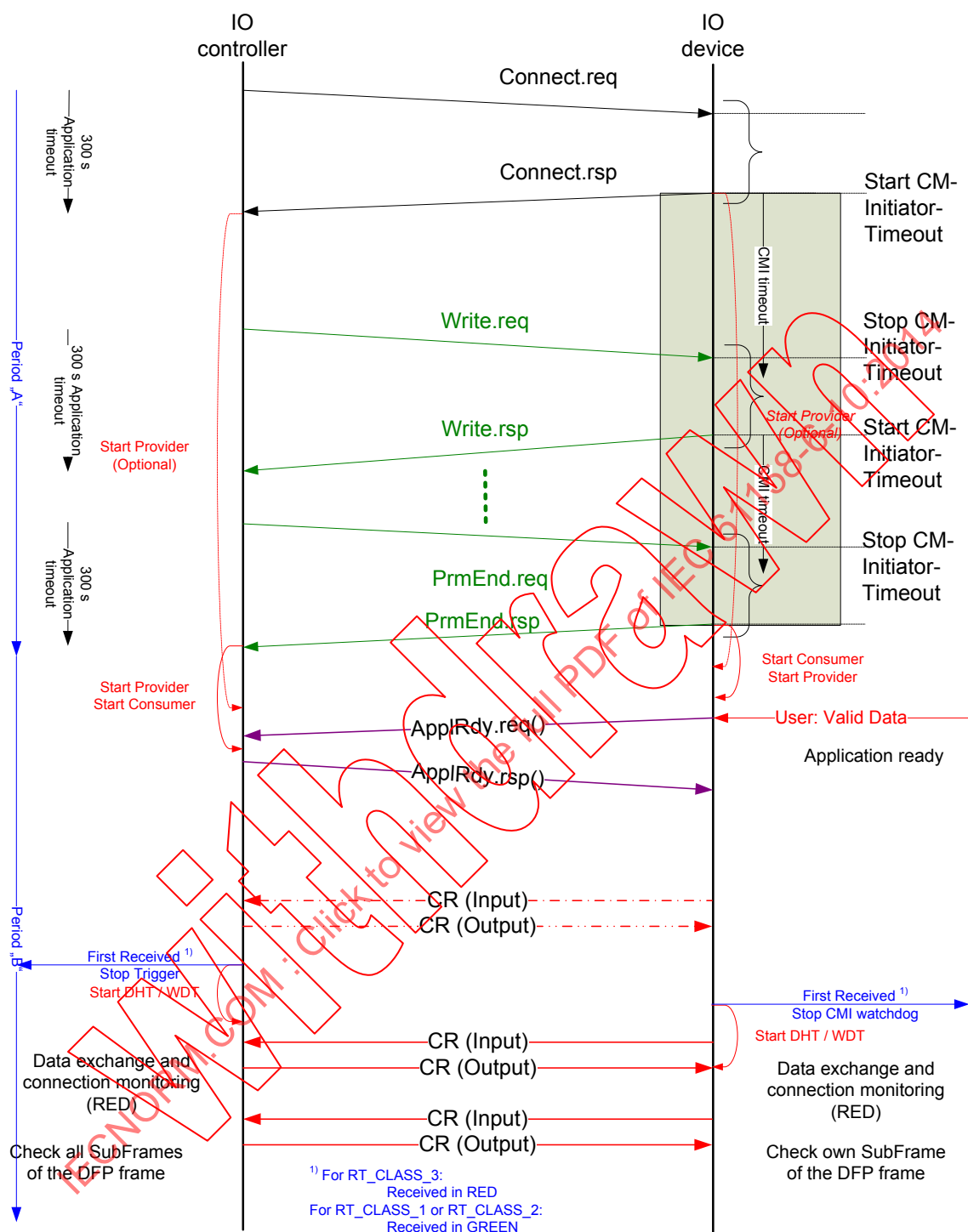


Figure A.2 – Establishing of an AR using RT_CLASS_1, RT_CLASS_2 or RT_CLASS_3 (Connection monitoring with RT)

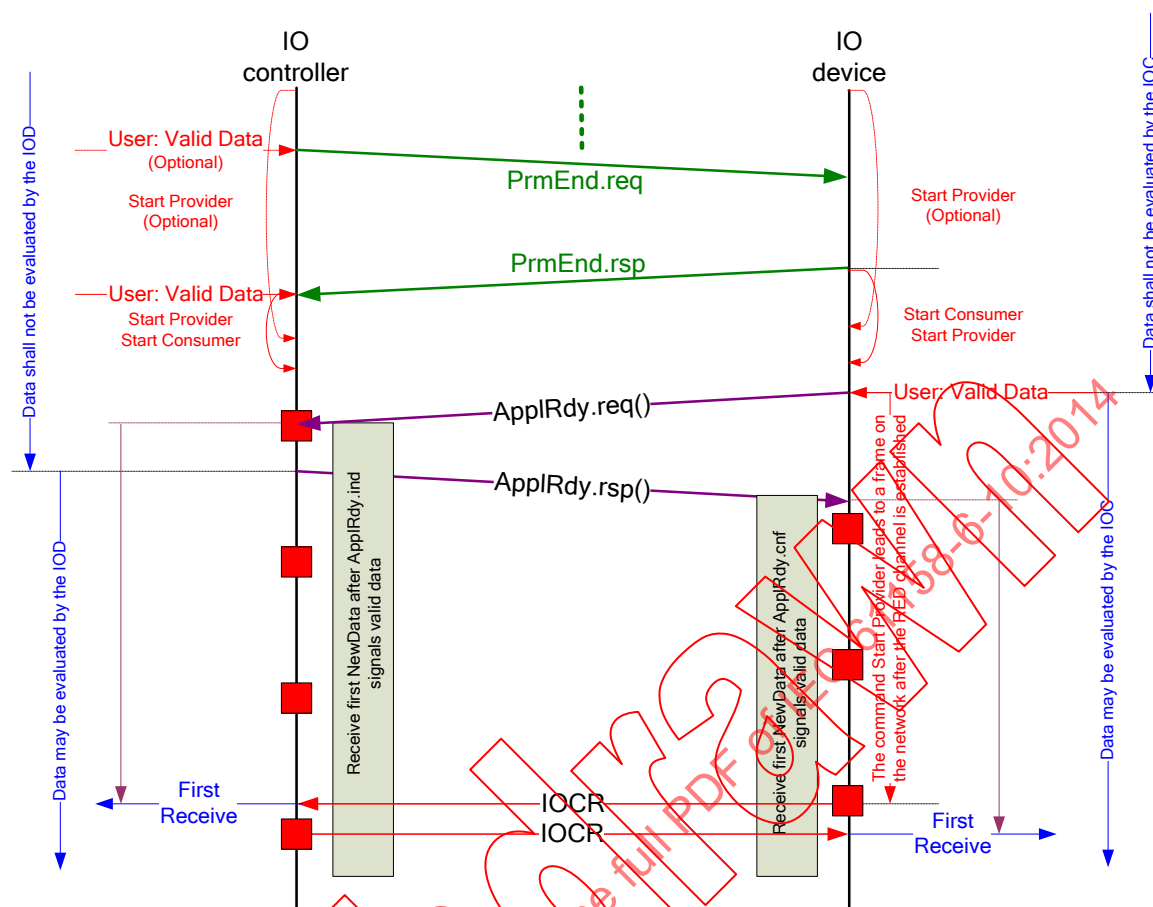


**Figure A.3 – Principle of the data evaluation during startup
(RED channel establishment delayed)**

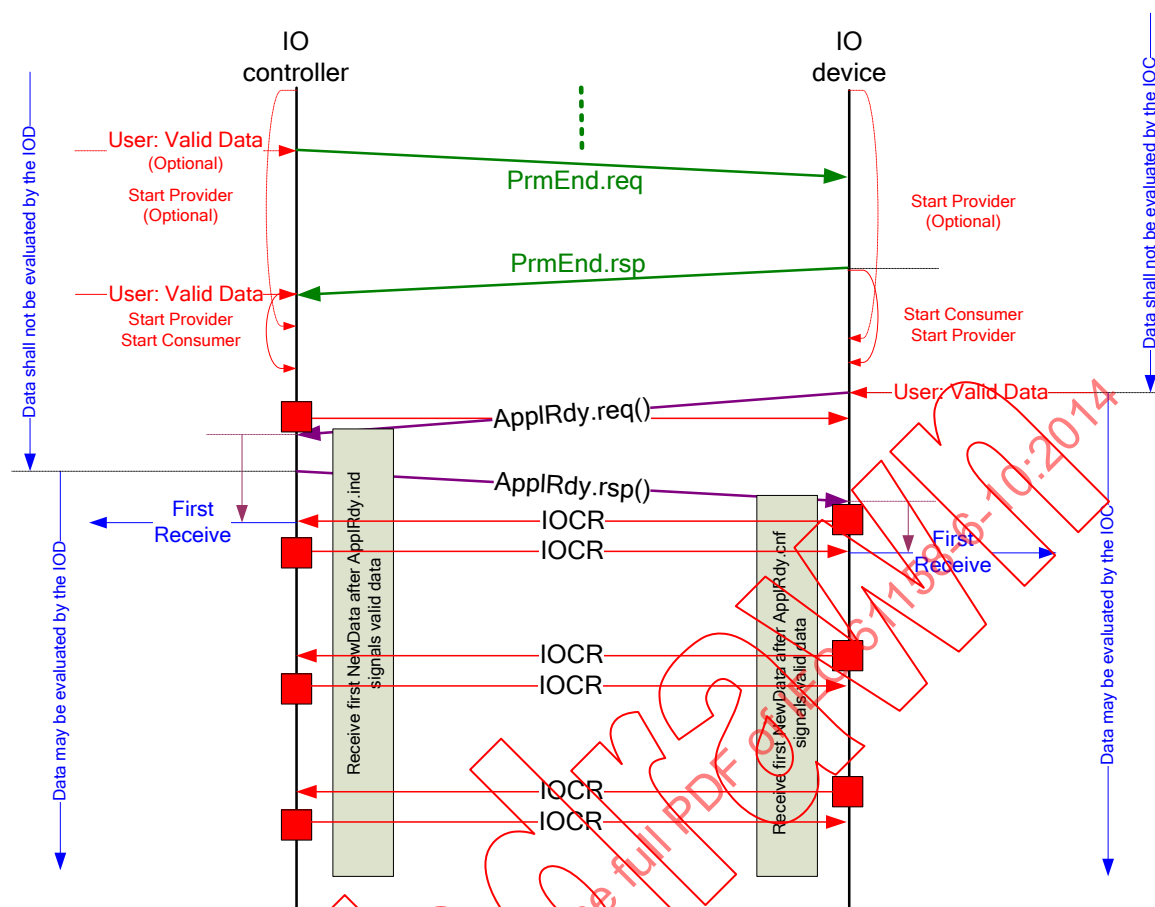


Figure A.4 – Principle of the data evaluation during startup (RED channel establishment early)

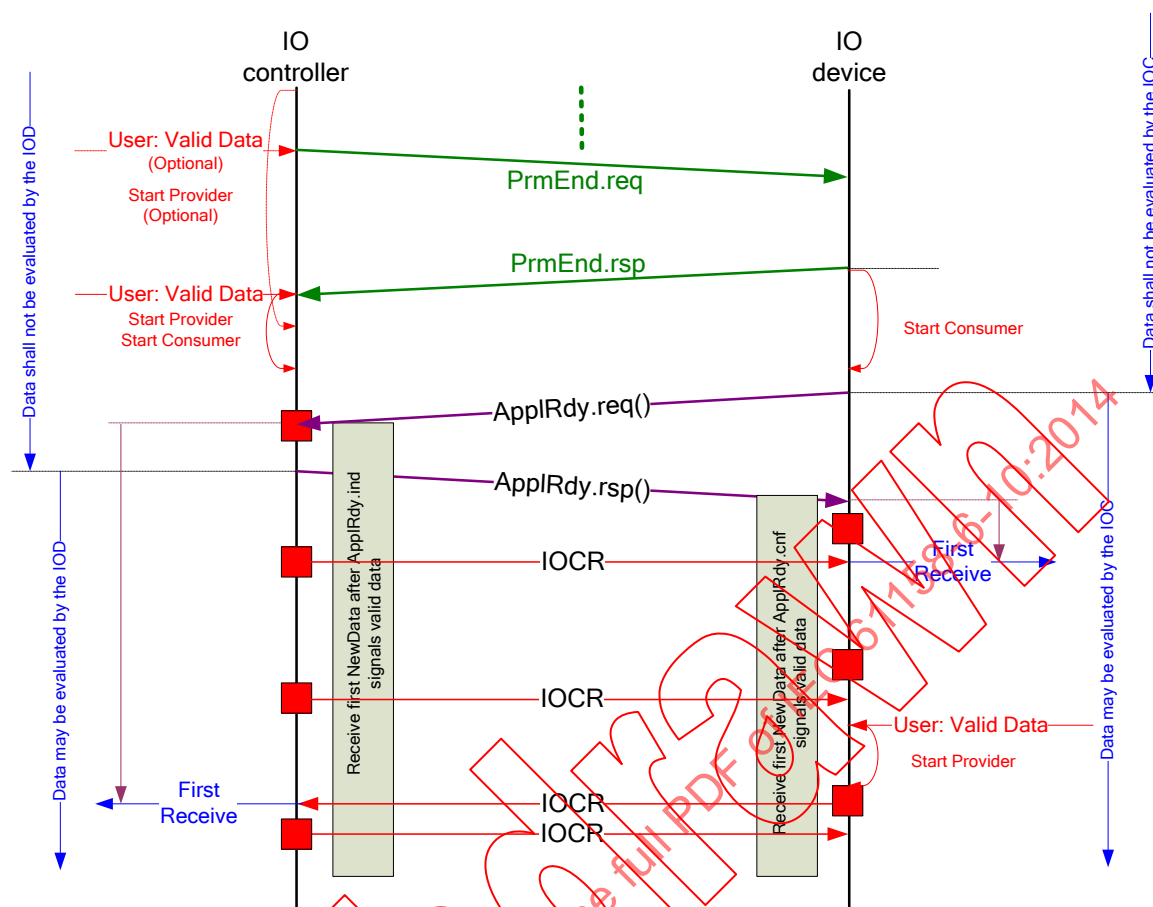


**Figure A.5 – Principle of the data evaluation during startup
(Special case: Isochronous mode application)**

An isochronous mode application needs to be isochronous at application level. This may delay the existence of valid data. The device may start later with its then valid data, to avoid a prolongation of Application Ready.

Annex B (normative)

Compatible establishing of an AR

This standard offers compatibility, encoded by `ARProperties.StartupMode == Legacy`, for all kind of ARs. Table B.1 shows the attributes of the examples. The IO device defines the speed of the establishing.

Table B.1 – Examples for compatible AR establishing

Figure	Mode	Meaning
Figure B.1	<code>ARProperties.StartupMode == Legacy</code>	IRT communication with <code>SendClock</code> greater or equal 250 μ s This compatible mode ensures interoperability with the earlier versions of this standard.
Figure B.2	<code>ARProperties.StartupMode == Legacy</code>	RT communication with <code>SendClock</code> greater or equal 250 μ s This compatible mode ensures interoperability with the earlier versions of this standard.

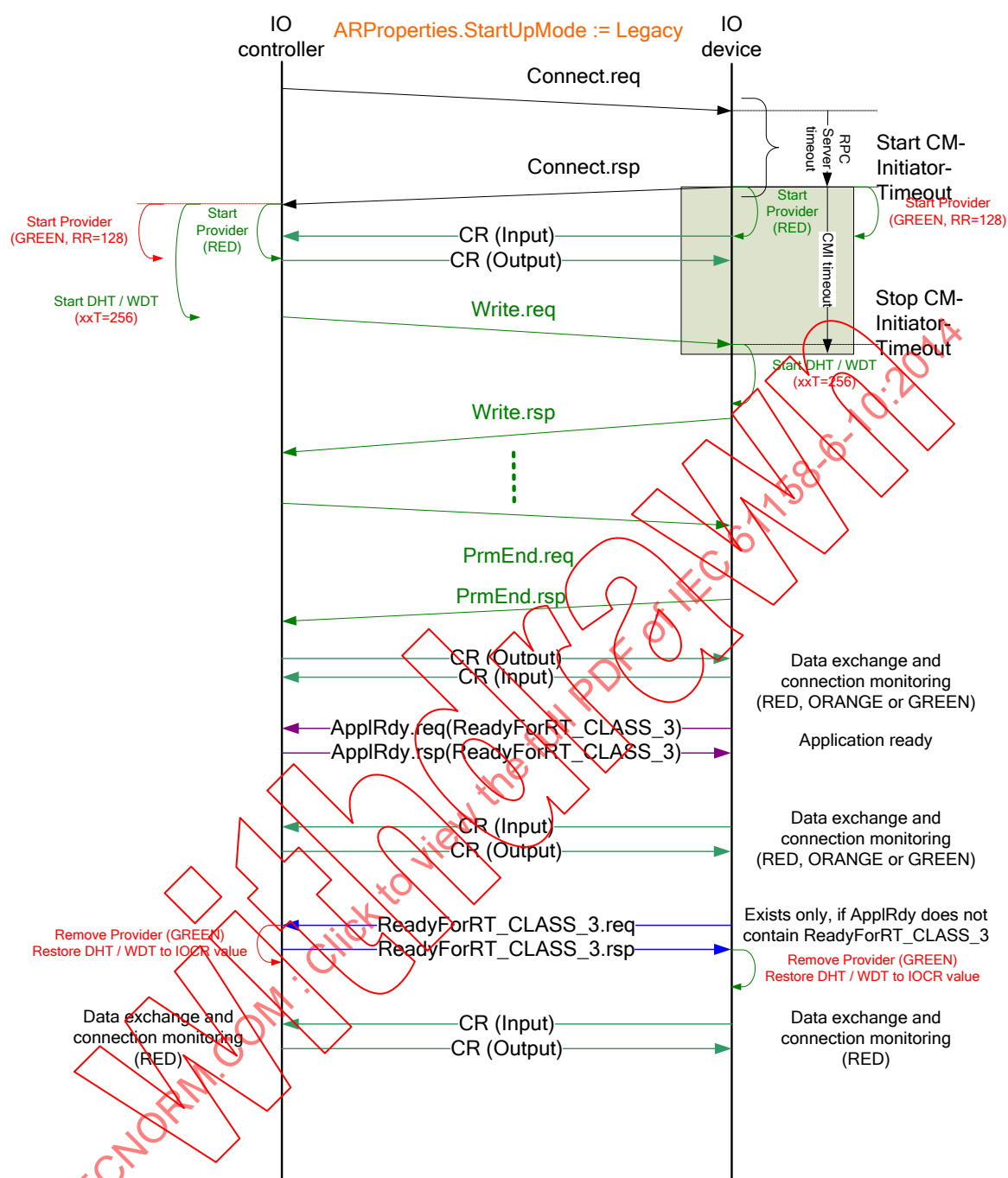


Figure B.1 – Establishing of an AR using RT_CLASS_3 AR with startup mode "Legacy"

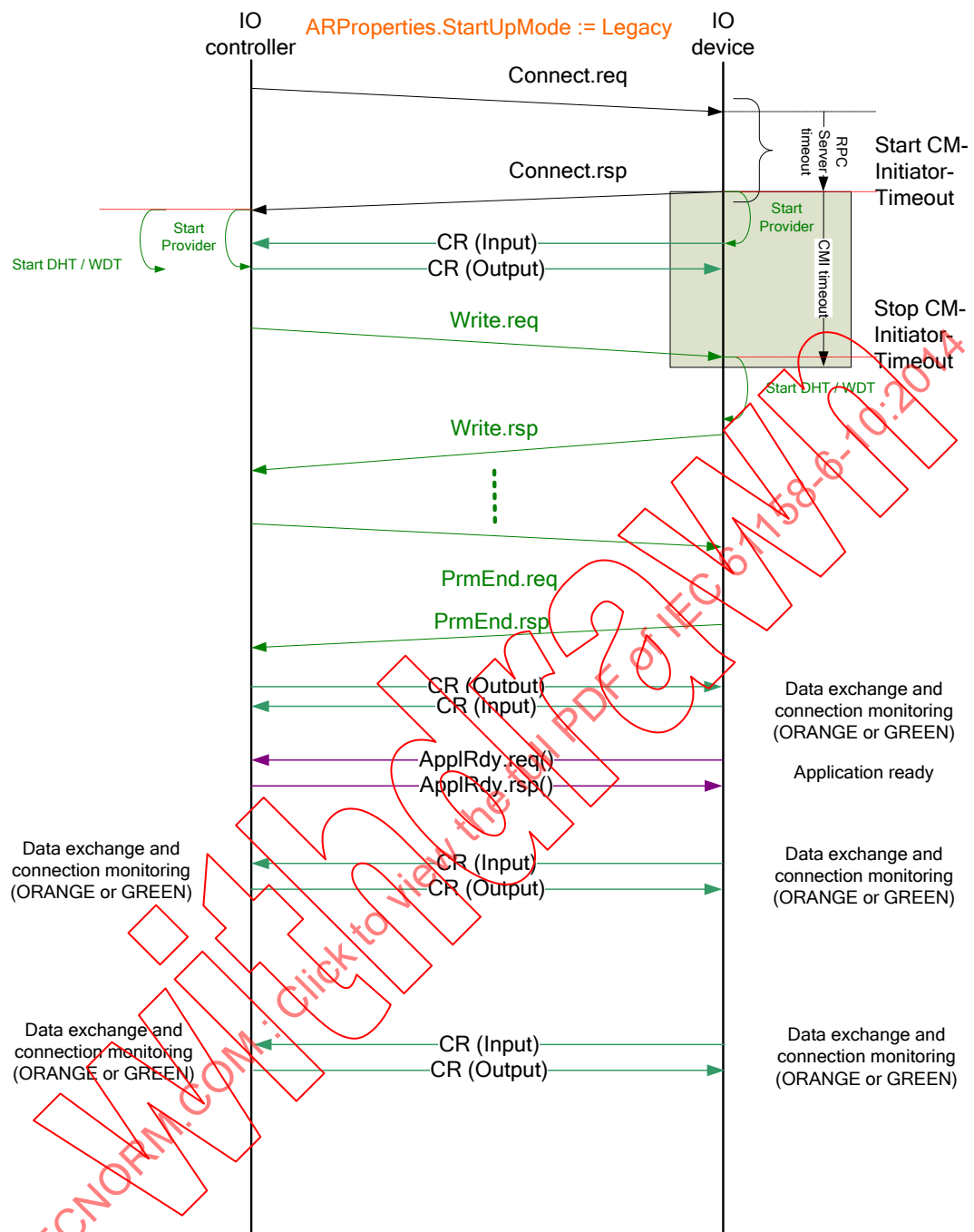


Figure B.2 – Establishing of an AR using RT_CLASS_1, 2 or UDP AR with startup mode "Legacy"

Annex C (informative)

Establishing of a device access AR

As an example of a device access AR see Figure C.1. Only Connectionless RPC is used to drive this kind of AR.

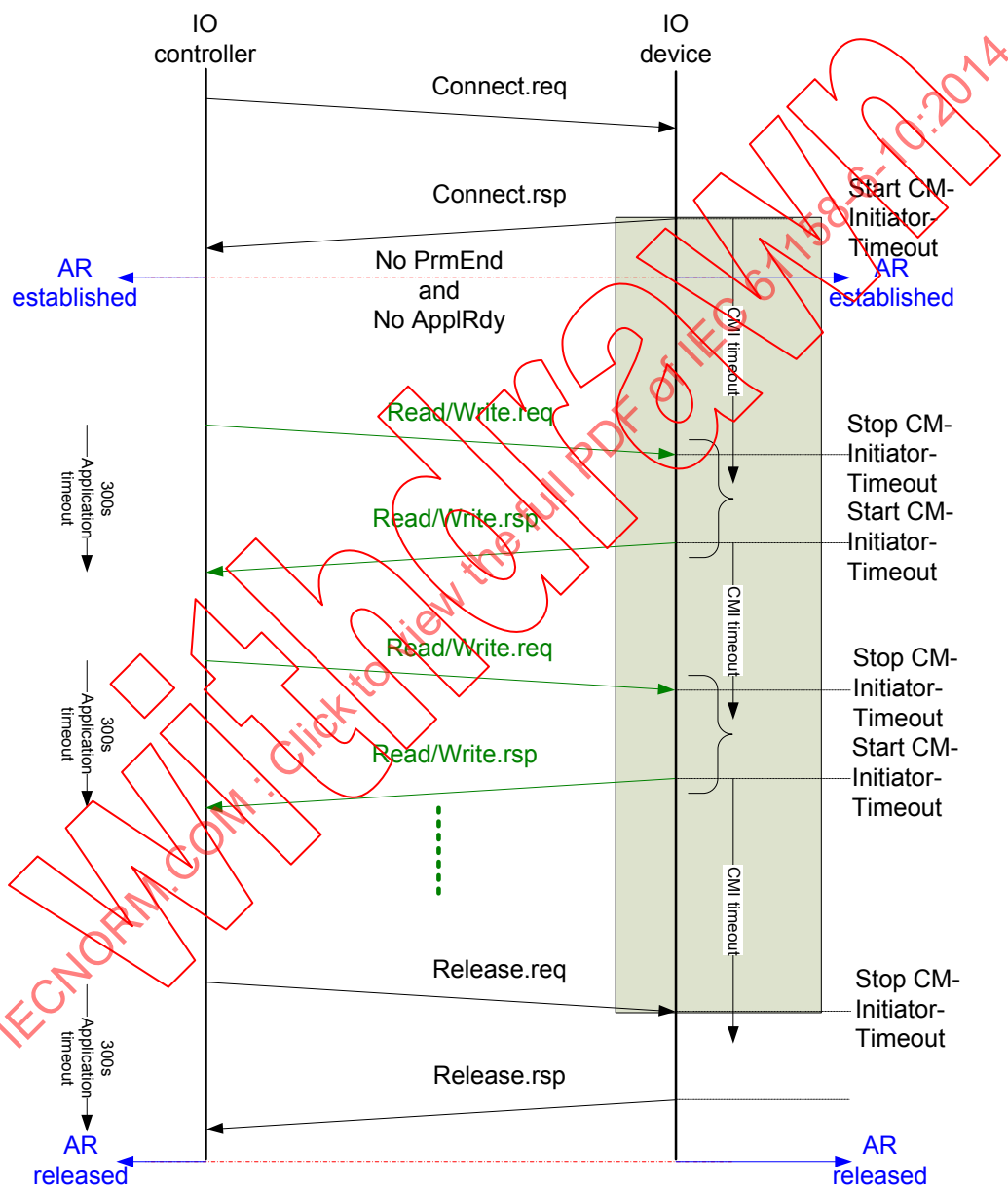


Figure C.1 – Establishing of a device access AR

Annex D (informative)

Establishing of an AR (accelerated procedure)

As an example of an accelerated startup see Figure D.1. The timeout/response control for an accelerated startup is done by concurrent running timers. After the response the state machine of the IO controller go immediate to the next state. This model replaces the cascading of three timers (Identify timeout, Set timeout and ARP timeout) as shown in Formula (D.1) by a model which cascades only $\Delta t \text{ IOD}_x$ and $\Delta t \text{ IOC}_y$ as shown in Formula (D.2).

$$\begin{aligned} \text{Startup Delay before Connect} = & \Delta t \text{ IOC}_1 + \text{Identify timeout}(1 \text{ s}) \\ & + \Delta t \text{ IOC}_2 + \Delta t \text{ IOC}_3 \\ & + \text{ARP timeout}(2 \text{ s}) + \Delta t \text{ IOC}_4 \end{aligned} \quad (\text{D.1})$$

where

Startup Delay before Connect	is the startup before connect variable
$\Delta t \text{ IOC}_1$	is the time between IO controller startup and first received hello
Identify timeout	is the identify timeout variable
$\Delta t \text{ IOC}_2$	is the time between sending the ARP request after identification
$\Delta t \text{ IOC}_3$	is the time between sending the set request after address resolution
$\Delta t \text{ IOC}_4$	is the time between sending the connect request after setting the IP address
ARP timeout	is the ARP timeout variable

$$\begin{aligned} \text{Startup Delay before Connect} = & \Delta t \text{ IOC}_1 + \Delta t \text{ IOD}_1 + \Delta t \text{ IOC}_2 + \Delta t \text{ IOD}_2 \\ & + \Delta t \text{ IOC}_3 + \Delta t \text{ IOD}_3 + \Delta t \text{ IOC}_4 \end{aligned} \quad (\text{D.2})$$

where

Startup Delay before Connect	is the startup before connect variable
$\Delta t \text{ IOC}_1$	is the time between IO controller startup and first received hello
$\Delta t \text{ IOD}_1$	is the time between responding the identify request
$\Delta t \text{ IOC}_2$	is the time between sending the ARP request after identification
$\Delta t \text{ IOD}_2$	is the time between responding the ARP request
$\Delta t \text{ IOC}_3$	is the time between sending the set request after address resolution
$\Delta t \text{ IOD}_3$	is the time between responding the set request
$\Delta t \text{ IOC}_4$	is the time between sending the connect request after setting the IP address

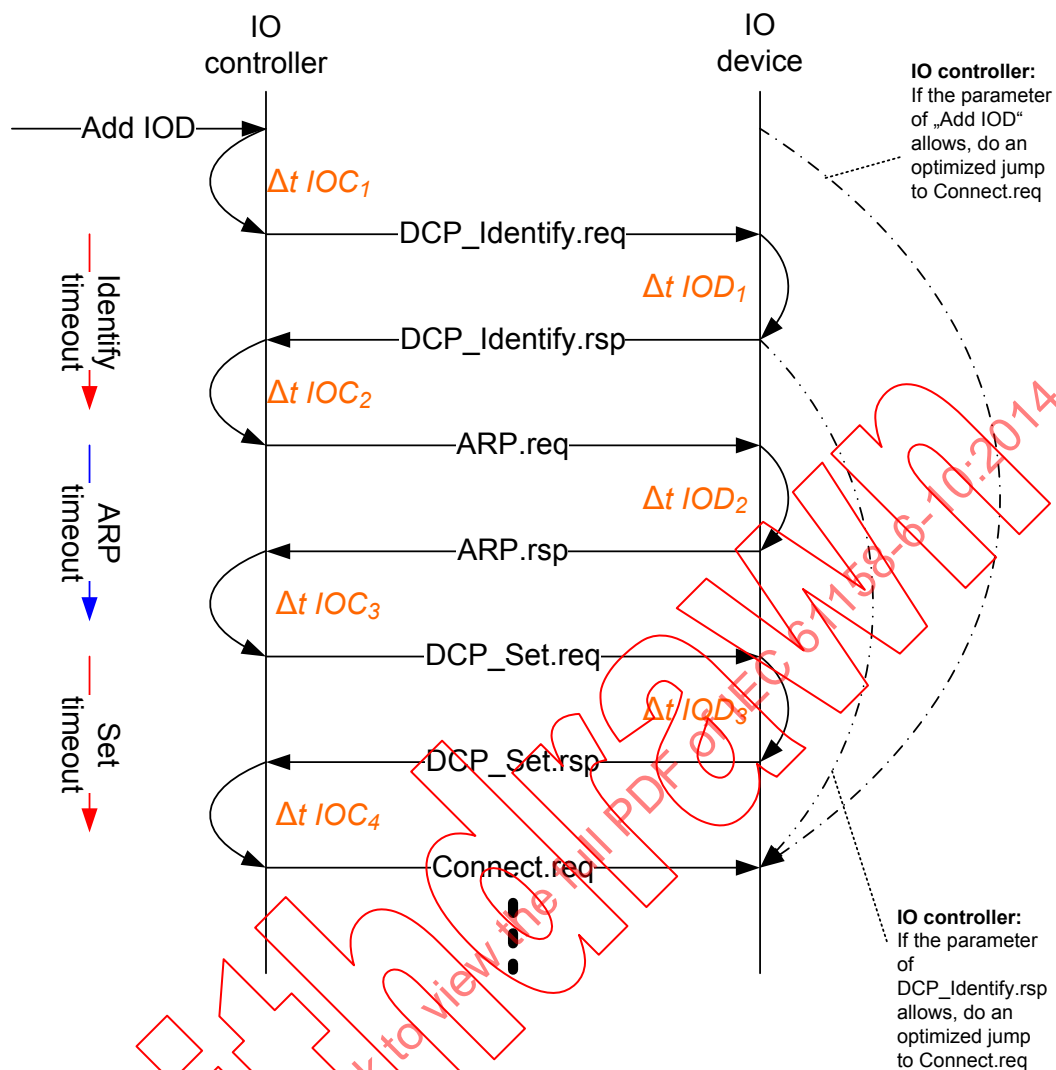


Figure D.1 – Accelerated establishing of an IOAR without error

Figure D.2 shows an example of an accelerated startup with a “late” error. In this case the connect procedure shall be abort according to the state machine of the IO controller.

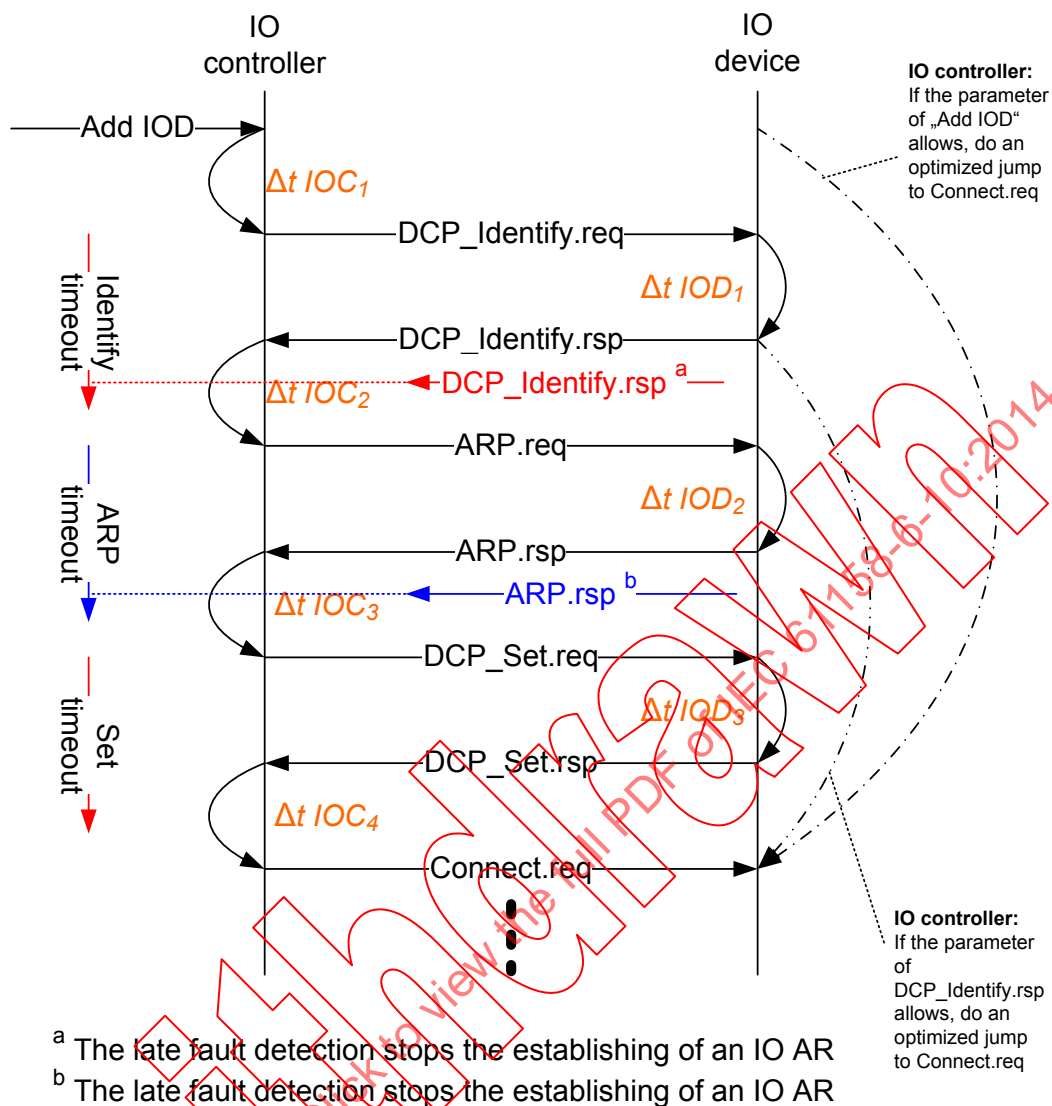


Figure D.2 – Accelerated establishing of an IOAR with “late” error

Annex E (informative)

Establishing of an AR (fast startup procedure)

The accelerated startup shown in Annex D offers a fair ability for faster systems. For systems which produce with different tools e.g. robots with toolkits, the accelerated startup needs additional enhancements. This so called fast startup is shown in Figure E.1. In this case the IO device offers the IO controller with a DCP_Hello.req all required information to directly jump to the Connect service. The required delay is decrease as shown in Formula (E.1) and (E.2).

$$\text{Startup Delay before Connect} = \Delta t \text{ IOC}_1 + \Delta t \text{ IOC}_2 \quad (\text{E.1})$$

where

Startup Delay before Connect	is the startup before connect variable
$\Delta t \text{ IOC}_1$	is the time between IO controller startup and first received hello
$\Delta t \text{ IOC}_2$	is the time between first received hello and connect service

If the startup state machine of the IO controller is active before the IO device is switched on, then Formula (E.2) can be assumed.

$$\text{Startup Delay before Connect} = \Delta t \text{ IOC}_2 \quad (\text{E.2})$$

where

Startup Delay before Connect	is the startup before connect variable
$\Delta t \text{ IOC}_2$	is the time between first received hello and connect service

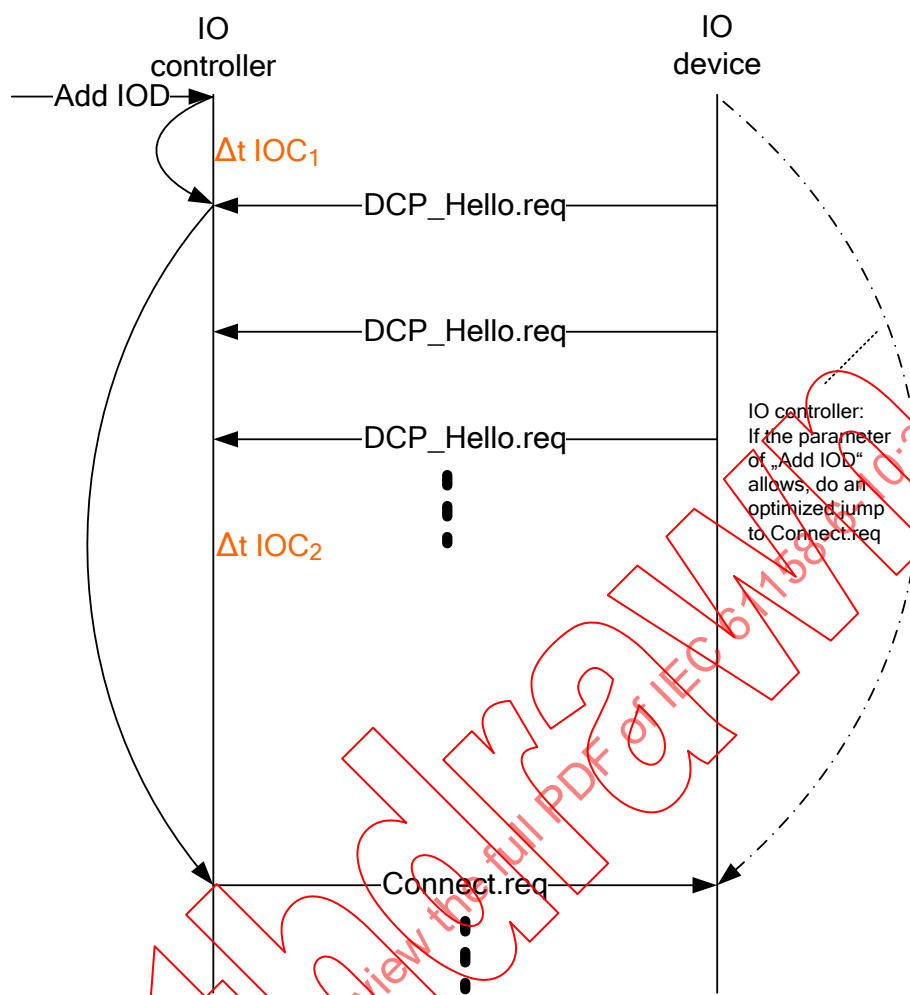


Figure E.1 – Establishing of an IOAR using fast startup

Annex F (informative)

Example of the upload, storage and retrieval procedure

More and more IO devices need commissioning before they can do their task. Sensors, robots and light shields need to be taught. This work, done by a commissioning engineer, leads to parameters which only exist in the IO device.

These commissioning or teaching parameters are stored persistently in the IO device, to keep them available after a power cycle.

To cover a malfunction of the IO device, these parameters should also be stored outside of the IO device. This external storage may be any kind of non-volatile media like a removable memory card. It also could be the memory of the host of the IO controller.

The upload and retrieval alarm notification offers in combination with read and write record the ability to do an upload with storage and a retrieval of these parameters.

Figure F.1 shows an example of upload with storage and Figure F.2 shows an example of retrieval of parameters.

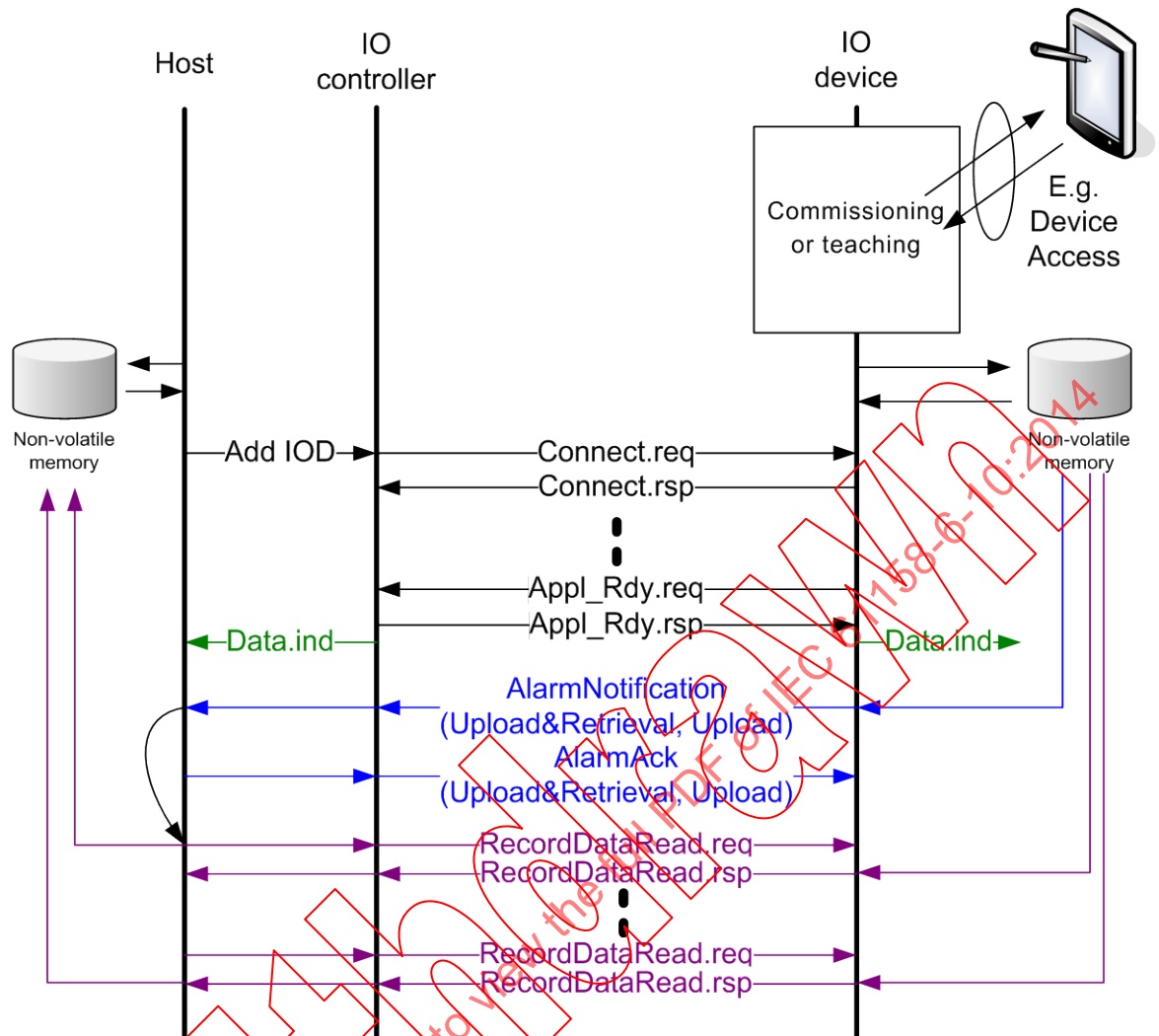


Figure F.1 – Example of upload with storage

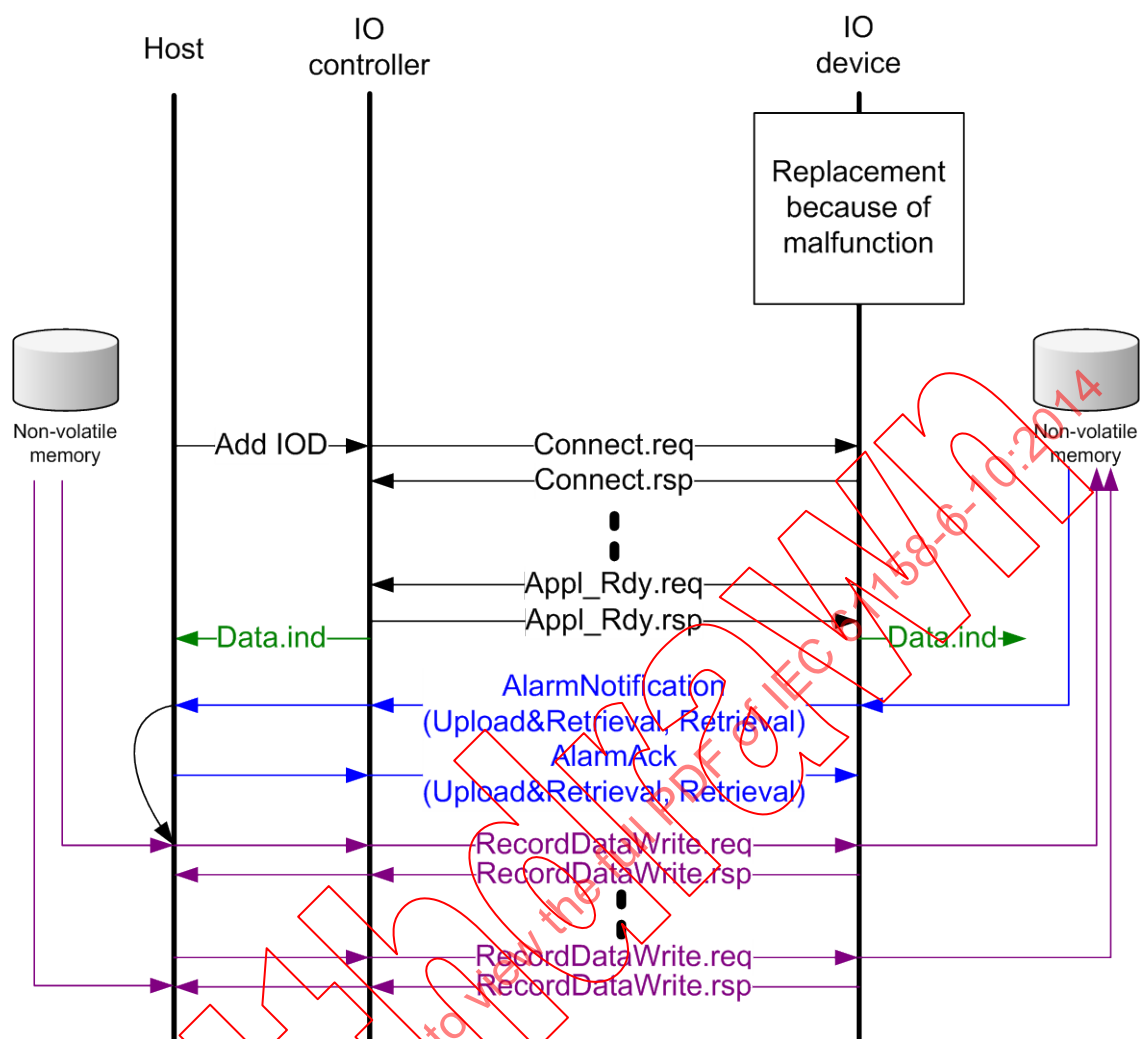


Figure F.2 – Example of retrieval with storage

Annex G (informative)

OSI reference model layers

Figure G.1 shows the assignment of used protocols to the OSI reference model layers.

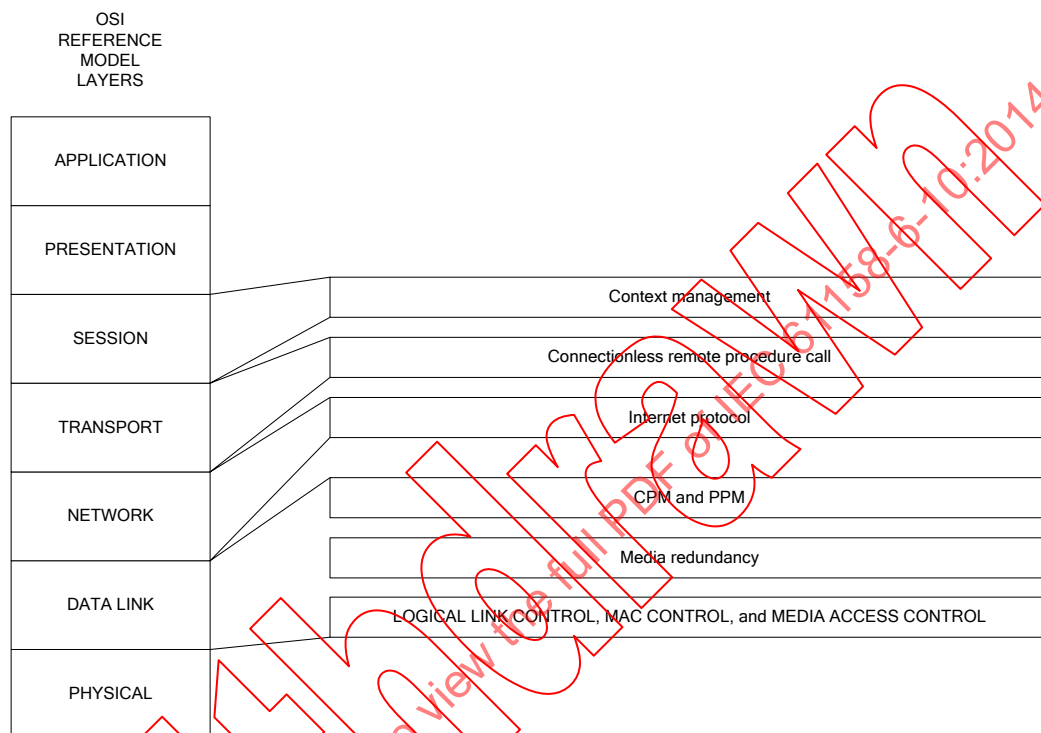


Figure G.1 – Assignment of the OSI reference model layers

Annex H (informative)

Overview of the IO controller and the IO device state machines

Figure H.1 illustrates the general structure of the AL of an IO controller by showing its state machines and the services used by them.

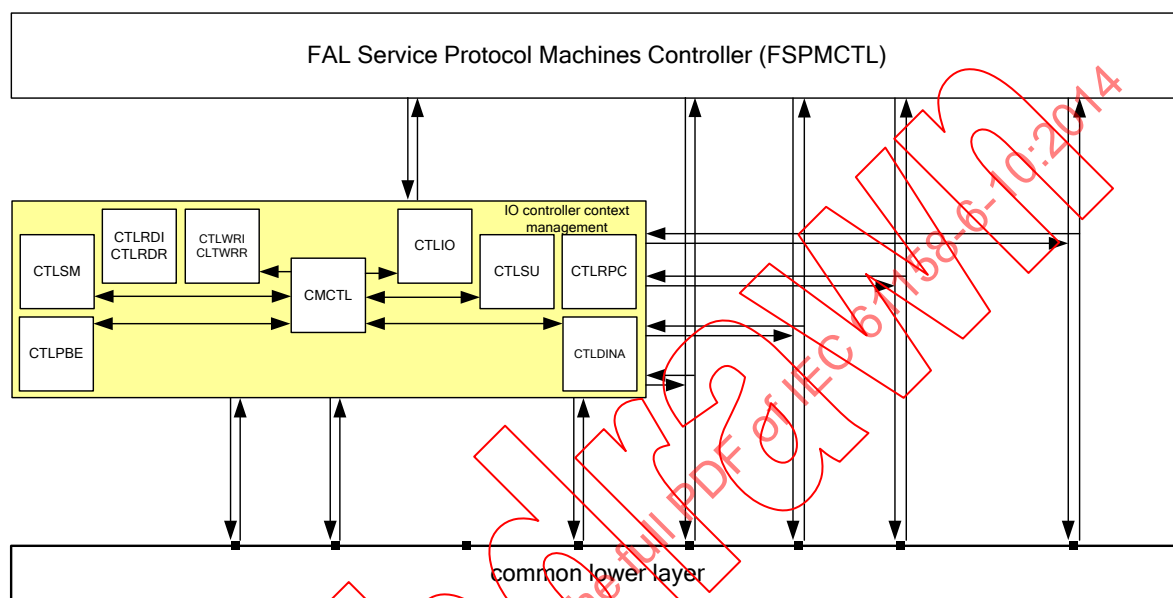


Figure H.1 – Overview of the IO controller state machines

Figure H.2 illustrates the general structure of the AL of an IO device by showing its state machines and the services used by them.

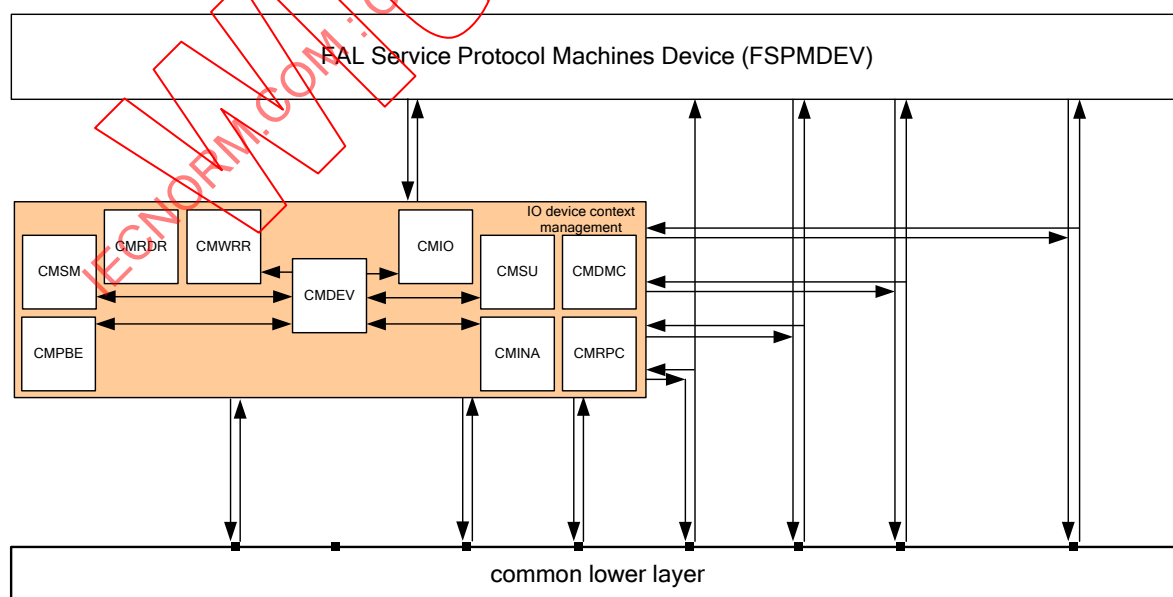


Figure H.2 – Overview of the IO device state machines

Figure H.3 illustrates the general structure of the common AL by showing its state machines.

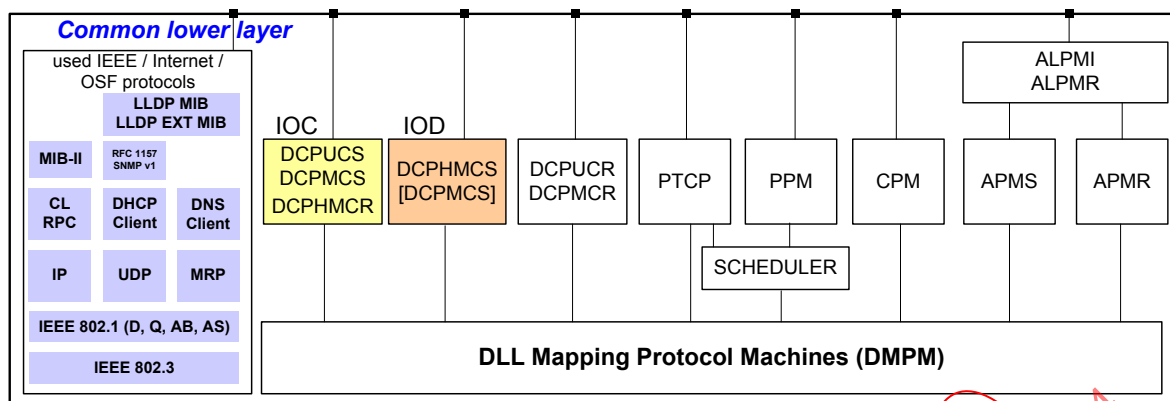


Figure H.3 – Overview of the common state machines

Annex I (informative)

Priority regeneration

According to IEEE 802.1Q and this standard Table I.1 shows the priority regeneration for a transmit port. The priority decreases from row “Network control high” to row “2, 1”.

Table I.1 – Priority regeneration and queue usage

Priority	Criteria	GREEN PERIOD	RED PERIOD	Meaning
Network control high (IEEE 802.1D)	PTCP synchronization frames	Transmit	Store	Network management PTCP synchronization
Network control low (IEEE 802.1D)	Other management protocols	Transmit	Store	Network management LLDP, PTCP Line delay measurement, media redundancy, ...
7 (IEEE 802.1Q)	VLAN priority	Transmit	Store	PTCP announce
6.2 Red queue “planned forwarding” (IEEE 802.1Q)	RT_CLASS_3 frames	Discard ^a	Transmit	Exists only if RED period is established No frames shall be in the red queue, if the RTC3 port state machine is not in state RTCLASS3_RUN. In this case all RT_CLASS_3 frames are received “outside” of the RED PERIOD
6.1 (IEEE 802.1Q)	RT_CLASS_1 and RT_CLASS_2 (GREEN) frames	Transmit	Store	RT_CLASS_1 frames and RT_CLASS_2 frames
6.0 (IEEE 802.1Q)	VLAN priority	Transmit	Store	Alarm high
5 (IEEE 802.1Q)	VLAN priority	Transmit	Store	Alarm low
4 (IEEE 802.1Q)	VLAN priority	Transmit	Store	—
3 – 0 (IEEE 802.1Q)	See Table I.2 or Table I.3. This standard recommends Table I.3.			

^a RED frames are always transmitted before the end of the RED period if stored in the red queue. This queue shall be discarded, if the RTC3PSM state is no longer RTCLASS3_RUN and any frame is stored.

Table I.2 – Priority regeneration – variant 1

Priority	Criteria	GREEN PERIOD	RED PERIOD	Meaning
3, 0 (IEEE 802.1Q)	VLAN priority	Transmit	Store	Other frames without VLAN TAG
2, 1 (IEEE 802.1Q)	VLAN priority	Transmit	Store	—

Table I.3 – Priority regeneration – variant 2

Priority	Criteria	GREEN PERIOD	RED PERIOD	Meaning
3, 2 (IEEE 802.1Q)	VLAN priority	Transmit	Store	—
0, 1 (IEEE 802.1Q)	VLAN priority	Transmit	Store	Other frames without VLAN TAG

NOTE For further information see IEEE 802.1Q.

Annex J (informative)

Overview of the PTCP synchronization master hierarchy

Figure J.1 and Figure J.2 illustrates the general structure of the hierarchy of synchronization masters. The top level masters may synchronize all lower levels, but a lower level master shall never synchronize a higher level.

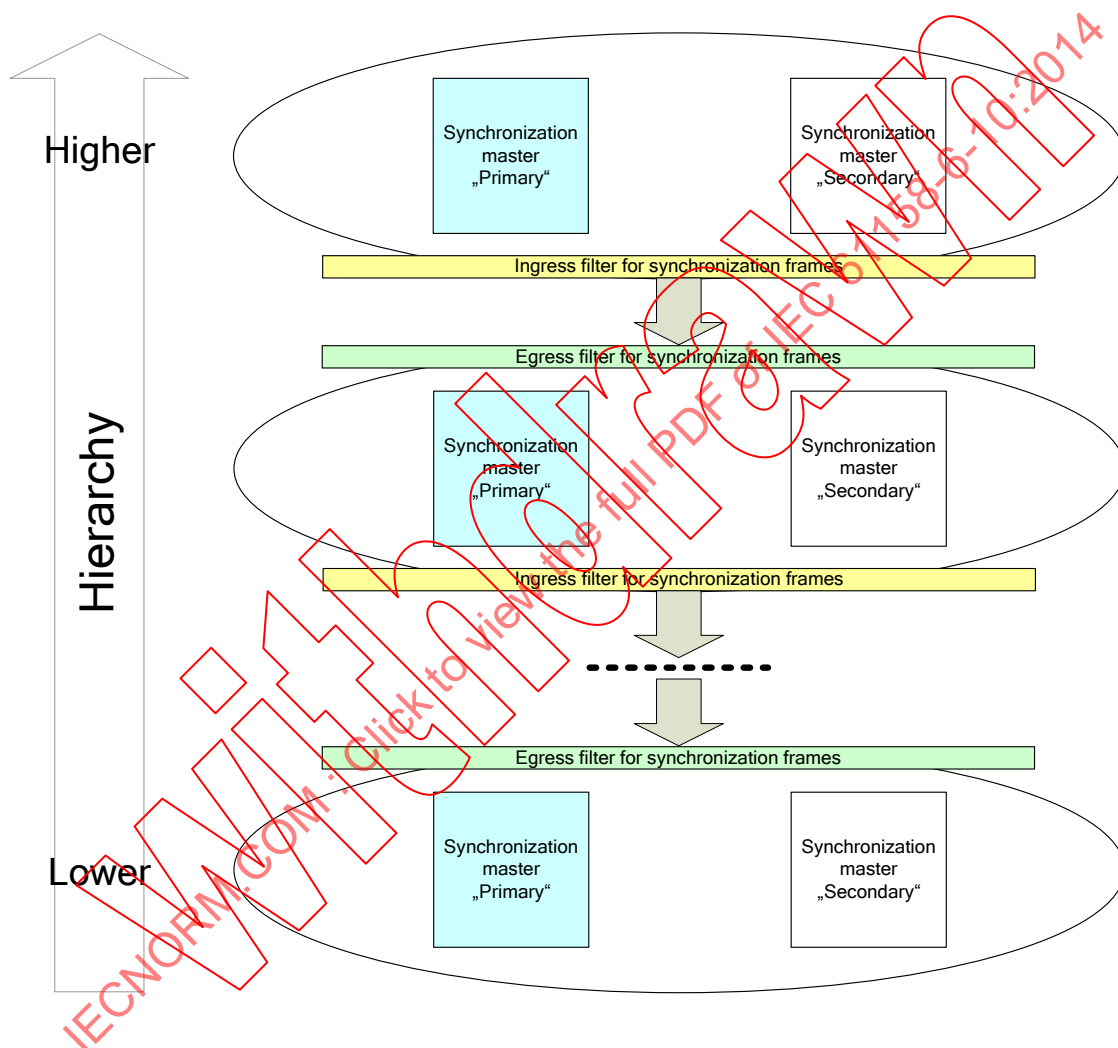


Figure J.1 – Level model for synchronization master hierarchy

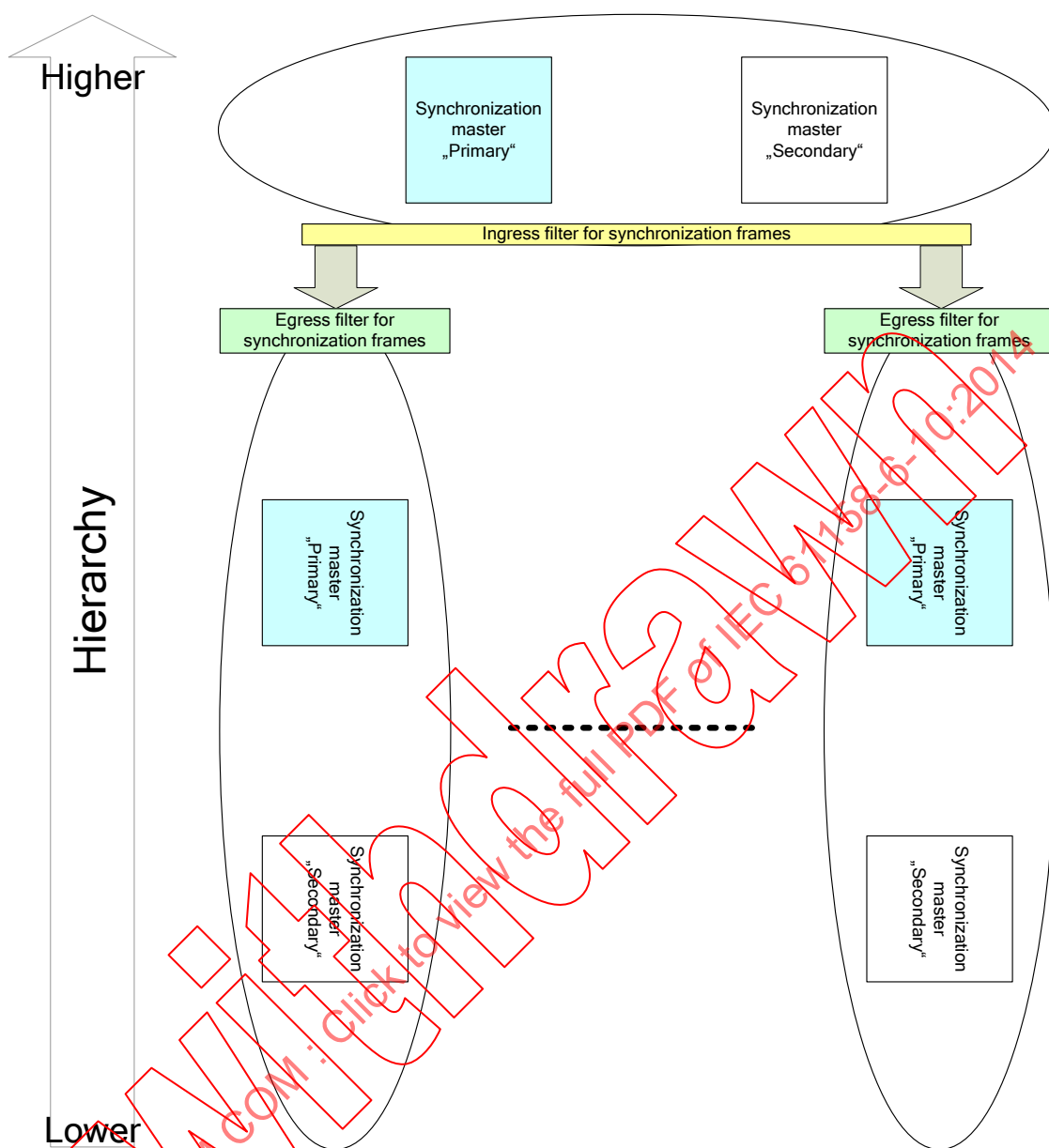


Figure J.2 – Two level variant of the synchronization master hierarchy

Annex K (informative)

Optimization of bandwidth usage

Figure K.1 illustrates the structure of a linear network. When transmitting frames through this network the line delay and the bridge delay delays them in each node. From the IOC point of view, Figure K.2 shows the transmit and Figure K.3 shows the receive direction in a full duplex network.

When reducing the RED period in each node to the required time, additional bandwidth will be available.

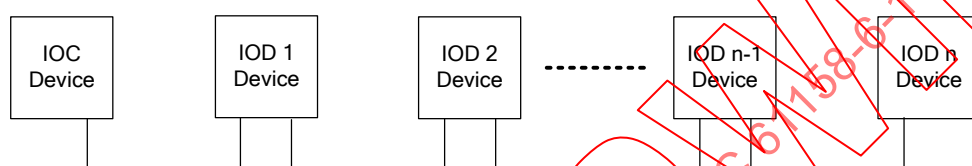


Figure K.1 – Devices build up in a linear structure

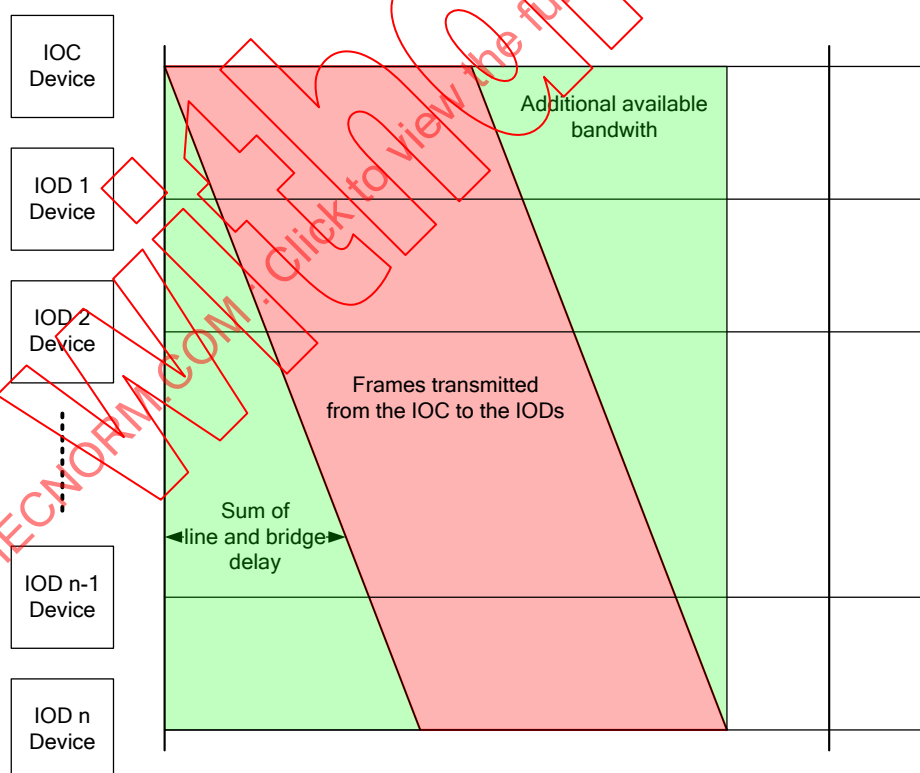


Figure K.2 – Propagation of frames in linear transmit direction

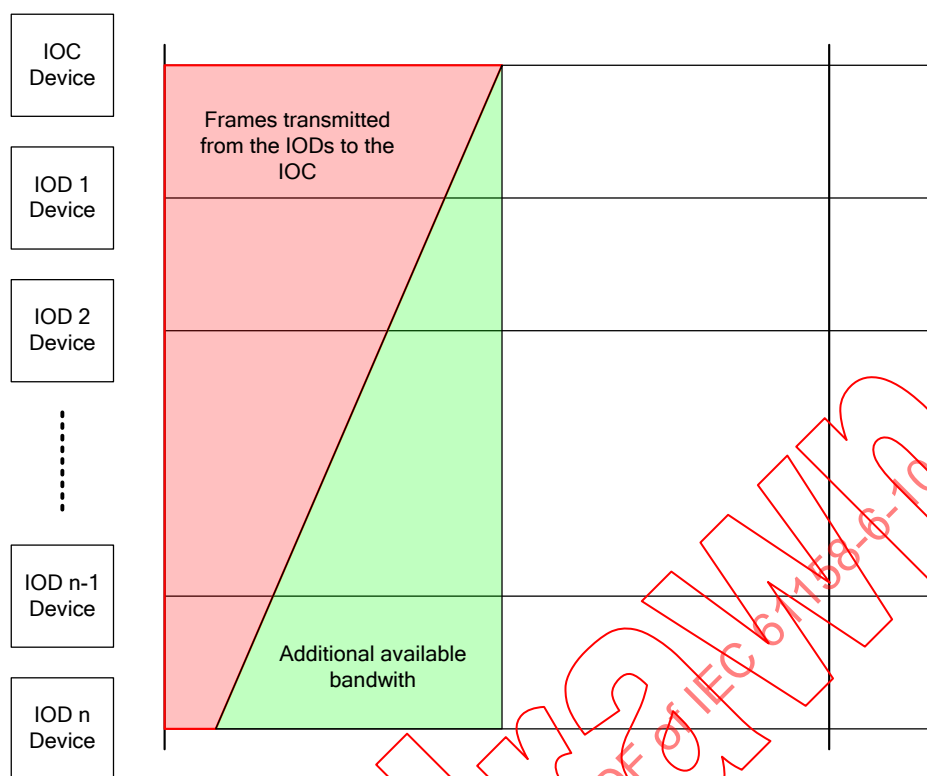


Figure K.3 – Propagation of a frames in receive direction

Annex L (informative)

Time constraints for bandwidth allocation

Figure L.1 illustrates the association between the begin and end of the RED phase of the transmitting and the begin and end of the RED phase of the receiving port.

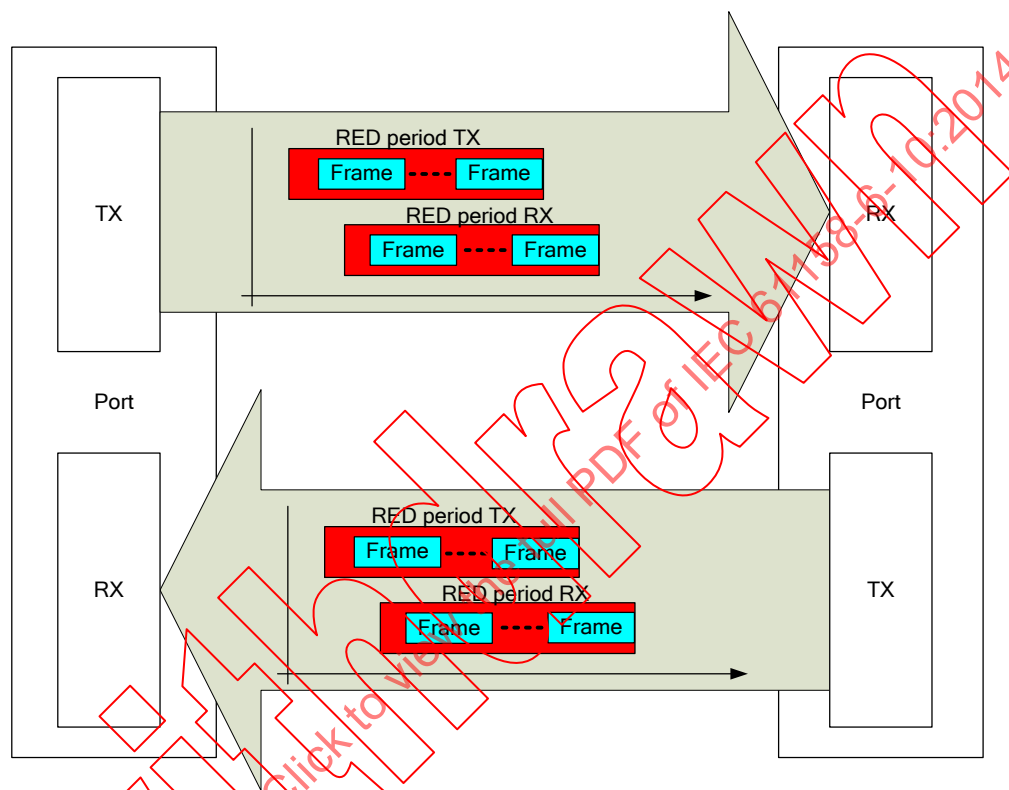


Figure L.1 – Overview of time constraints for bandwidth allocation

Figure L.2 illustrates the calculation of a RED period and Figure L.3 illustrates the calculation of a GREEN period for a port. This calculation uses the engineered payload, the MinSupportedFSO and the REDEndSafetyMargin for the length of RED period.

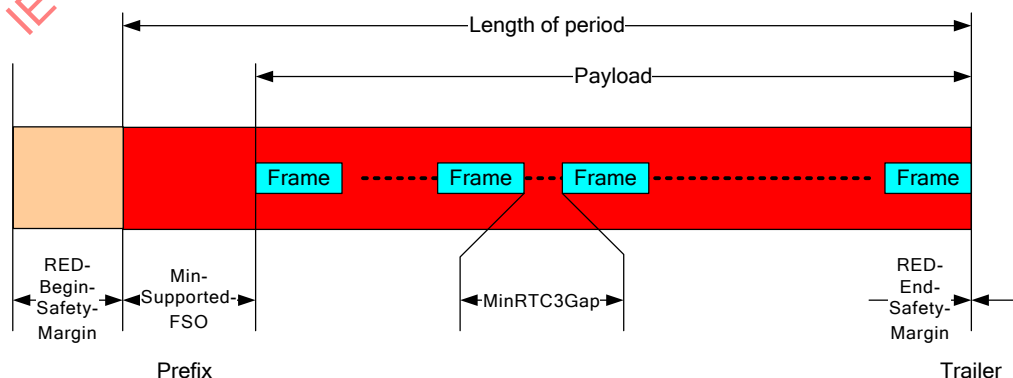


Figure L.2 – Calculation of the length of a RED period

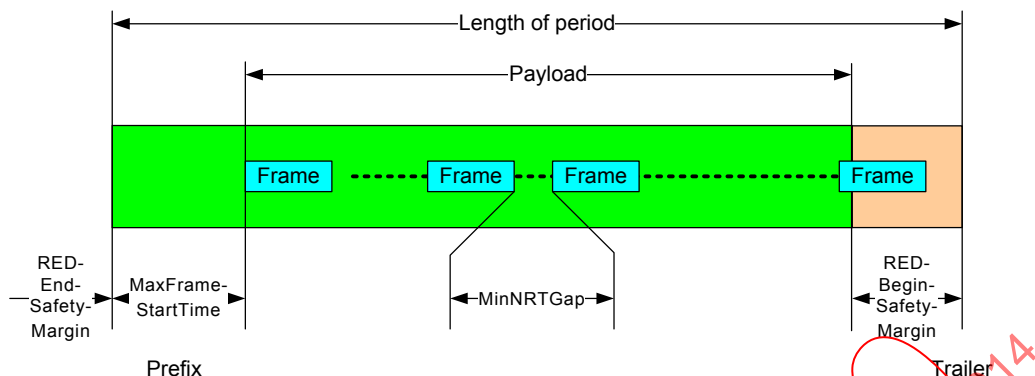


Figure L.3 – Calculation of the length of a GREEN period

Formula (L.1) shows the calculation of a RED period for a port.

$$\text{Length of period} = \text{MinSupportedFSO} + \text{payload} + \text{REDEndSafetyMargin} \quad (\text{L.1})$$

where

MinSupportedFSO	is the time the period shall start before the first frame is sent or expected to be received
payload	is the time which is necessary to transfer all frames of this period
REDEndSafetyMargin	is the time the period shall be prolonged after the last frame is sent or expected to be received

Formula (L.2) and (L.3) shows the calculation of the start of period for a port in principle. Both, TX start of period and RX start of period shall never be negative.

$$\text{TX start of period} = 0 \quad (\text{L.2})$$

$$\text{RX start of period} = 0 \quad (\text{L.3})$$

The engineering shall use the LineDelay (or the MaxLineDelay) by planning the FrameSendOffset. Because the LineDelay and the jitter of synchronization is always less than the MinSupportedFSO, every frame will be transmitted and received inside the calculated period.

The YellowTime, always active before RED, ensures that in the above defined environment, only RT_CLASS_3 frames are transmitted and received in the RED.

Annex M (informative)

Time constraints for the forwarding of a frame

The forwarding of a frame in a network depends on the bridge and the line delay. The line delay cumulates the cable and the port delay.

The minimum bridge delay (see Figure 27) depends on the needed octets for the forwarding decision. For this standard the reception of a frame until the FrameID is necessary.

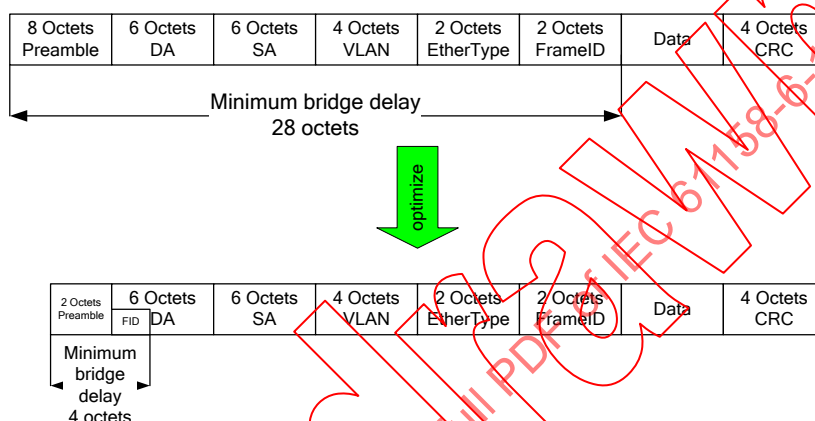


Figure M.1 – Minimization of bridge delay

Figure M.1 shows the optimization. The first step optimize the bridge delay to the minimum is a hardware that add no additional delay to the theoretical minimum Ethernet bridge delay. The second step uses a different strategy to make the forwarding decision by means of a different coding of the DA field. The third step reduces the received octets without a change of the required octets for the forwarding decision.

An optimization of the line delay (see Figure 27) is only possible by a reduction of the PortRxDelay and PortTxDelay values, which depends on the MAU.

Annex N (informative)

Principle of dynamic frame packing

The protocol vs. data ratio for IO devices with a small amount of data could be optimized using dynamic frame packing.

The dynamic frame packing may be used for input and output data as shown in Figure N.1, Figure N.2 and Figure N.3.

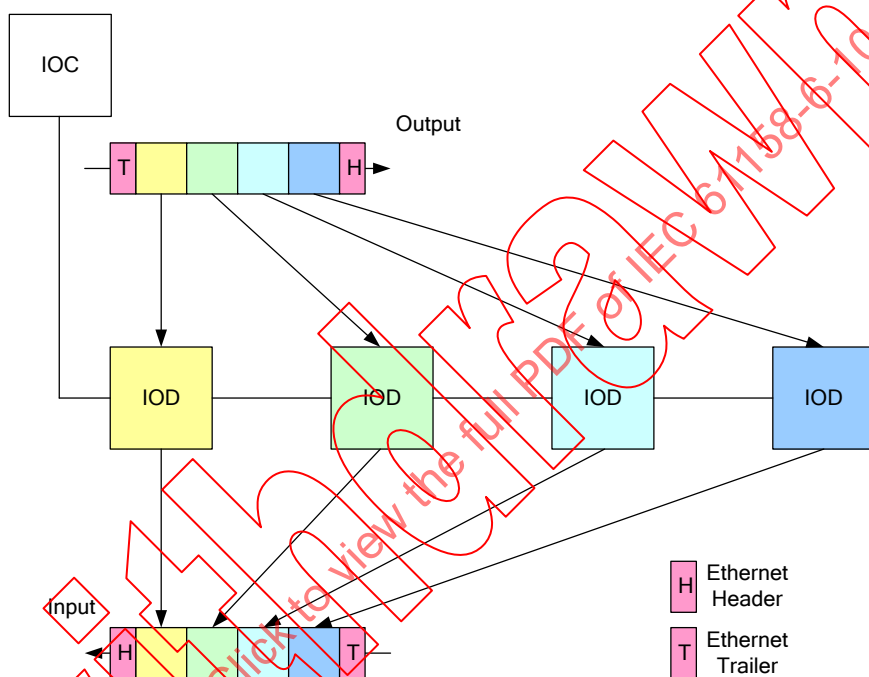


Figure N.1 – Dynamic frame packing

As an additional optimization the output frame (Provider from the IO controller point of view) shall be truncated as shown in Figure N.2.

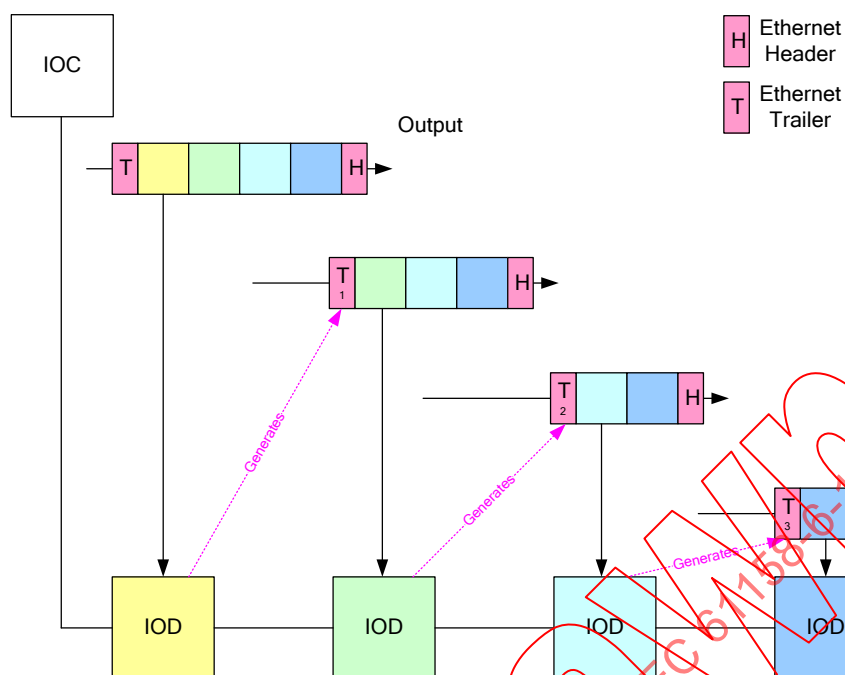


Figure N.2 – Dynamic frame packing – truncation of outputs

The input frame (Consumer from the IO controller point of view) shall be concatenated as shown in Figure N.3 for optimization.

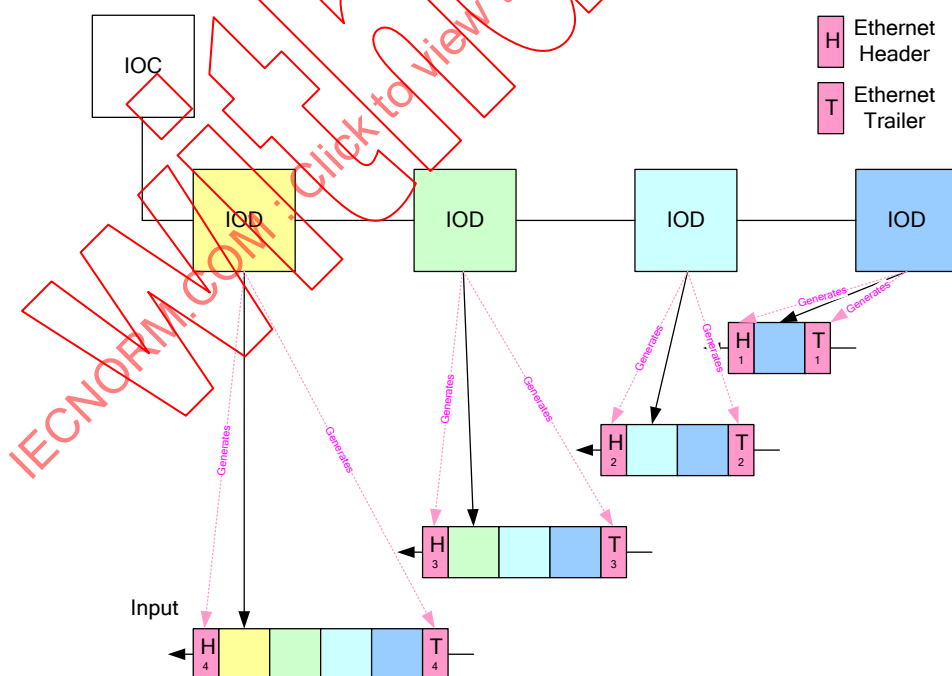


Figure N.3 – Dynamic frame packing – concatenation of inputs

The model of dynamic frame packing offers two variants. The first, the “pack on transport” variant is shown before. The second, the end node variant is shown in Figure N.4. In this case, all Subframes are generated by one IOC and one IOD.

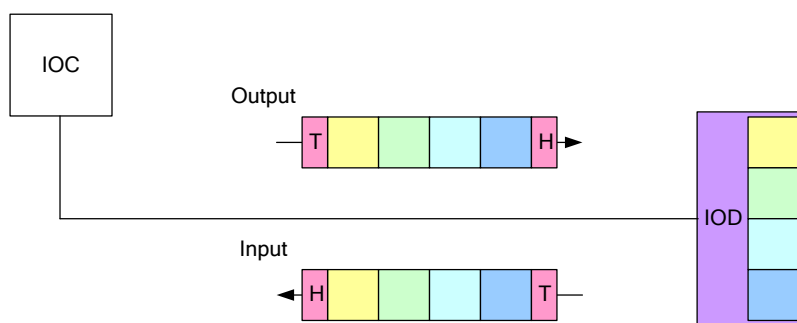


Figure N.4 – End node mode

The concatenation of inputs could not be done in zero time. Thus, an internal delay called DFPFeed exists. The parameter MaxDFPFeed is used to calculate the delay (DFPFeed) of the implementation, as shown in Figure Figure N.5.

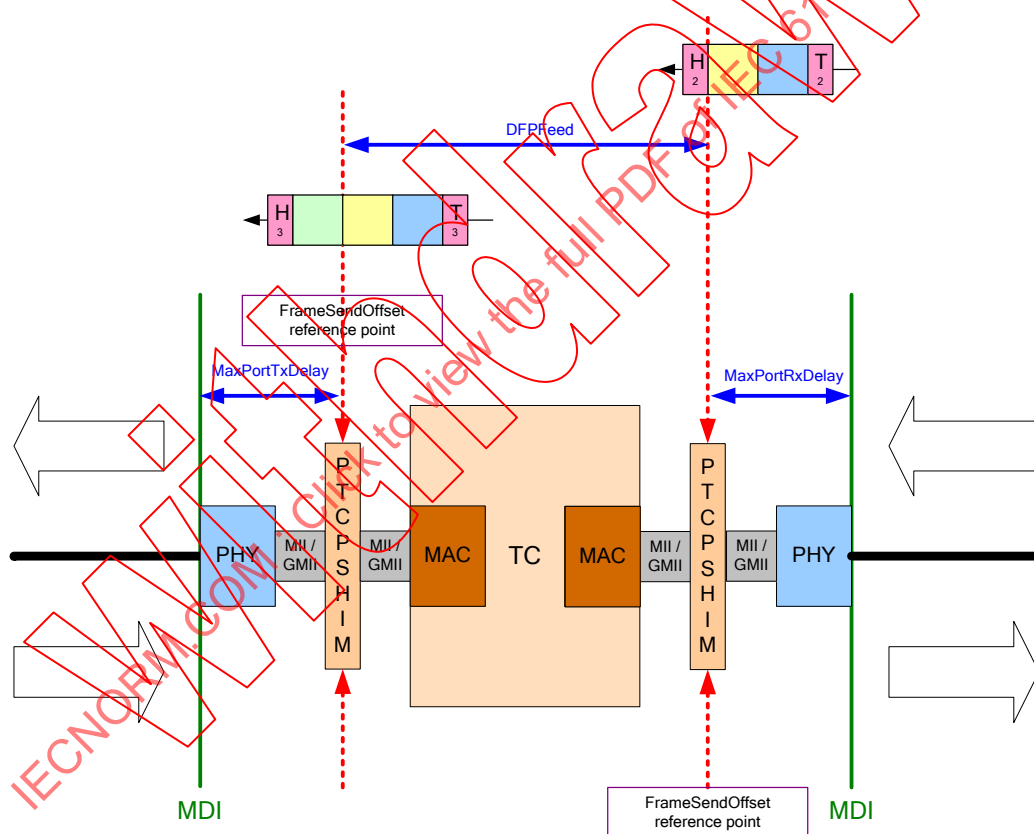


Figure N.5 – DFPFeed definition

The calculation of the DFPFeed shall be done by Formula (N.1). The engineering parameter MaxDFPFeed shall cover the maximum value of an implementation.

$$\text{DFPFeed} = \text{FrameSendOffset}(\text{Subframe}, \text{TxPort}) - \text{ReceiveTimeStamp}(\text{Subframe}, \text{RxPort}) \quad (\text{N.1})$$

where

DFFFeed	is the internal delay to concatenate a subframe to another
FrameSendOffset(Subframe, TxPort)	is the point in time when the subframe shall be transmitted
ReceiveTimeStamp(Subframe, RxPort)	is the point in time when the subframe is received

Annex O (informative)

Principle of Fragmentation

The concept of peer to peer fragmentation offers the possibility to reduce the SendClock to 31,25 μ s or to enhance the RED bandwidth for a SendClock over 50 %.

Example The RED period for a SendClock of 1 ms may be 900 μ s and the GREEN period 100 μ s.

Figure O.1 shows the principle of the fragmentation and Figure O.2 the protocol elements of fragmentation as an example for one frame.

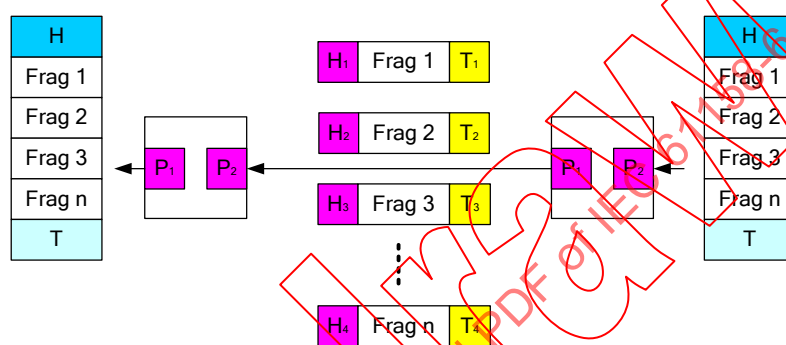


Figure O.1 – Principle of fragmentation

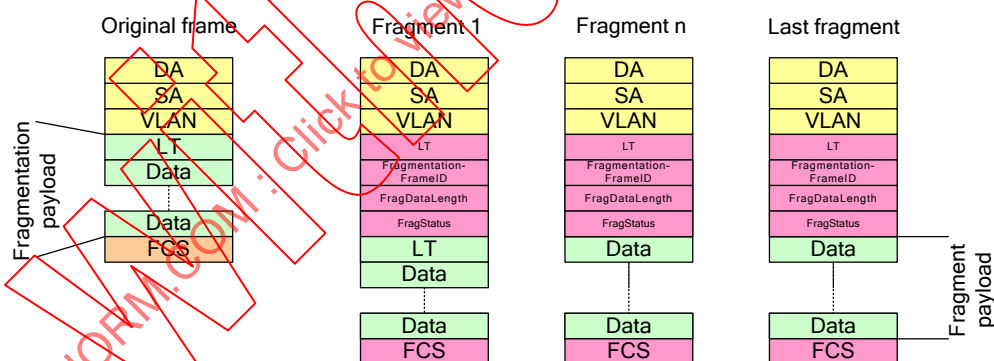


Figure O.2 – Protocol elements of fragments

The generation of fragments shall ensure that a fragment is small enough to be transmitted during GREEN and big enough to make a good usage of the bandwidth. In best case the transmitter generates fragments which are as long as the residual GREEN part of the period. The performance in this case is equivalent to 250 μ s length of period.

Figure O.3 shows the principle of bandwidth allocation when using fragmentation. In this case the bandwidth is parted into a RT_CLASS_3 period, a RT_CLASS_2 / _1 communication period and a NRT communication period.



The engineering shall check the needed bandwidth for all periods and also keep in mind, that RT_CLASS_2 / _1 frames are never fragmented and that this kind of frames are forwarded in the cut-through mode.

The RT_CLASS_2 / _1 communication period and NRT communication period may be combined for cases which need a minimum SendClockFactor. In these cases, the principle of the ReductionRatio is used to guarantee bandwidth for the NRT communication.

Figure O.4 shows the combination of a fragmentation and a non fragmentation domain. The rules for the forwarding of RT_CLASS_2 / _1 frames and the protection of the RED period makes a fragmentation domain delicate for unexpected too long RT_CLASS_2 / _1 frames.



Thus, a guardian which checks whether the to be forwarded RT_CLASS_2 / _1 frames are valid for the fragmentation domain is useful. Only RT_CLASS_2 / _1 frames which comply to the Yellow period rule should be forwarded.

IECNORM.COM : Click to view the full PDF of IEC 61158-6-10:2014

Withdrawing

Annex P (informative)

MRPD – Principle of seamless media redundancy

The concept seamless media redundancy offers a reliable network infrastructure in case of fault. Figure P.1, Figure P.2 and Figure P.3 shows the concept in principle.

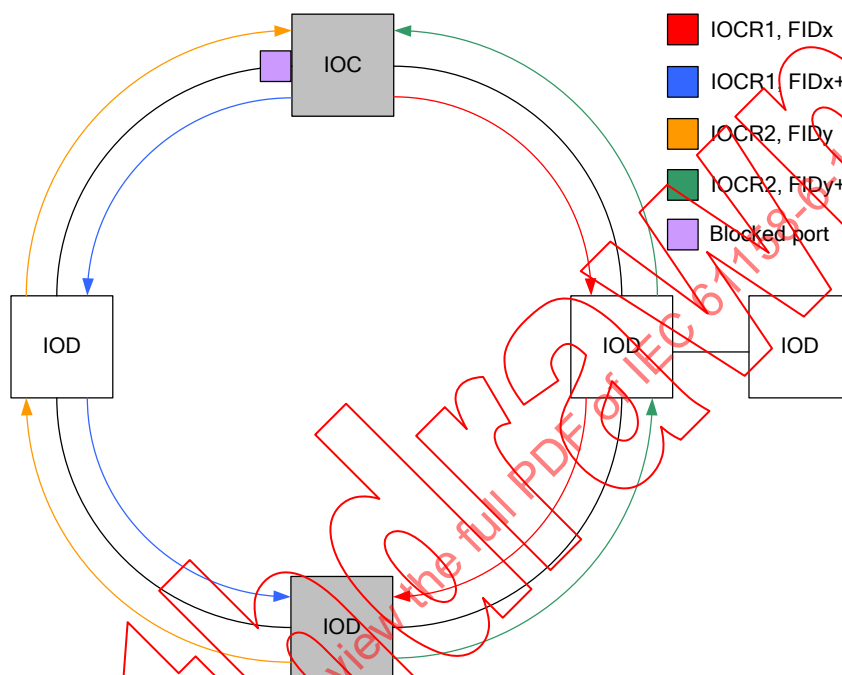


Figure P.1 – Principle of seamless media redundancy – I/OCR

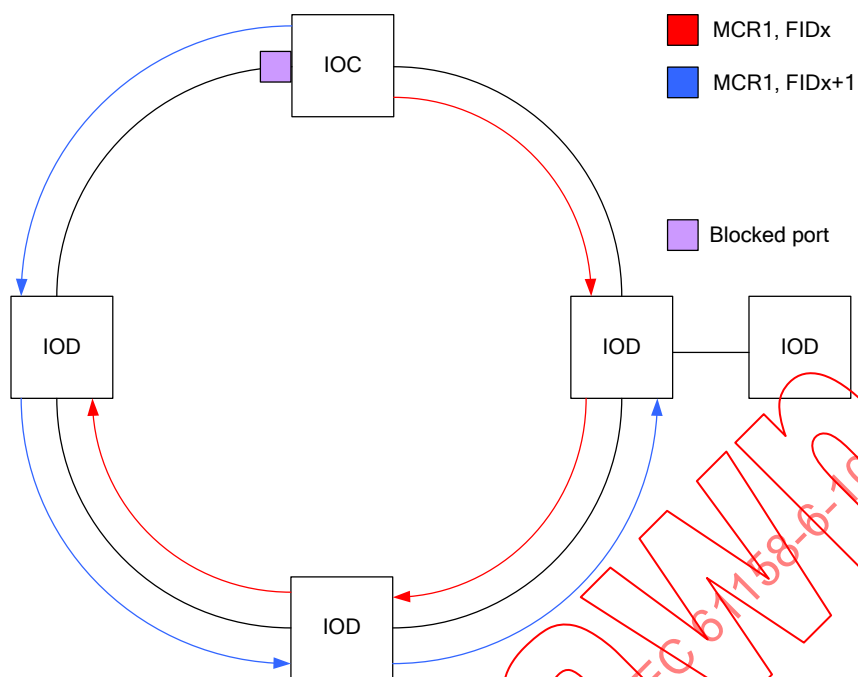


Figure P.2 – Principle of seamless media redundancy – MCR

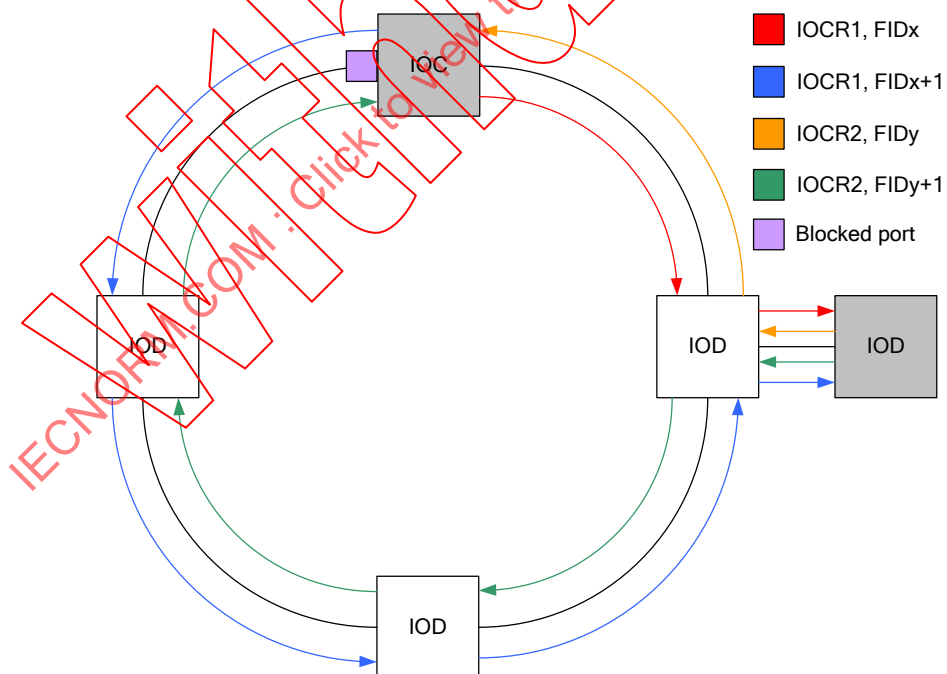


Figure P.3 – Principle of seamless media redundancy – Line

Annex Q (normative)

Principle of a RED_RELAY without forwarding information in PDIRFrameData

The RED_RELAY needs information about the forwarding port and the sending time of a frame. This information is given by the PDIRFrameData. As an optimization the necessary information for the RED_RELAY may be calculated using the PDIRGlobalData. Figure Q.1 shows the concept.

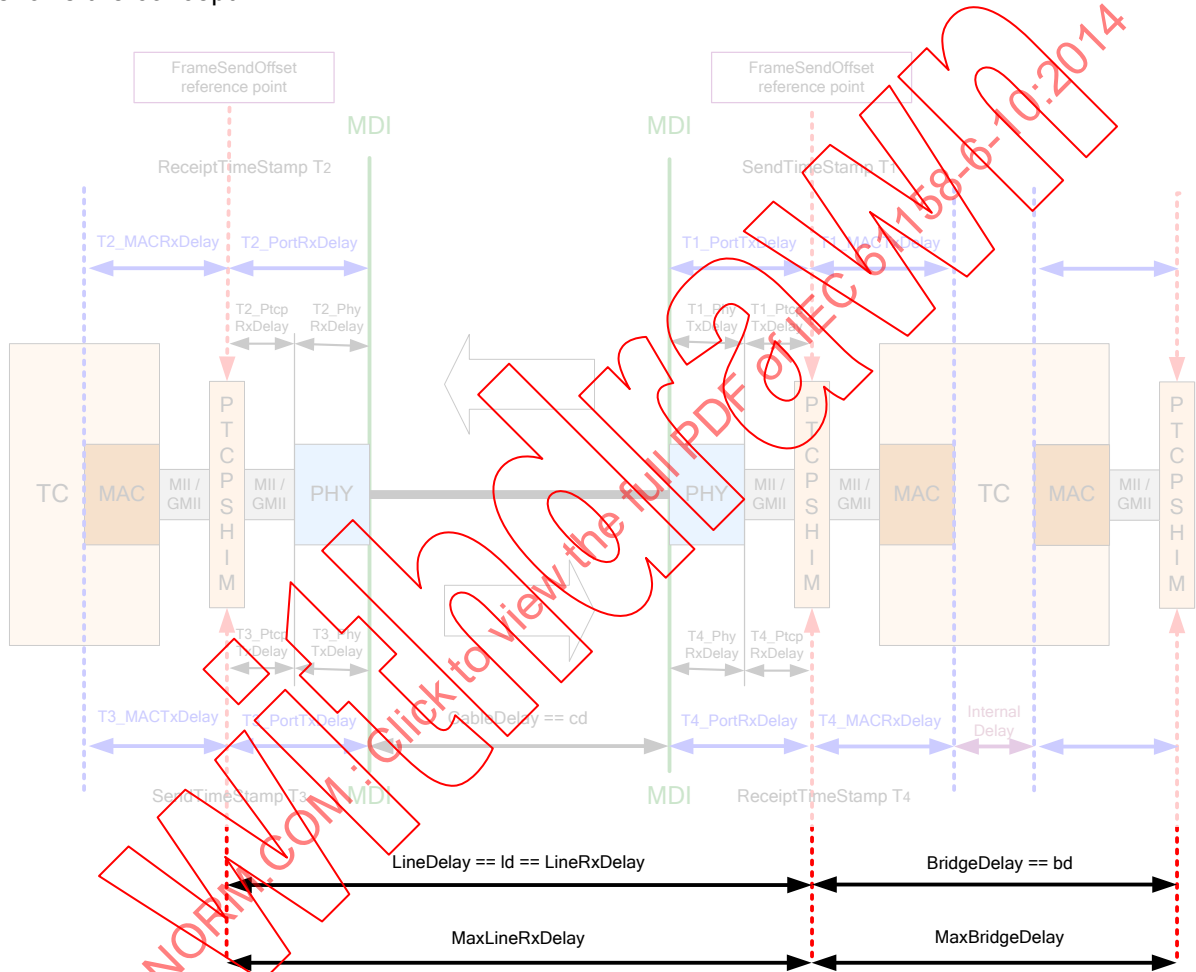


Figure Q.1 – Generating the FrameSendOffset for a RED_RELAY without forwarding information in PDIRFrameData

NOTE Figure Q.1 is formatted partial as transparent to highlight the important part for this annex.

The calculation of the required FrameSendOffset for each frame shall be done by Formula (Q.1).

$$\begin{aligned} \text{FrameSendOffset}(\text{Frame}, \text{TxPort}) = & \text{ReceiveTimeStamp}(\text{Frame}, \text{RxPort}) \\ & + (\text{PDIRGlobalData.MaxLineRxDelay} - \text{LineRxDelay}) \\ & + (\text{PDIRGlobalData.MaxBridgeDelay}) \\ & - \text{SafetyMargin} \end{aligned} \quad (\text{Q.1})$$

where

FrameSendOffset(Frame, TxPort)	is the point in time when the frame shall be transmitted
ReceiveTimeStamp(Frame, RxPort)	is the point in time when the frame is received
PDIRGlobalData.MaxLineRxDelay	is the time which is used by the engineering as LineDelay
LineRxDelay	is the time which is measured by the node
PDIRGlobalData.MaxBridgeDelay	is the time which is used by the engineering as BridgeDelay.
	The BridgeDelay given by the node shall be less than MaxBridgeDelay.
SafetyMargin	is the time which covers measurement errors (e.g. 100 ns)

For nodes with two ports an assumption for the forwarding port is done. For nodes with more than two ports additional information, at least which TxPort shall be used, is needed.

The calculation of the RetentionTime for each frame shall be done by Formula (Q.2).

$$\text{FrameSendOffset}(\text{Frame}, \text{TxPort}) = \text{ReceiveTimeStamp}(\text{Frame}, \text{RxPort}) + \text{RetentionTime}(\text{Frame}) \quad (\text{Q.2})$$

where

FrameSendOffset(Frame, TxPort)	is the calculated point in time when the frame shall be transmitted
ReceiveTimeStamp(Frame, RxPort)	is the point in time when the frame is received
RetentionTime(Frame)	is the time where a frame is hold up in the bridge

The calculation of the required RetentionTime and MaxRetentionTime for each frame shall be done by Formula (Q.3) and (Q.4).

$$\begin{aligned} \text{RetentionTime}(\text{Frame}) &= (\text{MaxLineRxDelay}(T_{n+1}) - \text{RealLineRxDelay}(T_{n+1})) + (\text{MaxBridgeDelay}(T_{n+1}) - \text{RealBridgeDelay}(T_{n+1})) \\ &= (\text{MaxPortTxDelay}(T_n) - \text{RealPortTxDelay}(T_n)) + (\text{MaxCableDelay} - \text{RealCableDelay}) + (\text{MaxPortRxDelay}(T_{n+1}) - \text{RealPortRxDelay}(T_{n+1})) + (\text{MaxBridgeDelay}(T_{n+1}) - \text{RealBridgeDelay}(T_{n+1})) \end{aligned} \quad (\text{Q.3})$$

$$\text{MaxRetentionTime}(\text{Frame}) = (\text{MaxLineRxDelay}(T_{n+1}) - 0) + (\text{MaxBridgeDelay}(T_{n+1}) - \text{RealBridgeDelay}(T_{n+1})) \quad (\text{Q.4})$$

where

RetentionTime(Frame)	is the difference between the engineered and the real value of the line delay and bridge delay which shall be compensated by the node
MaxRetentionTime(Frame)	is the maximum difference between the engineered and the real value of the line delay and bridge delay which shall be compensated by the node
MaxLineRxDelay	is the engineered value of the line delay
RealLineRxDelay	is the real value of the line delay
MaxBridgeDelay	is the engineered value of the bridge delay
RealBridgeDelay	is the real value of the bridge delay
MaxPortTxDelay	is the engineered delay of the port transmitting a frame
RealPortTxDelay	is the real delay of the port transmitting a frame
MaxPortRxDelay	is the engineered delay of the port receiving a frame
RealPortRxDelay	is the real delay of the port receiving a frame
MaxCableDelay	is the engineered delay of the cable
RealCableDelay	is the real delay of the cable
T_n	is the node transmitting the frame
T_{n+1}	is the node receiving the frame and ensuring the Retention Time for the transmitting of the frame

Annex R (informative)

Optimization for fast startup without autonegotiation

To increase the startup time of an Ethernet connection the auto negotiation time shall be avoided. To do this, the wiring of one port shall have the required crossover to avoid the usage of crossover cables, as shown in Figure R.1.

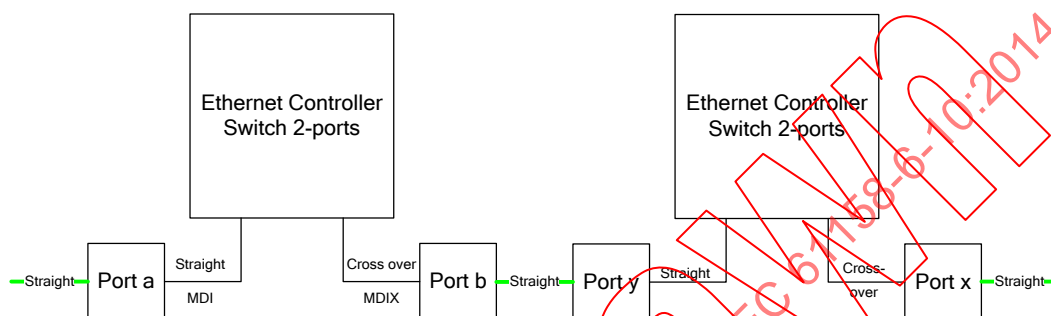


Figure R.1 – Scheme of a 2-port switch

NOTE 1 The MDI interface is intended to be used by end stations and routers (single port per interface devices).

NOTE 2 The MDIX interface is intended to be used by hubs and switches (multiple port per interface devices) to enable the usage of straight-through cables for end stations.

NOTE 3 Up to date components use auto MDI/MDIX (autonegotiation) to automatically switch to the proper configuration once a cable is connected.

NOTE 4 Nodes with more than two ports may provide one straight and many cross over ports.

The behavior of the possible combinations of PHY / MAC modes is shown in Figure R.2 and Table R.1

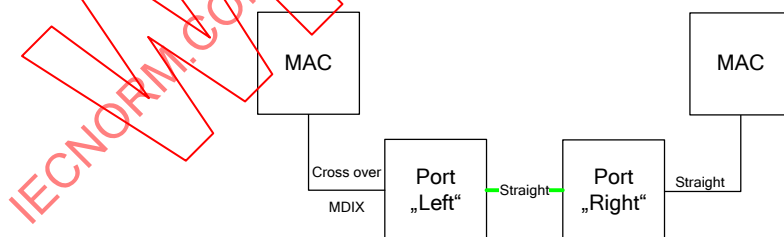


Figure R.2 – Scheme of 2-ports

Table R.1 – Truth table

Parameterized mode of Port “Left”	Negotiate mode of Port “Left”	Negotiate mode of Port “Right”	Parameterized mode of Port “Right”
AutoNegotiation	100BASETXFD	100BASETXFD	AutoNegotiation
100BASETXFD	100BASETXFD	100BASETXHD → 100BASETXFD ^a	AutoNegotiation
100BASETXFD	100BASETXFD	100BASETXFD ^b	AutoNegotiation
100BASETXFD	100BASETXFD	100BASETXFD	100BASETXFD
^a This standard requires at least 100 Mbit/s. If the negotiation process signals 100BASETXHD the application should change it to 100BASETXFD.			
^b A PHY / MAC combination may optimize the negotiation for this standard.			

Annex S (informative)

TX-error handling

The IEEE 802 offers the possibility to encode errors during the transmission of a frame in the MAC. The recommended coding is shown in Table S.1.

Table S.1 – TX-error

Error-Code (hexadecimal)	Meaning
0x0	Transmission aborted due to receive error (RX_ERR := '1')
0x1	Transmission aborted due to receive error (frame FCS error)
0x2	Transmission aborted due to receive error (frame too long)
0x3	Transmission aborted due to receive error (frame too short)
0x4	Transmission aborted due to receive error (subframe CRC16 error)
0x5	Transmission aborted due to receive error (unexpected end of a DFP frame)
0x6	Transmission aborted due to receive error (unexpected subframe of a DFP frame)
0x7	Transmission aborted due to resource error
0x8	Transmission aborted due to MAC error
0x9	Transmission aborted due to SA := own MAC address error
0xA	Transmission aborted due to disable port or disable MAC
0xB	Transmission aborted due to unexpected structure of a DFP frame
0xC	Transmission aborted due to unexpected structure of a PTCP frame
0xD	Reserved
0xE	Reserved
0xF	Reserved

Annex T (informative)

Example of a PrmBegin, PrmEnd and ApplRdy sequence

This standard introduces the PrmBegin, PrmEnd and ApplRdy procedure as a basic feature for system redundancy and configure in run.

Figure T.1 shows an example of this procedure.

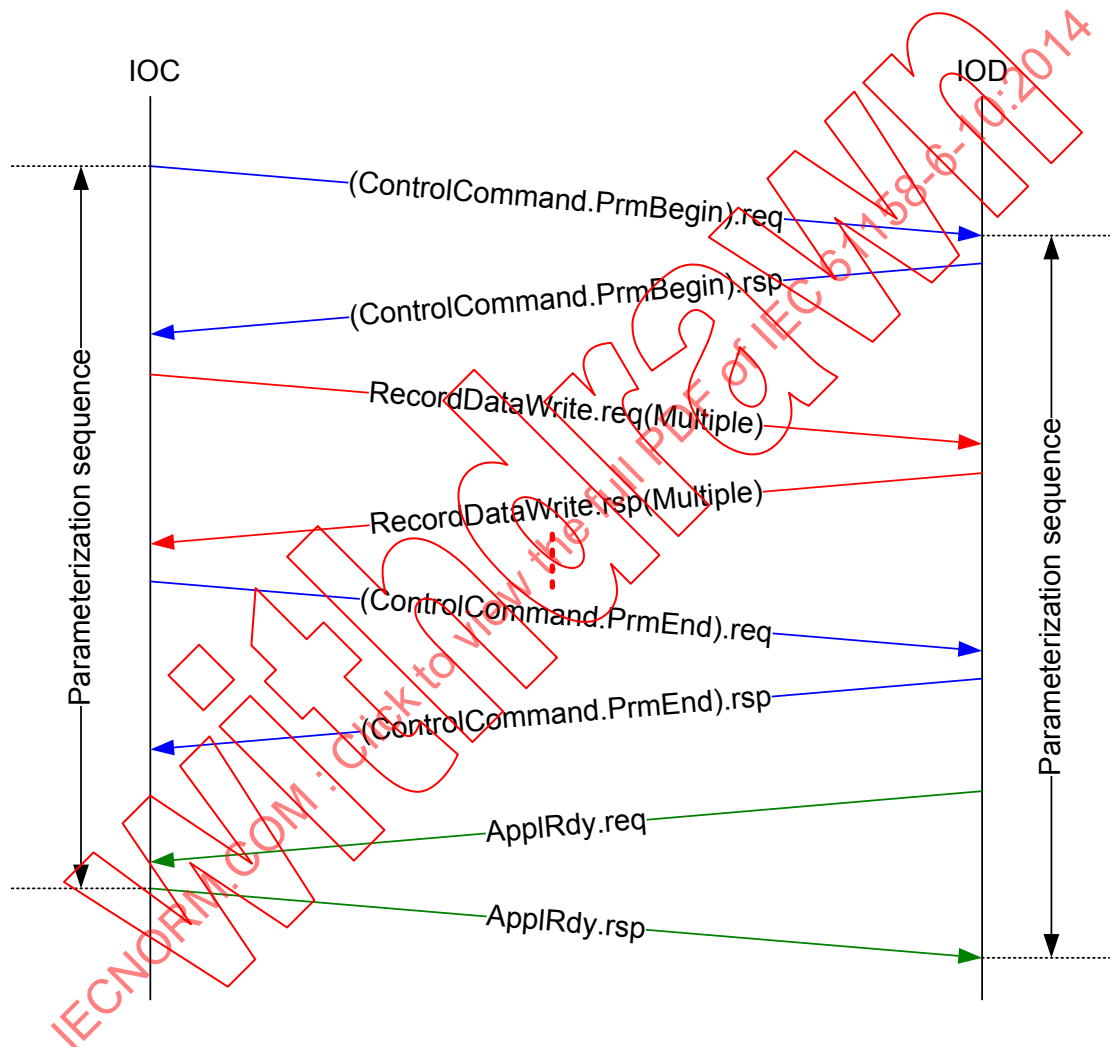


Figure T.1 – PrmBegin, PrmEnd and ApplRdy procedure

Annex U (informative)

List of supported MIBs

This standard defines a list of supported MIBs for the devices. Table U.1 shows the defined or referenced MIBs.

Table U.1 – List of supported MIBs

Name	Meaning
MIB II	MIB from IETF RFC 1213
Interface Group MIB	MIB from IETF RFC 2863 NOTE This MIB supports interface and port mapping. It is used by newer MIBs to provide an overall view to the interfaces and ports.
LLDP-MIB	MIB from IEEE 802.1AB
LLDP-EXT-DOT1-MIB	Extension of the IEEE 802.1AB MIB by IEEE 802.1
LLDP-EXT-DOT3-MIB	Extension of the IEEE 802.1AB MIB by IEEE 802.3
LLDP-EXT-PNIO-MIB	Extension of the IEEE 802.1AB MIB by this standard
PI-MIB	MIB from this standard
IEEE8021-AS-MIB	MIB from IEEE 802.1AS
MRP-MIB	MIB from IEC 62439-2

Annex V

(informative)

Structure and content of BLOB

The BLOB used for the GSD contains the binary representation of the result of a Lempel-Ziv-Markow-Algorithm compression of a directory structure shown in Table V.1.

The compression should be done by using i.e. the program 7zip. The BLOB is the binary content of the created archive file.

Table V.1 – Content of archive

Directory	Meaning
Icons	Containing the icons used for the GSD. May be empty or non existent.
Interface	Containing the interface specification for the provided controller to controller communication interface. May be empty or non existent.
GSD	Contains GSD file.
Other	Additional information not fitting into the other directories. May be empty or non existent.

Annex W (normative)

LLDP EXT MIB

This standard defines an extension to the IEEE 802.1AB-2005 MIB which is shown below.

LLDP-EXT-PNO-MIB DEFINITIONS ::= BEGIN

IMPORTS

SnmpAdminString FROM SNMP-FRAMEWORK-MIB

Unsigned32,
MODULE-IDENTITY,
OBJECT-TYPE FROM SNMPv2-SMI

MacAddress,
DisplayString,
TruthValue FROM SNMPv2-TC

MODULE-COMPLIANCE,
OBJECT-GROUP FROM SNMPv2-CONF

lldpExtensions,
lldpRemTimeMark,
lldpRemLocalPortNum,
lldpRemIndex,
lldpPortConfigEntry,
lldpLocPortNum FROM LLDP-MIB;

lldpXPnoMIB

MODULE-IDENTITY
LAST-UPDATED "201003010000Z" -- March 1, 2010
ORGANIZATION "PROFIBUS & PROFINET INTERNATIONAL (PI)"
CONTACT-INFO
URL: <http://www.profibus.com>
email: info@profibus.com

Postal: Haid-und-Neu-Strasse 7
D-76131 Karlsruhe

Tel: ++49 721 9658 - 590

DESCRIPTION

module for

information.

The LLDP Management Information base extension

PROFINET organizationally defined discovery

(2005).

Copyright © PROFIBUS Nutzerorganisation e.V.

REVISION

DESCRIPTION

"200602280000Z" - February 28, 2006

"added MRP UUID information, port status values and
naming of tables changed"

REVISION

DESCRIPTION

"200508310000Z" - August 31, 2005

"added RT port status"

REVISION

DESCRIPTION

"200505300000Z" - May 30, 2005

"initial version"

OUI for PNO 3791 (00-0E-CF)

::= { lldpExtensions 3791 }

Organizationally Defined Information Extension - PNO

lldpXPnoObjects

OBJECT IDENTIFIER ::= { lldpXPnoMIB 1 }

LLDP PNO extension MIB groups

lldpXPnoConfig

OBJECT IDENTIFIER ::= { lldpXPnoObjects 1 }

lldpXPnoLocalData

OBJECT IDENTIFIER ::= { lldpXPnoObjects 2 }

lldpXPnoRemoteData

OBJECT IDENTIFIER ::= { lldpXPnoObjects 3 }

PNO - Configuration

lldpXPnoConfigTable

OBJECT-TYPE

SYNTAX

SEQUENCE OF LldpXPnoConfigEntry

MAX-ACCESS

not-accessible

STATUS	current
DESCRIPTION	" A table that controls selection of LLDP TLVs to be transmitted on individual ports. "
::= { lldpXPnoConfig 1 }	
lldpXPnoConfigEntry	OBJECT-TYPE
SYNTAX	LldpXPnoConfigEntry
MAX-ACCESS	not-accessible
STATUS	current
DESCRIPTION	" LLDP configuration information that controls the transmission of PNO organizationally defined TLVs on LLDP transmission capable ports. This configuration object augments the lldpPortConfigEntry of the LLDP-MIB, therefore it is only present along with the port configuration defined by the associated lldpPortConfigEntry entry. Each active lldpXPnoConfigEntry must be restored from non-volatile storage (along with the corresponding lldpPortConfigEntry) after a re-initialization of the management system. "
AUGMENTS	{ lldpPortConfigEntry }
::= { lldpXPnoConfigTable 1 }	
LldpXPnoConfigEntry ::=	SEQUENCE {
lldpXPnoConfigSPDtxEnable	TruthValue,
lldpXPnoConfigPortStatusTxEnable	TruthValue,
lldpXPnoConfigAliasTxEnable	TruthValue,
lldpXPnoConfigMrpTxEnable	TruthValue,
lldpXPnoConfigPtcpTxEnable	TruthValue
}	}
lldpXPnoConfigSPDtxEnable	OBJECT-TYPE
SYNTAX	TruthValue
MAX-ACCESS	read-write
STATUS	current
DESCRIPTION	" The lldpXPnoConfigSPDtxEnable, which is defined as a truth value and configured by the network management, determines whether the PNO organizationally defined signal propagation delay TLV transmission is allowed on a given LLDP transmission capable port. The signal propagation delay is composed of the transmission delay, the port receiving delay and the line delay. These values can't be transmitted independently of each other. The value of this object must be restored from non-volatile storage after a re-initialization of the management system. "
DEFVAL	{ true }
::= { lldpXPnoConfigEntry 1 }	
lldpXPnoConfigPortStatusTxEnable	OBJECT-TYPE
SYNTAX	TruthValue
MAX-ACCESS	read-write
STATUS	current
DESCRIPTION	" The lldpXPnoConfigPortStatusTxEnable, which is defined as a truth value and configured by the network management,

	determines whether the PNO organizationally defined RT port status TLV transmission is allowed on a
given	LLDP transmission capable port.
volatile	The value of this object must be restored from non-
system.	storage after a re-initialization of the management
DEFVAL	"
::= { lldpXPnoConfigEntry 2 }	{ true }
lldpXPnoConfigAliasTxEnable	OBJECT-TYPE
SYNTAX	TruthValue
MAX-ACCESS	read-write
STATUS	current
DESCRIPTION	"
as	The lldpXPnoConfigAliasTxEnable, which is defined
management,	a truth value and configured by the network
given	determines whether the PNO organizationally defined alias TLV (chassisId) transmission is allowed on a
	LLDP transmission capable port.
volatile	The value of this object must be restored from non-
system.	storage after a re-initialization of the management
DEFVAL	"
::= { lldpXPnoConfigEntry 3 }	{ true }
lldpXPnoConfigMrpTxEnable	OBJECT-TYPE
SYNTAX	TruthValue
MAX-ACCESS	read-write
STATUS	current
DESCRIPTION	"
management,	The lldpXPnoConfigMrpTxEnable, which is defined as
	a truth value and configured by the network
	determines whether the PNO organizationally defined MRP TLV transmission is allowed on a given
	LLDP transmission capable port.
volatile	The value of this object must be restored from non-
system.	storage after a re-initialization of the management
DEFVAL	"
::= { lldpXPnoConfigEntry 4 }	{ true }
lldpXPnoConfigPtcpTxEnable	OBJECT-TYPE
SYNTAX	TruthValue
MAX-ACCESS	read-write
STATUS	current
DESCRIPTION	"
management,	The lldpXPnoConfigPtcpTxEnable, which is defined as
	a truth value and configured by the network
	determines whether the PNO organizationally defined PTCP TLV transmission is allowed on a given
	LLDP transmission capable port.
volatile	The value of this object must be restored from non-
system.	storage after a re-initialization of the management
DEFVAL	"
::= { lldpXPnoConfigEntry 5 }	{ true }
PNO - Local System Information	
lldpXPnoLocTable	OBJECT-TYPE

SYNTAX	SEQUENCE OF LldpXPnoLocEntry
MAX-ACCESS	not-accessible
STATUS	current
DESCRIPTION	" This table contains one row per port for PNO organizationally defined LLDP extension on the local system known to this agent. "
::= { lldpXPnoLocalData 1 }	
lldpXPnoLocEntry	OBJECT-TYPE
SYNTAX	LldpXPnoLocEntry
MAX-ACCESS	not-accessible
STATUS	current
DESCRIPTION	" Additional information about a particular port component. This object is indexed by the lldpLocPortNum of the LLDP-MIB, therefore it is only present along with the port entry defined by the associated lldpLocPortEntry entry. Each active lldpXPnoLocEntry must be restored from non-volatile storage (along with the corresponding lldpLocPortEntry) after a re-initialization of the management system. " { lldpLocPortNum }
INDEX	{ lldpLocPortNum }
::= { lldpXPnoLocTable 1 }	
LldpXPnoLocEntry ::=	SEQUENCE {
lldpXPnoLocLPDValue	Unsigned32,
lldpXPnoLocPortTxDValue	Unsigned32,
lldpXPnoLocPortRxDValue	Unsigned32,
lldpXPnoLocPortStatusRT2	INTEGER,
lldpXPnoLocPortStatusRT3	INTEGER,
lldpXPnoLocPortNoS	DisplayString,
lldpXPnoLocPortMrpUnId	OCTET STRING,
lldpXPnoLocPortMrrtStatus	INTEGER,
lldpXPnoLocPortPtcpMaster	MacAddress,
lldpXPnoLocPortPtcpSubdomainUNID	OCTET STRING,
lldpXPnoLocPortPtcpIRDataUNID	OCTET STRING,
lldpXPnoLocPortModERT3	INTEGER,
lldpXPnoLocPortPeriodLength	Unsigned32,
lldpXPnoLocPortPeriodValidity	TruthValue,
lldpXPnoLocPortRedOffset	Unsigned32,
lldpXPnoLocPortRedValidity	TruthValue,
lldpXPnoLocPortOrangeOffset	Unsigned32,
lldpXPnoLocPortOrangeValidity	TruthValue,
lldpXPnoLocPortGreenOffset	Unsigned32,
lldpXPnoLocPortGreenValidity	TruthValue,
lldpXPnoLocPortStatusRT3OptimizationSupported	TruthValue,
lldpXPnoLocPortStatusRT3PreambleShorteningSupported	TruthValue,
lldpXPnoLocPortStatusRT3PreambleShortening	TruthValue,
lldpXPnoLocPortStatusRT3FragmentationSupported	TruthValue,
lldpXPnoLocPortStatusRT3Fragmentation	TruthValue
}	
lldpXPnoLocLPDValue	OBJECT-TYPE
SYNTAX	Unsigned32
UNITS	"ns"
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	" This integer value represents the line propagation delay in nanoseconds which was measured by the local system on the corresponding port. A value of zero shall be used if the system either could not accomplish the measurement or does not support such a measurement. " { 0 }
DEFVAL	{ 0 }
::= { lldpXPnoLocEntry 1 }	

lldpXPnoLocPortTxDValue	OBJECT-TYPE
SYNTAX	Unsigned32
UNITS	"ns"
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	" This integer value represents the PortTxDelay in nanoseconds which was measured by the local system on the corresponding port. A value of zero shall be used if the system either could not accomplish the measurement or does not support such a measurement." "
DEFVAL	{ 0 }
::= { lldpXPnoLocEntry 2 }	
lldpXPnoLocPortRxDValue	OBJECT-TYPE
SYNTAX	Unsigned32
UNITS	"ns"
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	" This integer value represents the PortRxDelay in nanoseconds which was measured by the local system on the corresponding port. A value of zero shall be used if the system either could not accomplish the measurement or does not support such a measurement." "
DEFVAL	{ 0 }
::= { lldpXPnoLocEntry 3 }	
lldpXPnoLocPortStatusRT2	OBJECT-TYPE
SYNTAX	INTEGER { off(0), configured(1), running(2) }
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	" This value represents the status of the corresponding port of the local system according to RT class 2. A value of off(0) means that there isn't any RT2 capability available for this port. When the port is configured for RT2 mode, but the mode isn't active yet the value will be configured(1). If the RT2 mode is configured for this port and the mode is active, the value will be running(2). "
::= { lldpXPnoLocEntry 4 }	
lldpXPnoLocPortStatusRT3	OBJECT-TYPE
SYNTAX	INTEGER { off(0), configured(1), up(2), down(3), running(4) }
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	" This value represents the status of the corresponding port of the local system according to RT class 3. A value of off(0) means that there isn't any RT3 capability available for this port. When the port is

yet
reception

configured for RT3 mode, but the mode isn't active
the value will be configured(1).
When the port is ready for transmission and
of RT3 traffic, the port status will be running(4).
"

```
 ::= { lldpXPnoLocEntry 5 }
```

lldpXPnoLocPortNoS
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
DisplayString
read-only
current
"

The local PROFINET NameofStation. If the station
configured yet, the value of this object will be
MAC address of the device as a string.
"

```
 ::= { lldpXPnoLocEntry 6 }
```

lldpXPnoLocPortMrpUuid
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
OCTET STRING (SIZE (16))
read-only
current
"

The UUID of the MRP domain to which this port
belongs
to. If the port doesn't belong to a MRP domain, the
value
must be NIL ('0000000000000000').
"

```
 ::= { lldpXPnoLocEntry 7 }
```

lldpXPnoLocPortMrrtStatus
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
INTEGER {
off(0),
configured(1),
up(2)
}
read-only
current
"

This object reports the status of the MRRT entity
of the
corresponding port.
A value of off(0) means that there isn't any MRRT
capability available for this port or it is
switched off.
The value configured(1) indicates that MRRT is
configured
for the port. When MRRT is active on the port, the
value
will be up(2).
"

```
 ::= { lldpXPnoLocEntry 8 }
```

lldpXPnoLocPortPtcpMaster
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
MacAddress
read-only
current
"

The interface MAC address of the PTCP sync master
device
of the PTCP subdomain. If unknown it shall be set
to zero.
"

```
 ::= { lldpXPnoLocEntry 9 }
```

lldpXPnoLocPortPtcpSubdomainUUID
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
OCTET STRING (SIZE (16))
read-only
current
"

The UUID of the PTCP subdomain to which this port
belongs

or the subdomain
be

::= { lldpXPnoLocEntry 10 }

lldpXPnoLocPortPtcpIRDataUUID
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

belongs

or the domain
be

::= { lldpXPnoLocEntry 11 }

lldpXPnoLocPortModeRT3
SYNTAX

MAX-ACCESS
STATUS
DESCRIPTION

object is

valid, else only

::= { lldpXPnoLocEntry 12 }

lldpXPnoLocPortPeriodLength
SYNTAX
UNITS
MAX-ACCESS
STATUS
DESCRIPTION

cycle

::= { lldpXPnoLocEntry 13 }

lldpXPnoLocPortPeriodValidity
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

value of

entry is

::= { lldpXPnoLocEntry 14 }

lldpXPnoLocPortRedOffset
SYNTAX
UNITS
MAX-ACCESS
STATUS
DESCRIPTION

RT_CLASS_3 period

corresponding port

nanoseconds.

to. If the port doesn't belong to a PTCIP subdomain
is invalid or unknown the value of this object must
be
NIL ('0000000000000000').
"

OBJECT-TYPE
OCTET STRING (SIZE (16))
read-only
current
"

The UUID of the IR data domain to which this port

to. If the port doesn't belong to a IR data domain
is invalid or unknown the value of this object must
be
NIL ('0000000000000000').
"

OBJECT-TYPE
INTEGER
{
 standard(1),
 optimized(0)
}

read-only
current
"

The mode in which the RT3 status is given. If the
in standard mode all five values of RT3 status are

"

OBJECT-TYPE
Unsigned32 (31250..4000000)
"ns"
read-only
current
"

This integer value represents the duration of a
in nanoseconds on the corresponding port.
The value shall be a multiple of 31250 nanoseconds.
"

OBJECT-TYPE
TruthValue
read-only
current
"

The value of this object indicates whether the

lldpXPnoLocPortPeriodLength of the according table
valid or not.
"

OBJECT-TYPE
Unsigned32 (0..3999999)
"ns"
read-only
current
"

This integer value represents the begin of the

of the cycle of the receive direction on the

as an offset relative to the begin of the cycle in

```

::= { lldpXPnoLocEntry 15 }

lldpXPnoLocPortRedValidity      OBJECT-TYPE
    SYNTAX                      TruthValue
    MAX-ACCESS                  read-only
    STATUS                      current
    DESCRIPTION                  "
                                The value of this object indicates whether the
value of                        lldpXPnoLocPortRedOffset of the according table
entry is                        valid or not.
                                "

::= { lldpXPnoLocEntry 16 }

lldpXPnoLocPortOrangeOffset     OBJECT-TYPE
    SYNTAX                      Unsigned32 (0..3999999)
    UNITS                        "ns"
    MAX-ACCESS                  read-only
    STATUS                      current
    DESCRIPTION                  "
                                This integer value represents the begin of the
RT_CLASS_2 period              of the cycle of the receive direction on the
corresponding port              as an offset relative to the begin of the cycle in
nanoseconds.                    "

::= { lldpXPnoLocEntry 17 }

lldpXPnoLocPortOrangeValidity   OBJECT-TYPE
    SYNTAX                      TruthValue
    MAX-ACCESS                  read-only
    STATUS                      current
    DESCRIPTION                  "
                                The value of this object indicates whether the
value of                        lldpXPnoLocPortOrangeOffset of the according table
entry is                        valid or not.
                                "

::= { lldpXPnoLocEntry 18 }

lldpXPnoLocPortGreenOffset      OBJECT-TYPE
    SYNTAX                      Unsigned32 (0..3999999)
    UNITS                        "ns"
    MAX-ACCESS                  read-only
    STATUS                      current
    DESCRIPTION                  "
                                This integer value represents the begin of the
unrestricted period             of the cycle of the receive direction on the
corresponding port              as an offset relative to the begin of the cycle in
nanoseconds.                    "

::= { lldpXPnoLocEntry 19 }

lldpXPnoLocPortGreenValidity    OBJECT-TYPE
    SYNTAX                      TruthValue
    MAX-ACCESS                  read-only
    STATUS                      current
    DESCRIPTION                  "
                                The value of this object indicates whether the
value of                        lldpXPnoLocPortGreenOffset of the according table
entry is                        valid or not.
                                "

::= { lldpXPnoLocEntry 20 }

lldpXPnoLocPortStatusRT3OptimizationSupported OBJECT-TYPE
    SYNTAX                      TruthValue
    MAX-ACCESS                  read-only
    STATUS                      current
    DESCRIPTION                  "

```

corresponding
RTC3PSM.
The value true(1) indicates the support, the value false(2) indicates no support of the optimized
RTC3PSM.
"

::= { lldpXPnoLocEntry 21 }

lldpXPnoLocPortStatusRT3PreambleShorteningSupported OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This value represents the status of the
corresponding
port of the local system according to the
shortening of the Preamble.
The value true(1) indicates the support, the value false(2) indicates no support of the shortening of
the Preamble.
"

::= { lldpXPnoLocEntry 22 }

lldpXPnoLocPortStatusRT3PreambleShortening OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This value represents the status of the
corresponding
port of the local system according to the Preamble
lengths.
The value true(1) indicates a Preamble length of
one octet,
the value false(2) indicates a Preamble length of
seven octets.
"

::= { lldpXPnoLocEntry 23 }

lldpXPnoLocPortStatusRT3FragmentationSupported OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This value represents the status of the
corresponding
port of the local system according to the
fragmentation.
The value true(1) indicates the support, the value false(2) indicates no support of fragmentation.
"

::= { lldpXPnoLocEntry 24 }

lldpXPnoLocPortStatusRT3Fragmentation OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This value represents the status of the
corresponding
port of the local system according to the
fragmentation.
The value true(1) indicates the activation, the
value
false(2) indicates the deactivation.
"

::= { lldpXPnoLocEntry 25 }

PNO - Remote System Information

lldpXPnoRemTable OBJECT-TYPE
SYNTAX SEQUENCE OF LldpXPnoRemEntry
MAX-ACCESS not-accessible
STATUS current
DESCRIPTION
"

network	This table contains one or more rows per physical connection known to this agent. The agent may wish to ensure that only one lldpXPnoRemEntry is present for each local port, or it may choose to maintain multiple lldpXPnoRemEntries for the same local port.
::= { lldpXPnoRemoteData 1 }	
lldpXPnoRemEntry	OBJECT-TYPE
SYNTAX	LldpXPnoRemEntry
MAX-ACCESS	not-accessible
STATUS	current
DESCRIPTION	"
the	Each entry represents the received information of communication partner on this physical connection.
	The entries feature multiple indices from the lldpRemEntry of the LLDP-MIB, therefore it is only present along with the description defined by the associated lldpRemEntry entry.
INDEX	"
lldpRemIndex }	{ lldpRemTimeMark, lldpRemLocalPortNum,
::= { lldpXPnoRemTable 1 }	
LldpXPnoRemEntry ::=	SEQUENCE {
lldpXPnoRemLPDValue	Unsigned32,
lldpXPnoRemPortTxDValue	Unsigned32,
lldpXPnoRemPortRxDValue	Unsigned32,
lldpXPnoRemPortStatusRT2	INTEGER,
lldpXPnoRemPortStatusRT3	INTEGER,
lldpXPnoRemPortNoS	DisplayString,
lldpXPnoRemPortMrpUuId	OCTET STRING,
lldpXPnoRemPortMrtrStatus	INTEGER,
lldpXPnoRemPortPtcpMaster	MacAddress,
lldpXPnoRemPortPtcpSubdomainUID	OCTET STRING,
lldpXPnoRemPortPtcpIRDataUUID	OCTET STRING,
lldpXPnoRemPortModeRT3	INTEGER,
lldpXPnoRemPortPeriodLength	Unsigned32,
lldpXPnoRemPortPeriodValidity	TruthValue,
lldpXPnoRemPortRedOffset	Unsigned32,
lldpXPnoRemPortRedValidity	TruthValue,
lldpXPnoRemPortOrangeOffset	Unsigned32,
lldpXPnoRemPortOrangeValidity	TruthValue,
lldpXPnoRemPortGreenOffset	Unsigned32,
lldpXPnoRemPortGreenValidity	TruthValue,
lldpXPnoRemPortStatusRT3PreambleShortening	TruthValue,
lldpXPnoRemPortStatusRT3Fragmentation	TruthValue
}	
lldpXPnoRemLPDValue	OBJECT-TYPE
SYNTAX	Unsigned32
UNITS	"ns"
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"
remote	This integer value represents the line propagation delay in nanoseconds which was measured by the system on the corresponding port.
either	A value of zero shall be used if the remote system could not accomplish the measurement or does not support such a measurement.
::= { lldpXPnoRemEntry 1 }	"
lldpXPnoRemPortTxDValue	OBJECT-TYPE
SYNTAX	Unsigned32
UNITS	"ns"
MAX-ACCESS	read-only

STATUS	current
DESCRIPTION	"
	This integer value represents the PortTxDelay in nanoseconds which was measured by the remote system on the corresponding port.
either	A value of zero shall be used if the remote system
support	could not accomplish the measurement or does not
	such a measurement.
	"
::= { lldpXPnoRemEntry 2 }	
lldpXPnoRemPortRxDValue	OBJECT-TYPE
SYNTAX	Unsigned32
UNITS	"ns"
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"
	This integer value represents the PortRxDelay in nanoseconds which was measured by the remote system on the corresponding port.
either	A value of zero shall be used if the remote system
support	could not accomplish the measurement or does not
	such a measurement.
	"
::= { lldpXPnoRemEntry 3 }	
lldpXPnoRemPortStatusRT2	OBJECT-TYPE
SYNTAX	INTEGER {
	off(0),
	configured(1),
	running(2)
	}
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"
corresponding	This value represents the status of the
	port of the remote system according to RT class 2.
is	A value of off(0) means that there isn't any RT2
yet	capability available for this port. When the port
the	configured for RT2 mode, but the mode isn't active
	the value will be configured(1). If the RT2 mode is
	configured for this port and the mode is active,
	value will be running(2).
	"
::= { lldpXPnoRemEntry 4 }	
lldpXPnoRemPortStatusRT3	OBJECT-TYPE
SYNTAX	INTEGER {
	off(0),
	configured(1),
	up(2),
	down(3),
	running(4)
	}
MAX-ACCESS	read-only
STATUS	current
DESCRIPTION	"
corresponding	This value represents the status of the
	port of the remote system according to RT class 3.
is	A value of off(0) means that there isn't any RT3
yet	capability available for this port. When the port
	configured for RT3 mode, but the mode isn't active
	the value will be configured(1).

reception

When the port is ready for transmission and of RT3 traffic, the port status will be running(4).
"

::= { lldpXPnoRemEntry 5 }

lldpXPnoRemPortNoS
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
DisplayString
read-only
current
"

The PROFINET NameofStation of the remote partner.

If the
object

station isn't configured yet, the value of this will be the MAC address of the device as a string.
"

::= { lldpXPnoRemEntry 6 }

lldpXPnoRemPortMrpUuid
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
OCTET STRING (SIZE (16))
read-only
current
"

The UUID of the MRP domain to which the corresponding port of the remote system belongs to. If the port doesn't belong to a MRP domain, the value must be NIL
"

('0000000000000000').

::= { lldpXPnoRemEntry 7 }

lldpXPnoRemPortMrrtStatus
SYNTAX

OBJECT-TYPE
INTEGER {
off(0),
configured(1),
up(2)
}
read-only
current
"

MAX-ACCESS
STATUS
DESCRIPTION

This object reports the status of the MRRT entity of the corresponding port.

A value of off(0) means that there isn't any MRRT capability available for this port or it is switched off.

The value configured(1) indicates that MRRT is configured for the port. When MRRT is active on the port, the value will be up(2).
"

::= { lldpXPnoRemEntry 8 }

lldpXPnoRemPortPtcpMaster
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
MacAddress
read-only
current
"

The interface MAC address of the PTCP sync master device of the PTCP subdomain. If unknown it shall be set to zero.
"

::= { lldpXPnoRemEntry 9 }

lldpXPnoRemPortPtcpSubdomainUUID
SYNTAX
MAX-ACCESS
STATUS
DESCRIPTION

OBJECT-TYPE
OCTET STRING (SIZE (16))
read-only
current
"

The UUID of the PTCP subdomain to which this port belongs to. If the port doesn't belong to a PTCP subdomain or the subdomain

be is invalid or unknown the value of this object must
NIL ('0000000000000000').

::= { lldpXPnoRemEntry 10 }

lldpXPnoRemPortPtcPIRDataUUID OBJECT-TYPE
SYNTAX OCTET STRING (SIZE (16))
MAX-ACCESS read-only
STATUS current
DESCRIPTION
The UUID of the IR data domain to which this port
belongs to. If the port doesn't belong to a IR data domain
or the domain is invalid or unknown the value of this object must
be NIL ('0000000000000000').

::= { lldpXPnoRemEntry 11 }

lldpXPnoRemPortModeRT3 OBJECT-TYPE
SYNTAX INTEGER
{
standard(1),
optimized(0)
}
MAX-ACCESS read-only
STATUS current
DESCRIPTION
The mode in which the RT3 status is given. If the
object is in standard mode all five values of RT3 status are
valid, else only

::= { lldpXPnoRemEntry 12 }

lldpXPnoRemPortPeriodLength OBJECT-TYPE
SYNTAX Unsigned32 (31250..4000000)
UNITS "ns"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This integer value represents the duration of a
cycle in nanoseconds on the corresponding port.
The value shall be a multiple of 31250 nanoseconds.

::= { lldpXPnoRemEntry 13 }

lldpXPnoRemPortPeriodValidity OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
The value of this object indicates whether the
value of lldpXPnoRemPortPeriodLength of the according table
entry is valid or not.

::= { lldpXPnoRemEntry 14 }

lldpXPnoRemPortRedOffset OBJECT-TYPE
SYNTAX Unsigned32 (0..3999999)
UNITS "ns"
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This integer value represents the begin of the
RT_CLASS_3 period of the cycle of the receive direction on the
corresponding port as an offset relative to the begin of the cycle in
nanoseconds.

::= { lldpXPnoRemEntry 15 }

lldpXPnoRemPortRedValidity OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The value of this object indicates whether the value of lldpXPnoRemPortRedOffset of the according table entry is valid or not."
 ::= { lldpXPnoRemEntry 16 }

lldpXPnoRemPortOrangeOffset OBJECT-TYPE
 SYNTAX Unsigned32 (0..3999999)
 UNITS "ns"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "This integer value represents the begin of the RT_CLASS_2 period of the cycle of the receive direction on the corresponding port as an offset relative to the begin of the cycle in nanoseconds."
 ::= { lldpXPnoRemEntry 17 }

lldpXPnoRemPortOrangeValidity OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The value of this object indicates whether the value of lldpXPnoRemPortOrangeOffset of the according table entry is valid or not."
 ::= { lldpXPnoRemEntry 18 }

lldpXPnoRemPortGreenOffset OBJECT-TYPE
 SYNTAX Unsigned32 (0..3999999)
 UNITS "ns"
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "This integer value represents the begin of the unrestricted period of the cycle of the receive direction on the corresponding port as an offset relative to the begin of the cycle in nanoseconds."
 ::= { lldpXPnoRemEntry 19 }

lldpXPnoRemPortGreenValidity OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "The value of this object indicates whether the value of lldpXPnoRemPortGreenOffset of the according table entry is valid or not."
 ::= { lldpXPnoRemEntry 20 }

lldpXPnoRemPortStatusRT3PreambleShortening OBJECT-TYPE
 SYNTAX TruthValue
 MAX-ACCESS read-only
 STATUS current
 DESCRIPTION
 "This value represents the status of the corresponding

lengths.
one octet,
seven octets.

port of the remote system according to the Preamble
The value true(1) indicates a Preamble length of
the value false(2) indicates a Preamble length of
"

```
 ::= { lldpXPnoRemEntry 21 }
```

lldpXPnoRemPortStatusRT3Fragmentation OBJECT-TYPE
SYNTAX TruthValue
MAX-ACCESS read-only
STATUS current
DESCRIPTION
This value represents the status of the
corresponding
port of the remote system according to the
fragmentation.
The value true(1) indicates the activation, the
value
false(2) indicates the deactivation.
"

```
 ::= { lldpXPnoRemEntry 22 }
```

Conformance Information

lldpXPnoConformance OBJECT IDENTIFIER ::= { lldpXPnoMIB 2 }
lldpXPnoCompliances OBJECT IDENTIFIER ::= { lldpXPnoConformance 1 }
lldpXPnoGroups OBJECT IDENTIFIER ::= { lldpXPnoConformance 2 }

compliance statements

lldpXPnoCompliance MODULE-COMPLIANCE
STATUS current
DESCRIPTION
The compliance statement for SNMP entities which
implement the PNO organizationally defined LLDP
extension MIB.
"
MODULE -- compliant to this module
MANDATORY-GROUPS { lldpXPnoConfigGroup, lldpXPnoLocGroup,
lldpXPnoRemGroup }
GROUP lldpXPnoMrpGroup
DESCRIPTION "Required if the system provides MRP."
GROUP lldpXPnoPtcpGroup
DESCRIPTION "Required if the system provides PTCP."
 ::= { lldpXPnoCompliances 1 }

MIB groupings

lldpXPnoConfigGroup OBJECT-GROUP
OBJECTS {
lldpXPnoConfigSPDtxEnable,
lldpXPnoConfigPortStatusTxEnable,
lldpXPnoConfigAliasTxEnable
}
STATUS current
DESCRIPTION
The collection of objects which are used to
configure the
PNO organizationally defined LLDP extension
implementation behavior.
This group is mandatory for agents which implement
the PNO
organizationally defined LLDP extension, because
the information
about the signal propagation delay is necessary to
configure
PROFINET domains.
"

```
 ::= { lldpXPnoGroups 1 }
```

lldpXPnoLocGroup OBJECT-GROUP
OBJECTS { lldpXPnoLocLPDValue,
lldpXPnoLocPortTxDValue,
lldpXPnoLocPortRxDValue,
lldpXPnoLocPortStatusRT2,

```

        lldpXPnoLocPortStatusRT3,
        lldpXPnoLocPortNoS,
        lldpXPnoLocPortModeRT3,
        lldpXPnoLocPortPeriodLength,
        lldpXPnoLocPortPeriodValidity,
        lldpXPnoLocPortRedOffset,
        lldpXPnoLocPortRedValidity,
        lldpXPnoLocPortOrangeOffset,
        lldpXPnoLocPortOrangeValidity,
        lldpXPnoLocPortGreenOffset,
        lldpXPnoLocPortGreenValidity,
        lldpXPnoLocPortStatusRT3OptimizationSupported,

lldpXPnoLocPortStatusRT3PreambleShorteningSupported,
        lldpXPnoLocPortStatusRT3PreambleShortening,
        lldpXPnoLocPortStatusRT3FragmentationSupported,
        lldpXPnoLocPortStatusRT3Fragmentation
    }
    STATUS
    DESCRIPTION
    current
    "
    The collection of objects which are used to
    configure the
    implementation behavior.
    PNO organizationally defined LLDP extension

    This group is mandatory for agents which implement
    the PNO
    organizationally defined LLDP extension, because
    the information
    about the signal propagation delay is necessary to
    configure
    PROFINET domains.
    "

    ::= { lldpXPnoGroups 2 }

lldpXPnoRemGroup
OBJECTS
OBJECT-GROUP
{ lldpXPnoRemLPDValue,
  lldpXPnoRemPortTxDValue,
  lldpXPnoRemPortRxDValue,
  lldpXPnoRemPortStatusRT2,
  lldpXPnoRemPortStatusRT3,
  lldpXPnoRemPortNoS,
  lldpXPnoRemPortModeRT3,
  lldpXPnoRemPortPeriodLength,
  lldpXPnoRemPortPeriodValidity,
  lldpXPnoRemPortRedOffset,
  lldpXPnoRemPortRedValidity,
  lldpXPnoRemPortOrangeOffset,
  lldpXPnoRemPortOrangeValidity,
  lldpXPnoRemPortGreenOffset,
  lldpXPnoRemPortGreenValidity,
  lldpXPnoRemPortStatusRT3PreambleShortening,
  lldpXPnoRemPortStatusRT3Fragmentation
}
    STATUS
    DESCRIPTION
    current
    "
    The collection of objects which are used to
    configure the
    implementation behavior.
    PNO organizationally defined LLDP extension

    This group is mandatory for agents which implement
    the PNO
    organizationally defined LLDP extension, because
    the information
    about the signal propagation delay is necessary to
    configure
    PROFINET domains.
    "

    ::= { lldpXPnoGroups 3 }

lldpXPnoMrpGroup
OBJECTS
OBJECT-GROUP
{ lldpXPnoConfigMrpTxEnable,
  lldpXPnoLocPortMrpUuid,
  lldpXPnoLocPortMrpStatus,
  lldpXPnoRemPortMrpUuid,
  lldpXPnoRemPortMrpStatus
}

```

<pre> STATUS DESCRIPTION configure the implementation behavior. the PNO the information configure ::= { lldpXPnoGroups 4 } lldpXPnoPtcpGroup OBJECTS STATUS DESCRIPTION configure the implementation behavior. the PNO the information configure ::= { lldpXPnoGroups 5 } END </pre>	<pre> current " The collection of objects which are used to PNO organizationally defined LLDP extension This group is mandatory for agents which implement organizationally defined LLDP extension, because about the signal propagation delay is necessary to PROFINET domains. " OBJECT-GROUP { lldpXPnoConfigPtcpTxEnable, lldpXPnoLocPortPtcpMaster, lldpXPnoLocPortPtcpSubdomainUUID, lldpXPnoLocPortPtcpIRDataUUID, lldpXPnoRemPortPtcpMaster, lldpXPnoRemPortPtcpSubdomainUUID, lldpXPnoRemPortPtcpIRDataUUID } current " The collection of objects which are used to PNO organizationally defined LLDP extension This group is mandatory for agents which implement organizationally defined LLDP extension, because about the signal propagation delay is necessary to PROFINET domains. " </pre>



Bibliography

IEC 61784-1, *Industrial communication networks – Profiles – Part 1: Fieldbus profiles*

IEC 61784-2, *Industrial communication networks – Profiles – Part 2: Additional fieldbus profiles for real-time networks based on ISO/IEC 8802-3*

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

IEC 60793-2-30, *Optical fibres – Part 2-30: Product specifications – Sectional specification for category A3 multimode fibres*

IEC 60793-2-40, *Optical fibres – Part 2-40: Product specifications – Sectional specification for category A4 multimode fibres*

ISO/IEC 8859-1, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

ISO 15745-4:2003, *Industrial automation systems and integration – Open systems application integration framework – Part 4: Reference description for Ethernet-based control systems Amendment 1:2006, PROFINET profiles*

IEEE 802.11n, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications – Amendment 5: Enhancements for Higher Throughput*, available at <<http://www.ieee.org>>

IETF RFC 2737, *Entity MIB (Version 2)*, available at <<http://www.ietf.org>>

IETF RFC 3376, *Internet Group Management Protocol – Version 3*, available at <<http://www.ietf.org>>

IETF RFC 4133, *Entity MIB (Version 3)*, available at <<http://www.ietf.org>>

IETF RFC 4604, *Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast*, available at <<http://www.ietf.org>>

IETF RFC 4632, *Classless Inter-domain Routing (CIDR): The Internet Address Assignment and Aggregation Plan*, available at <<http://www.ietf.org>>

IETF RFC 5494, *IANA Allocation Guidelines for the Address Resolution Protocol (ARP)*, available at <<http://www.ietf.org>>

IETF RFC 5891, *Internationalized Domain Names in Applications (IDNA): Protocol*, available at <<http://www.ietf.org>>

SOMMAIRE

AVANT-PROPOS	754
INTRODUCTION.....	757
1 Domaine d'application	759
1.1 Généralités.....	759
1.2 Spécifications.....	759
1.3 Conformité	760
2 Références normatives.....	760
3 Termes, définitions, abréviations, symboles et conventions	762
3.1 Termes et définitions référencés	762
3.2 Termes et définitions supplémentaires pour la périphérie décentralisée.....	763
3.3 Abréviations et symboles.....	772
3.4 Conventions	777
4 Spécification du protocole de couche application pour les protocoles communs	785
4.1 Description de la syntaxe FAL	785
4.2 Syntaxe de transfert	788
4.3 Reconnaissance et configuration de base.....	801
4.4 Contrôle de temps de travail précis	840
4.5 Synchronisation temporelle.....	918
4.6 Redondance de support	918
4.7 Cyclique en temps réel.....	918
4.8 Acyclique en temps-réel	949
4.9 Fragmentation	967
4.10 Appel de procédure à distance	984
4.11 Reconnaissance de couche de liaison	1002
4.12 Ponts MAC	1013
4.13 Ponts virtuels.....	1043
4.14 Suite IP.....	1052
4.15 Système de noms de domaine.....	1055
4.16 Configuration d'hôte dynamique	1056
4.17 Gestion de réseau simple	1058
4.18 Machines de Protocole de Mapping DLL communes.....	1058
5 Spécification du protocole de couche application pour la périphérie décentralisée.....	1064
5.1 Description de la syntaxe FAL	1064
5.2 Syntaxe de transfert	1082
5.3 Diagrammes d'états de protocole FAL	1271
5.4 Diagramme d'états de contexte AP	1273
5.5 Machines de Protocole de Service FAL	1273
5.6 Machines de protocole de relation entre applications.....	1293
5.7 Machines de Protocole de Mapping DLL.....	1413
Annexe A (normative) Etablissement unifié d'une AR pour toutes les classes RT.....	1414
Annexe B (normative) Etablissement compatible d'une AR	1423
Annexe C (informative) Etablissement d'une AR d'accès à l'appareil.....	1428
Annexe D (informative) Etablissement d'une AR (procédure accélérée)	1430
Annexe E (informative) Etablissement d'une AR (procédure de démarrage rapide)	1433
Annexe F (informative) Exemple de procédure de chargement, de stockage et d'extraction	1435

Annexe G (informative) Couches du modèle de référence OSI	1438
Annexe H (informative) Vue d'ensemble des diagrammes d'états de l'appareil de commande E/S et de l'appareil E/S	1439
Annexe I (informative) Régénération de priorité	1442
Annexe J (informative) Présentation de la hiérarchie des maîtres de synchronisation PTCP	1444
Annexe K (informative) Optimisation de l'utilisation de la largeur de bande	1446
Annexe L (informative) Contraintes de temps pour l'allocation de largeur de bande	1449
Annexe M (informative) Contraintes de temps pour le transfert d'une trame	1452
Annexe N (informative) Principe de la combinaison de trame dynamique	1453
Annexe O (informative) Principe de Fragmentation	1457
Annexe P (informative) MRPD – Principe de redondance de support sans coupure	1460
Annexe Q (normative) Principe d'un RED_RELAY sans information de transfert dans PDIRFrameData	1463
Annexe R (informative) Optimisation du démarrage rapide sans autonégociation	1466
Annexe S (informative) Traitement des erreurs TX	1468
Annexe T (informative) Exemple d'une séquence PrmBegin, PrmEnd et ApplRdy	1469
Annexe U (informative) Liste des MIB pris en charge	1470
Annexe V (informative) Structure et contenu de BLOB	1471
Annexe W (normative) LLDP EXT MIB	1472
Bibliographie	1489
Figure 1 – Structure commune des champs spécifiques	778
Figure 2 – Structure commune de champs spécifiques pour l'octet 1 (élevé)	779
Figure 3 – Structure commune de champs spécifiques pour l'octet 2 (bas)	779
Figure 4 – Structure commune de champs spécifiques pour l'octet 1 (élevé)	780
Figure 5 – Structure commune de champs spécifiques pour l'octet 2	780
Figure 6 – Structure commune de champs spécifiques pour l'octet 3	780
Figure 7 – Structure commune de champs spécifiques pour l'octet 4 (bas)	781
Figure 8 – Codage du type de données BinaryDate	790
Figure 9 – Codage d'une valeur TimeOfDay avec indication de date	790
Figure 10 – Codage d'une valeur TimeOfDay sans indication de date	791
Figure 11 – Encodage d'une valeur TimeDifference avec indication de date	791
Figure 12 – Encodage d'une valeur TimeDifference sans indication de date	791
Figure 13 – FastForwardingMulticastMACAdd	795
Figure 14 – Schéma de transition d'états de DCPUCS	825
Figure 15 – Schéma de transition d'états de DCPUCR	829
Figure 16 – Schéma de transition d'états de DCPMCS	832
Figure 17 – Schéma de transition d'états du DCPMCR	835
Figure 18 – Schéma de transition d'états de DCPHMCS	838
Figure 19 – Schéma de transition d'états de DCPHMCR	839
Figure 20 – Plage de valeurs PTCP_SequenceID	845
Figure 21 – Correspondance d'échelles de temps entre PTCP_Time et CycleCounter	848

Figure 22 – Point d'horodatage de message	853
Figure 23 – Modèle de temporisateur	854
Figure 24 – Quatre horodatages de message.....	855
Figure 25 – Protocole de retard de ligne avec suivi.....	856
Figure 26 – Protocole de retard de ligne sans suivi.....	856
Figure 27 – Mesure du retard de ligne	858
Figure 28 – Paramètre de modèle pour l'utilisation GSDML.....	860
Figure 29 – Mesure du délai de pontage	861
Figure 30 – Accumulation de retard	862
Figure 31 – Cas le plus défavorable pour l'écart de temps de synchronisation cumulé.....	863
Figure 32 – Schéma de la mesure de l'écart	863
Figure 33 – Mesure de l'écart	864
Figure 34 – Maître PTCP envoyant une Sync-Frame (trame de synchronisation) sans Follow Up-Frame (trame de suivi)	865
Figure 35 – Maître PTCP envoyant une Sync-Frame (trame de synchronisation) avec FollowUp-Frame (trame de suivi)	866
Figure 36 – Esclave de synchronisation !FU transférant une Sync-Frame (trame de synchronisation)	867
Figure 37 – Esclave de synchronisation FU transférant une Sync-Frame (trame de synchronisation) et une FollowUp-Frame (trame de suivi).....	868
Figure 38 – Esclave de synchronisation FU transférant une Sync-Frame et générant une FollowUp-Frame (trame de suivi)	869
Figure 39 – Principe de surveillance de la mesure de retard de ligne.....	871
Figure 40 – Schéma de transition d'états de DELAY_REQ	872
Figure 41 – Schéma de transition d'états de DELAY_RSP	879
Figure 42 – Vue d'ensemble de PTCP.....	883
Figure 43 – Schéma de transition d'états de SYN_BMA	886
Figure 44 – Schéma de transition d'états de SYN_MPSM	895
Figure 45 – Schéma de transition d'états de SYN_SPSM.....	901
Figure 46 – Schéma de transition d'états de SYNC_RELAY.....	908
Figure 47 – Schéma de transition d'états de SCHEDULER	915
Figure 48 – Plage de valeurs CycleCounter	920
Figure 49 – Structure de CycleCounter	922
Figure 50 – Paramètre CycleCounter optimisé	922
Figure 51 – Règle de génération de SFCRC16	926
Figure 52 – Plage de valeurs SFCycleCounter.....	928
Figure 53 – Structure de base d'une PPM comportant une structure de trame.....	931
Figure 54 – Structure de base d'une PPM comportant une structure de sous-trame	932
Figure 55 – Schéma de transition d'états of PPM	934
Figure 56 – Structure de base d'une CPM.....	938
Figure 57 – Schéma de transition d'états de CPM.....	940
Figure 58 – Diagramme d'adressage de RTA.....	950
Figure 59 – Structure de l'APM	954
Figure 60 – Structure de l'APMS	955
Figure 61 – Schéma de transition d'états d'APMS	957

Figure 62 – Structure de l'APMR.....	962
Figure 63 – Schéma de transition d'états d'APMR.....	964
Figure 64 – Schéma de transition d'états de FRAG_D.....	974
Figure 65 – Schéma de transition d'états de FRAG_S.....	978
Figure 66 – Schéma de transition d'états de DEFRAG	981
Figure 67 – Schéma de transition d'états de RTC3PSM	1017
Figure 68 – Schéma de transition d'états pour générer des événements	1021
Figure 69 – Schéma de transition d'états de RED_RELAY	1023
Figure 70 – Diagramme de DFP_RELAY.....	1027
Figure 71 – Diagramme de DFP_INBOUND et de DFP_STORAGE	1027
Figure 72 – Diagramme de DFP_OUTBOUND	1028
Figure 73 – Schéma de transition d'états de DFP_RELAY.....	1029
Figure 74 – Schéma de transition d'états de DFP_RELAY_INBOUND.....	1032
Figure 75 – Schéma de transition d'états de DFP_RELAY_IN_STORAGE.....	1036
Figure 76 – Schéma de transition d'états de DFP_RELAY_OUTBOUND.....	1040
Figure 77 – Schéma de transition d'états de MUX.....	1044
Figure 78 – Schéma de transition d'états de DEMUX.....	1049
Figure 79 – Structure des machines de protocole dans DMPM (pont)	1060
Figure 80 – Schéma de transition d'états du LMPM.....	1062
Figure 81 – Plage de valeurs AlarmSpecifie.SequenceNumber	1097
Figure 82 – FrameSendOffset par rapport à la durée d'un cycle.....	1136
Figure 83 – Classification selon la sévérité de diagnostic, maintenance et qualified.....	1182
Figure 84 – Principes de calcul pour un cycle.....	1201
Figure 85 – Principe de calcul du YellowTime minimal	1201
Figure 86 – Définition de l'intervalle réservé	1208
Figure 87 – Vue de niveau supérieur de PLL window	1211
Figure 88 – Définition de PLL window	1212
Figure 89 – Vue supérieure de la fenêtre PLL de temps.....	1215
Figure 90 – Définition de la fenêtre PLL de temps.....	1216
Figure 91 – Détection des trames supprimées — appear	1224
Figure 92 – Détection des trames supprimées — disappear.....	1225
Figure 93 – Détection d'erreur tardive de DFP — appear and disappear	1232
Figure 94 – MediaRedundancyWatchDog expiré — appear and disappear	1233
Figure 95 – Relation parmi les machines de protocoles.....	1271
Figure 96 – Schéma de transition d'états d'ALPMI	1294
Figure 97 – Schéma de transition d'états d'ALPMR.....	1298
Figure 98 – Diagramme du CM d'appareil E/S.....	1301
Figure 99 – Schéma de transition d'états du CM d'appareil E/S	1303
Figure 100 – Schéma de transition d'états de CMDEV	1307
Figure 101 – Diagramme du CM d'appareil E/S – accès à l'appareil.....	1312
Figure 102 – Schéma de transition d'états de CMDEV_DA.....	1314
Figure 103 – Schéma de transition d'états de CMSU	1318
Figure 104 – Schéma de transition d'états de CMIO.....	1323

Figure 105 – Schéma de transition d'états de CMWRR	1327
Figure 106 – Schéma de transition d'états de CMRDR	1331
Figure 107 – Schéma de transition d'états de CMSM	1334
Figure 108 – Schéma de transition d'états de CMPBE	1337
Figure 109 – Schéma de transition d'états de CMDMC	1342
Figure 110 – Schéma de transition d'états de CMINA	1346
Figure 111 – Schéma de transition d'états de CMRPC	1355
Figure 112 – Diagramme du CM d'un appareil de commande E/S	1361
Figure 113 – Schéma de transition d'états de IO controller CM (CM d'appareil de commande E/S)	1362
Figure 114 – Schéma de transition d'états de CMCTL	1366
Figure 115 – Schéma de transition d'états de CTLSM	1372
Figure 116 – Schéma de transition d'états de CTLIO	1375
Figure 117 – Schéma de transition d'états de CTRLDI	1378
Figure 118 – Schéma de transition d'états de CTRLDR	1381
Figure 119 – Schéma de transition d'états de CTRLRPC	1385
Figure 120 – Schéma de transition d'états de CTLSU	1390
Figure 121 – Schéma de transition d'états de CTLWRI	1395
Figure 122 – Schéma de transition d'états de CTLWRR	1399
Figure 123 – Schéma de transition d'états de CTLPBE	1402
Figure 124 – Schéma de transition d'états de CTLDINA	1407
Figure 125 – Attribution automatique de NameOfStation	1413
Figure A.1 – Etablissement d'une AR avec RT_CLASS_1, RT_CLASS_2 ou RT_CLASS_3 (Surveillance de la connexion initiale sans RT)	1416
Figure A.2 – Etablissement d'une AR avec RT_CLASS_1, RT_CLASS_2 ou RT_CLASS_3 (Surveillance de la connexion avec RT)	1418
Figure A.3 – Principe d'évaluation des données lors du démarrage (établissement du canal ROUGE retardé)	1419
Figure A.4 – Principe d'évaluation des données lors du démarrage (établissement du canal ROUGE tôt)	1420
Figure A.5 – Principe d'évaluation des données lors du démarrage (Cas particulier: application en mode isochrone)	1421
Figure B.1 – Etablissement d'une AR avec une AR de RT_CLASS_3 en mode de démarrage "Legacy" (version originelle)	1425
Figure B.2 – Etablissement d'une AR avec RT_CLASS_1, 2 ou AR UDP avec le mode de démarrage "Legacy" (version originelle)	1427
Figure C.1 – Etablissement d'une AR d'accès à l'appareil	1429
Figure D.1 – Etablissement accéléré d'un IOAR sans erreur	1431
Figure D.2 – Etablissement accéléré d'un IOAR avec erreur "tardive"	1432
Figure E.1 – Etablissement d'une IOAR avec un démarrage rapide	1434
Figure F.1 – Exemple de chargement avec stockage	1436
Figure F.2 – Exemple d'extraction avec stockage	1437
Figure G.1 – Attribution des couches du modèle de référence OSI	1438
Figure H.1 – Présentation des diagrammes d'états d'un appareil de commande E/S	1439
Figure H.2 – Présentation des diagrammes d'états d'un appareil E/S	1440
Figure H.3 – Présentation des diagrammes d'états communs	1441

Figure J.1 – Modèle de niveau de la hiérarchie des maîtres de synchronisation	1444
Figure J.2 – Variante à deux niveaux de la hiérarchie des maîtres de synchronisation	1445
Figure K.1 – Développement d'appareils dans une structure linéaire	1446
Figure K.2 – Propagation des trames dans un sens d'émission linéaire	1447
Figure K.3 – Propagation des trames dans un sens de réception	1448
Figure L.1 – Présentation des contraintes de temps pour l'allocation de largeur de bande	1449
Figure L.2 – Calcul de la longueur d'une période ROUGE	1450
Figure L.3 – Calcul de la longueur d'une période VERTE	1450
Figure M.1 – Minimisation du délai de pont	1452
Figure N.1 – Combinaison de trame dynamique	1453
Figure N.2 – Combinaison de trame dynamique – Troncature des sorties	1454
Figure N.3 – Combinaison de trame dynamique – concaténation des entrées	1455
Figure N.4 – Mode nœud d'extrémité	1456
Figure N.5 – Définition de DFPFeed	1456
Figure O.1 – Principe de fragmentation	1457
Figure O.2 – Eléments de protocole des fragments	1458
Figure O.3 – Attribution de la largeur de base avec la fragmentation	1458
Figure O.4 – Gardien pour un domaine de fragmentation	1459
Figure P.1 – Principe de redondance de support sans coupure – I/OCR	1460
Figure P.2 – Principe de redondance de support sans coupure – MCR	1461
Figure P.3 – Principe de redondance de support sans coupure – Ligne	1462
Figure Q.1 – Génération de FrameSendOffset pour un RED_RELAY sans information de transfert dans PDIRFrameData	1463
Figure R.1 – Diagramme d'un commutateur à 2 ports	1466
Figure R.2 – Diagramme de 2 ports	1467
Figure T.1 – Procédure PrmBegin, PrmEnd et ApplRdy	1469
Tableau 1 – Eléments de description d'un diagramme d'états	782
Tableau 2 – Description des éléments d'un diagramme d'états	782
Tableau 3 – Conventions utilisées dans les diagrammes d'états	782
Tableau 4 – Conventions pour les services utilisées dans les diagrammes d'états	783
Tableau 5 – Syntaxe IEEE 802.3 pour la DLPDU	785
Tableau 6 – Syntaxe IEEE 802.11 pour la DLPDU	786
Tableau 7 – Syntaxe IEEE 802.15.1 pour une DLPDU	787
Tableau 8 – SourceAddress	792
Tableau 9 – DCP_MulticastMACAdd pour Identify	792
Tableau 10 – DCP_MulticastMACAdd pour Hello	793
Tableau 11 – DCP_MulticastMACAdd	793
Tableau 12 – PTCP_MulticastMACAdd Plage 1	793
Tableau 13 – PTCP_MulticastMACAdd Plage 2	793
Tableau 14 – PTCP_MulticastMACAdd Plage 3	793
Tableau 15 – PTCP_MulticastMACAdd Plage 4	794
Tableau 16 – PTCP_MulticastMACAdd Plage 5	794

Tableau 17 – PTCP_MulticastMACAdd Plage 6	794
Tableau 18 – PTCP_MulticastMACAdd Plage 7	794
Tableau 19 – PTCP_MulticastMACAdd Plage 8	794
Tableau 20 – Adresse de multidiffusion de destination RT_CLASS_3	796
Tableau 21 – Adresse de multidiffusion de trame non valide RT_CLASS_3	796
Tableau 22 – LT (Longueur/Type)	796
Tableau 23 – TagControllInformation.Priority	797
Tableau 24 – FrameID– Plage 1	797
Tableau 25 – FrameID– Plage 2	797
Tableau 26 – FrameID– Plage 3	798
Tableau 27 – FrameID plage 4	798
Tableau 28 – FrameID page 5	798
Tableau 29 – FrameID page 6	798
Tableau 30 – FrameID page 7	799
Tableau 31 – FrameID page 8	799
Tableau 32 – FrameID page 9	799
Tableau 33 – FrameID page 10	800
Tableau 34 – FrameID page 11	800
Tableau 35 – FrameID page 12	800
Tableau 36 – FrameID page 13	800
Tableau 37 – FragmentationFrameID.FragSequence	800
Tableau 38 – FragmentationFrameID.Constant	800
Tableau 39 – Syntaxe DCP APDU	801
Tableau 40 – Substitutions DCP	802
Tableau 41 – ServiceID	804
Tableau 42 – ServiceType.Selection	805
Tableau 43 – ServiceType.Reserved	805
Tableau 44 – ServiceType.Selection	805
Tableau 45 – ServiceType.Reserved_1	805
Tableau 46 – ServiceType.Response	806
Tableau 47 – ServiceType.Reserved_2	806
Tableau 48 – ResponseDelayFactor	806
Tableau 49 – Liste des options	808
Tableau 50 – Liste des sous-options d'IPOption	808
Tableau 51 – Liste des sous-options de DevicePropertiesOption	808
Tableau 52 – Liste des sous-options de DHCPOption	809
Tableau 53 – Liste des sous-options de ControlOption	809
Tableau 54 – Liste des sous-options de DeviceInitiativeOption	809
Tableau 55 – Liste des sous-options de AllSelectorOption	809
Tableau 56 – Liste des sous-options de ManufacturerSpecificOption	809
Tableau 57 – SuboptionDHCP	810
Tableau 58 – Codage de DCPBlockLength conjointement avec SuboptionStart	811
Tableau 59 – Codage de DCPBlockLength conjointement avec SuboptionStop	812

Tableau 60 – Codage de DCPBlockLength conjointement avec SuboptionSignal	812
Tableau 61 – Codage de DCPBlockLength conjointement avec SuboptionFactoryReset	812
Tableau 62 – Codage de DCPBlockLength conjointement avec SuboptionResetToFactory.....	813
Tableau 63 – Codage de DCPBlockLength conjointement avec SuboptionDeviceInitiative.....	813
Tableau 64 – BlockQualifier avec les options IPOption, DevicePropertiesOption, DHCPOption et ManufacturerSpecificOption	814
Tableau 65 – BlockQualifier avec l'option ControlOption et la sous-option SuboptionResetToFactory.....	814
Tableau 66 – BlockQualifier avec toutes les autres options et sous-options	815
Tableau 67 – BlockError	816
Tableau 68 – BlockInfo pour SuboptionIPParameter	816
Tableau 69 – Bit 1 et Bit 0 de BlockInfo pour SuboptionIPParameter	816
Tableau 70 – Bit 7 de BlockInfo pour SuboptionIPParameter	816
Tableau 71 – BlockInfo pour toutes les autres sous-options.....	817
Tableau 72 – DeviceInitiativeValue	817
Tableau 73 – SignalValue	817
Tableau 74 – DeviceRoleDetails	819
Tableau 75 – IPAddress.....	820
Tableau 76 – Subnetmask	820
Tableau 77 – StandardGateway	821
Tableau 78 – Primitives distantes émises ou reçues par DCPUCS.....	824
Tableau 79 – Primitives locales émises ou reçues par DCPUCS	824
Tableau 80 – Table d'états DCPUCS	825
Tableau 81 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPUCS.....	828
Tableau 82 – Primitives distantes émises ou reçues par DCPUCR.....	828
Tableau 83 – Primitives locales émises ou reçues par DCPUCR.....	829
Tableau 84 – Table d'états DCPUCR	830
Tableau 85 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPUCR	831
Tableau 86 – Primitives distantes émises ou reçues par DCPMCS.....	832
Tableau 87 – Primitives locales émises ou reçues par DCPMCS.....	832
Tableau 88 – Table d'états DCPMCS	833
Tableau 89 – Fonctions utilisées par le DCPMCS	834
Tableau 90 – Primitives distantes émises ou reçues par DCPMCR	835
Tableau 91 – Primitives locales émises ou reçues par DCPMCR	835
Tableau 92 – Table d'états DCPMCR.....	836
Tableau 93 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPMCR	836
Tableau 94 – Primitives distantes émises ou reçues par DCPHMCS	837
Tableau 95 – Primitives locales émises ou reçues par DCPHMCS	837
Tableau 96 – Table d'états DCPHMCS	838
Tableau 97 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPHMCS.....	839
Tableau 98 – Primitives distantes émises ou reçues par DCPHMCR.....	839
Tableau 99 – Primitives locales émises ou reçues par DCPHMCR.....	839

Tableau 100 – Table d'états DCPHMCR	840
Tableau 101 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPHMCR.....	840
Tableau 102 – Syntaxe PTCP APDU.....	841
Tableau 103 – Substitutions PTCP.....	841
Tableau 104 – PTCP_TLVHeader.Type	842
Tableau 105 – PTCP_Delay10ns	843
Tableau 106 – PTCP_Delay1ns_Byte.Value.....	843
Tableau 107 – PTCP_Delay1ns	843
Tableau 108 – PTCP_Delay1ns_FUP.....	844
Tableau 109 – PTCP_SequenceID.....	844
Tableau 110 – PTCP_SubType for OUI (=00-0E-CF)	845
Tableau 111 – PTCP_Seconds	846
Tableau 112 – PTCP_NanoSeconds	846
Tableau 113 – PTCP_Flags.LeapSecond	846
Tableau 114 – Correspondance d'échelles de temps entre MJD, TUC et PTCP_EPOCHNumber	847
Tableau 115 – Correspondance d'échelles de temps entre PTCP_EPOCHNumber, PTCP_Second, PTCP_Nanosecond, CycleCounter et SendClockFactor	847
Tableau 116 – PTCP_MasterPriority1.Priority for SyncID == 0 et SyncProperties.Role == 2.....	849
Tableau 117 – PTCP_MasterPriority1.Priority for SyncID == 0 et SyncProperties.Role == 1.....	850
Tableau 118 – PTCP_MasterPriority1.Level.....	850
Tableau 119 – PTCP_MasterPriority2	850
Tableau 120 – PTCP_ClockClass for SyncID == 0 (synchronisation de l'horloge de travail)	851
Tableau 121 – PTCP_ClockAccuracy.....	851
Tableau 122 – PTCP_ClockVariance	852
Tableau 123 – PTCP_T2PortRxDelay	852
Tableau 124 – PTCP_T3PortTxDelay.....	852
Tableau 125 – PTCP_T2TimeStamp	852
Tableau 126 – Primitives distantes émises ou reçues par DELAY_REQ.....	871
Tableau 127 – Primitives locales émises ou reçues par DELAY_REQ	871
Tableau 128 – Table d'états DELAY_REQ	873
Tableau 129 – Fonctions, macros, temporisateurs et variables utilisés par le DELAY_REQ	876
Tableau 130 – Primitives distantes émises ou reçues par DELAY_RSP	878
Tableau 131 – Primitives locales émises ou reçues par DELAY_RSP	879
Tableau 132 – Table d'états DELAY_RSP.....	880
Tableau 133 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DELAY_RSP.....	882
Tableau 134 – Primitives distantes émises ou reçues par SYN_BMA	884
Tableau 135 – Primitives locales émises ou reçues par SYN_BMA	884
Tableau 136 – Table d'états SYN_BMA	887

Tableau 137 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYN_BMA.....	891
Tableau 138 – Primitives distantes émises ou reçues par SYN_MPSM	894
Tableau 139 – Primitives locales émises ou reçues par SYN_MPSM	894
Tableau 140 – Table d'états SYN_MPSM.....	896
Tableau 141 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYN_MPSM.....	900
Tableau 142 – Primitives distantes émises ou reçues par SYN_SPSM.....	900
Tableau 143 – Primitives locales émises ou reçues par SYN_SPSM.....	901
Tableau 144 – Table d'états de SYN_SPSM	902
Tableau 145 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYN_SPSM	905
Tableau 146 – Table de vérité pour un SyncID pour la réception de trames de synchronisation et de suivi.....	906
Tableau 147 – Primitives distantes émises ou reçues par SYNC_RELAY.....	907
Tableau 148 – Primitives locales émises ou reçues par SYNC_RELAY.....	908
Tableau 149 – Table d'états de SYNC_RELAY	909
Tableau 150 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYNC_RELAY	910
Tableau 151 – Table de vérité pour un SyncID pour la réception.....	912
Tableau 152 – Table de vérité pour un SyncID pour la transmission	913
Tableau 153 – Primitives distantes émises ou reçues par SCHEDULER	913
Tableau 154 – Primitives locales émises ou reçues par SCHEDULER	914
Tableau 155 – Table d'états SCHEDULER.....	915
Tableau 156 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SCHEDULER.....	917
Tableau 157 – Table de vérité pour RxPeriodChecker d'un port	917
Tableau 158 – Table de vérité pour TxPeriodChecker d'un port	917
Tableau 159 – Syntaxe RTC APDU.....	918
Tableau 160 – Substitutions RTC.....	919
Tableau 161 – Différence CycleCounter.....	921
Tableau 162 – DataStatus.State	923
Tableau 163 – DataStatus.Redundancy	923
Tableau 164 – DataStatus.DataValid	923
Tableau 165 – DataStatus.ProviderState	924
Tableau 166 – DataStatus.StationProblemIndicator	924
Tableau 167 – DataStatus.Ignore d'une trame	924
Tableau 168 – DataStatus.Ignore d'une sous-trame	924
Tableau 169 – TransferStatus pour RT_CLASS_3	925
Tableau 170 – SFPosition.Position	926
Tableau 171 – SFPosition.Reserved	927
Tableau 172 – SFDataLength	927
Tableau 173 – Différence entre SFCycleCounter	928
Tableau 174 – IOxS.Extension.....	929
Tableau 175 – IOxS.Instance.....	929

Tableau 176 – IOxS.DataState.....	930
Tableau 177 – Primitives distantes émises ou reçues par PPM.....	932
Tableau 178 – Primitives locales émises ou reçues par PPM.....	933
Tableau 179 – Table d'états PPM.....	935
Tableau 180 – Fonctions, Macros, Temporisateurs et Variables utilisés par le PPM.....	936
Tableau 181 – Table de vérité utilisée par le PPM pour TxOption.....	936
Tableau 182 – Primitives distantes émises ou reçues par une CPM.....	939
Tableau 183 – Primitives locales émises ou reçues par CPM.....	939
Tableau 184 – Table d'états CPM.....	941
Tableau 185 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CPM.....	944
Tableau 186 – Table de vérité utilisée par le CPM pour RxOption.....	945
Tableau 187 – Table de vérité pour une trame utilisant RT_CLASS_x.....	946
Tableau 188 – Table de vérité pour une trame utilisant RT_CLASS_UDP.....	946
Tableau 189 – Table de vérité pour C_SDU.....	947
Tableau 190 – Table de vérité pour l'agencement de DHT et données.....	947
Tableau 191 – Table de vérité pour la sous-trame – vérification de la trame.....	947
Tableau 192 – Table de vérité pour la sous-trame – vérification de la sous-trame.....	948
Tableau 193 – Table de vérité pour la sous-trame – vérification des données de la sous-trame.....	948
Tableau 194 – Table de vérité de la sous-trame – DHT et données.....	948
Tableau 195 – Syntaxe RTA APDU.....	949
Tableau 196 – Substitutions RTA.....	949
Tableau 197 – PDUType.Type.....	951
Tableau 198 – PDUType.Version.....	951
Tableau 199 – AddFlags.WindowSize.....	951
Tableau 200 – AddFlags.TACK.....	952
Tableau 201 – SendSeqNum.....	952
Tableau 202 – AckSeqNum.....	952
Tableau 203 – VarPartLen.....	953
Tableau 204 – Primitives distantes émises ou reçues par APMS.....	956
Tableau 205 – Primitives locales émises ou reçues par APMS.....	956
Tableau 206 – Table d'états APMS.....	958
Tableau 207 – Fonctions, Macros, Temporisateur et Variables utilisés par l'APMS.....	960
Tableau 208 – Primitives distantes émises ou reçues par APMR.....	963
Tableau 209 – Primitives locales émises ou reçues par APMR.....	963
Tableau 210 – Table d'états APMR.....	965
Tableau 211 – Fonctions, Macros, Temporisateurs et Variables utilisés par l'APMR.....	967
Tableau 212 – TagControllInformation.Priority en fonction des flux.....	967
Tableau 213 – Limite inférieure des fragments.....	970
Tableau 214 – Syntaxe FRAG APDU.....	971
Tableau 215 – Substitutions FRAG.....	971
Tableau 216 – FragDataLength.....	972
Tableau 217 – FragStatus.FragmentNumber.....	972

Tableau 218 – FragStatus.Reserved	972
Tableau 219 – FragStatus.MoreFollows	973
Tableau 220 – Primitives distantes émises ou reçues par FRAG_D	973
Tableau 221 – Primitives locales émises ou reçues par FRAG_D	973
Tableau 222 – Table d'états FRAG_D (dynamique)	974
Tableau 223 – Fonctions, Macros, Temporisateurs et Variables utilisés par le FRAG_D (dynamique)	976
Tableau 224 – Primitives distantes émises ou reçues par FRAG_S	977
Tableau 225 – Primitives locales émises ou reçues par FRAG_S	977
Tableau 226 – Table d'états FRAG_S (statique)	978
Tableau 227 – Fonctions, Macros, Temporisateurs et Variables utilisés par le FRAG_S (statique)	980
Tableau 228 – Primitives distantes émises ou reçues par DEFRAG	980
Tableau 229 – Primitives locales émises ou reçues par DEFRAG	981
Tableau 230 – Table d'états DEFRAG	982
Tableau 231 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DEFRAG	982
Tableau 232 – Table de vérité pour DefragGuard – premier fragment	983
Tableau 233 – Table de vérité pour DefragGuard – fragment suivant	983
Tableau 234 – Table de vérité pour DefragGuard – dernier fragment	983
Tableau 235 – syntaxe RPC APDU	984
Tableau 236 – Substitutions RPC	984
Tableau 237 – RPCVersion	985
Tableau 238 – RPCPacketType	985
Tableau 239 – RPCFlags	986
Tableau 240 – RPCFlags2	986
Tableau 241 – RPCDRep.Character et IntegerEncoding	987
Tableau 242 – RPCDRep.Octet2 – Représentation à virgule flottante	987
Tableau 243 – RPCObjectUUID.Data4	988
Tableau 244 – RPCObjectUUID pour PNIO	988
Tableau 245 – RPCObjectUUID pour PNIO avec plusieurs interfaces	988
Tableau 246 – RPCInterfaceUUID for PNIO	989
Tableau 247 – RPCInterfaceUUID pour le mappeur de point d'extrémité RPC	989
Tableau 248 – RPCInterfaceVersion.Major	990
Tableau 249 – RPCInterfaceVersion.Minor	990
Tableau 250 – RPCOperationNmb (appareil E/S, appareil de commande E/S et superviseur E/S)	991
Tableau 251 – RPCOperationNmb pour le mappeur de point d'extrémité	991
Tableau 252 – RPCDataRepresentationUUID – valeurs définies	993
Tableau 253 – RPCInquiryType	995
Tableau 254 – RPCEPMapStatus	997
Tableau 255 – Valeurs de NCAFaultStatus	999
Tableau 256 – Valeurs de NCAResjectStatus	1000
Tableau 257 – Primitives distantes émises ou reçues par RPC	1001
Tableau 258 – Primitives locales émises ou reçues par RPC	1001

Tableau 259 – Syntaxe LLDP APDU	1002
Tableau 260 – Substitutions LLDP	1003
Tableau 261 – LLDP_ChassisID conjointement avec MultipleInterfaceMode.NameOfDevice et NameOfStation	1004
Tableau 262 – LLDP_PortID conjointement avec MultipleInterfaceMode.NameOfDevice ...	1004
Tableau 263 – LLDP_PNIO_SubType	1005
Tableau 264 – PTCP_PortRxDelayLocal	1005
Tableau 265 – PTCP_PortRxDelayRemote	1005
Tableau 266 – PTCP_PortTxDelayLocal	1005
Tableau 267 – PTCP_PortTxDelayRemote	1006
Tableau 268 – CableDelayLocal	1006
Tableau 269 – RTClass2_PortStatus.State avec ARProperties.StartupMode == Legacy ...	1006
Tableau 270 – RTClass2_PortStatus.State avec ARProperties.StartupMode == Advanced	1006
Tableau 271 – RTClass3_PortStatus.State	1007
Tableau 272 – RTClass3_PortStatus.Fragmentation	1007
Tableau 273 – RTClass3_PortStatus.PreambleLength	1007
Tableau 274 – Table de vérité pour raccourcir le préambule	1008
Tableau 275 – RTClass3_PortStatus.Optimized	1008
Tableau 276 – MRRT_PortStatus.State	1009
Tableau 277 – IRDataUUID	1009
Tableau 278 – LLDP_RedOrangePeriodBegin.Offset	1009
Tableau 279 – LLDP_RedOrangePeriodBegin.Valid	1009
Tableau 280 – LLDP_OrangePeriodBegin.Offset	1010
Tableau 281 – LLDP_OrangePeriodBegin.Valid avec ARProperties.StartupMode == Legacy	1010
Tableau 282 – LLDP_OrangePeriodBegin.Valid avec ARProperties.StartupMode == Advanced	1010
Tableau 283 – LLDP_GreenPeriodBegin.Offset	1011
Tableau 284 – LLDP_GreenPeriodBegin.Valid	1011
Tableau 285 – LLDP_LengthOfPeriod.Length	1011
Tableau 286 – LLDP_LengthOfPeriod.Valid	1011
Tableau 287 – Entrées FDB à diffusion unique	1014
Tableau 288 – Entrées FDB multidiffusion	1014
Tableau 289 – Entrée FDB Multidiffusion	1015
Tableau 290 – Primitives distantes émises ou reçues par MAC_RELAY	1016
Tableau 291 – Primitives locales émises ou reçues par MAC_RELAY	1016
Tableau 292 – Fonctions, Macros, Temporisateurs et Variables utilisés par le MAC_RELAY	1016
Tableau 293 – Primitives distantes émises ou reçues par RTC3PSM	1017
Tableau 294 – Primitives locales émises ou reçues par RTC3PSM	1017
Tableau 295 – Table d'états RTC3PSM	1018
Tableau 296 – Fonctions, Macros, Temporisateurs et Variables utilisés par le RTC3PSM	1018
Tableau 297 – Table de vérité pour le RTC3PSM	1020

Tableau 298 – RXBeginEndAssignment et TXBeginEndAssignment.....	1020
Tableau 299 – Table de fonction d'événement	1022
Tableau 300 – Primitives distantes émises ou reçues par RED_RELAY	1022
Tableau 301 – Primitives locales émises ou reçues par RED_RELAY	1023
Tableau 302 – Table d'états RED_RELAY	1024
Tableau 303 – Fonctions, Macros, Temporisateurs et Variables utilisés par le RED_RELAY.....	1025
Tableau 304 – Table de vérité pour le RedGuard avec vérification complète.....	1025
Tableau 305 – Table de vérité pour le RedGuard avec vérification réduite	1026
Tableau 306 – Table de vérité pour le RedGuard avec vérification minimale.....	1026
Tableau 307 – Primitives distantes émises ou reçues par DFP_RELAY	1028
Tableau 308 – Primitives locales émises ou reçues par DFP_RELAY	1029
Tableau 309 – Table d'états DFP_RELAY	1030
Tableau 310 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY	1030
Tableau 311 – Table de vérité pour DFPGuard	1031
Tableau 312 – Primitives distantes émises ou reçues par DFP_RELAY_INBOUND.....	1031
Tableau 313 – Primitives locales émises ou reçues par DFP_RELAY_INBOUND	1031
Tableau 314 – Table d'états DFP_RELAY_INBOUND	1032
Tableau 315 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY_INBOUND	1033
Tableau 316 – Table de vérité pour InboundGuard – vérification de trame	1033
Tableau 317 – Table de vérité pour InboundGuard – vérification de sous-trame.....	1034
Tableau 318 – Table de vérité pour InboundGuard – vérification des données de la sous-trame	1034
Tableau 319 – Table de vérité pour InboundGuard – vérification complète.....	1034
Tableau 320 – Primitives distantes émises ou reçues par DFP_RELAY_IN_STORAGE	1035
Tableau 321 – Primitives locales émises ou reçues par DFP_RELAY_IN_STORAGE.....	1035
Tableau 322 – Table d'états DFP_RELAY_IN_STORAGE	1036
Tableau 323 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY_IN_STORAGE	1038
Tableau 324 – Primitives distantes émises ou reçues par DFP_RELAY_OUTBOUND	1039
Tableau 325 – Primitives locales émises ou reçues par DFP_RELAY_OUTBOUND	1039
Tableau 326 – Table d'états DFP_RELAY_OUTBOUND.....	1041
Tableau 327 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY_OUTBOUND.....	1042
Tableau 328 – Table de vérité pour l'OutboundGuard – vérification de la trame	1042
Tableau 329 – Table de vérité pour l'OutboundGuard – vérification de la sous-trame.....	1042
Tableau 330 – Primitives distantes émises ou reçues par MUX.....	1044
Tableau 331 – Primitives locales émises ou reçues par MUX.....	1044
Tableau 332 – Table d'états MUX	1045
Tableau 333 – Fonctions, Macros, Temporisateurs et Variables utilisés par le MUX.....	1047
Tableau 334 – Table de vérité pour FrameSizeFits	1047
Tableau 335 – Table de vérité pour StateChecker.....	1048
Tableau 336 – Primitives distantes émises ou reçues par DEMUX	1048

Tableau 337 – Primitives locales émises ou reçues par DEMUX	1049
Tableau 338 – Table d'états DEMUX	1050
Tableau 339 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DEMUX	1052
Tableau 340 – Syntaxe IP/UDP APDU	1052
Tableau 341 – Substitutions IP/UDP	1053
Tableau 342 – UDP_SrcPort	1053
Tableau 343 – UDP_DstPort	1054
Tableau 344 – IP_DstIPAddress	1054
Tableau 345 – IP Multicast DstIPAddress conformément à RFC 2365	1054
Tableau 346 – Primitives distantes émises ou reçues par DNS	1056
Tableau 347 – Primitives locales émises ou reçues par DNS	1056
Tableau 348 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DNS	1056
Tableau 349 – Primitives distantes émises ou reçues par DHCP	1057
Tableau 350 – Primitives locales émises ou reçues par machines	1057
Tableau 351 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DHCP	1057
Tableau 352 – Numéro d'entreprise	1058
Tableau 353 – Primitives distantes émises ou reçues par LMPM	1060
Tableau 354 – Primitives locales émises ou reçues par LMPM	1062
Tableau 355 – Table d'états LMPM	1062
Tableau 356 – Fonctions, Macros, Temporisateurs et Variables utilisés par le LMPM	1064
Tableau 357 – Substitutions d'IO APDU	1064
Tableau 358 – BlockType	1083
Tableau 359 – BlockVersionHigh	1094
Tableau 360 – BlockVersionLow	1094
Tableau 361 – AlarmType	1094
Tableau 362 – AlarmSpecifier.SequenceNumber	1096
Tableau 363 – Différence AlarmSpecifier.SequenceNumber	1097
Tableau 364 – AlarmSpecifier.ChannelDiagnosis	1098
Tableau 365 – AlarmSpecifier.ManufacturerSpecificDiagnosis	1098
Tableau 366 – AlarmSpecifier.SubmoduleDiagnosisState	1098
Tableau 367 – AlarmSpecifier.ARDiagnosisState	1099
Tableau 368 – API	1099
Tableau 369 – SlotNumber	1100
Tableau 370 – SubslotNumber	1100
Tableau 371 – Expression 1 (spécifique au sous-emplacement)	1101
Tableau 372 – Expression 2 (spécifique à l'emplacement)	1101
Tableau 373 – Expression 3 (spécifique à la relation entre applications)	1102
Tableau 374 – Expression 4 (spécifique à l'API)	1102
Tableau 375 – Expression 5 (spécifique à l'appareil)	1102
Tableau 376 – Groupement de DiagnosisData	1102
Tableau 377 – Index (spécifique à l'utilisateur)	1103
Tableau 378 – Index (spécifique à un sous-emplacement)	1103
Tableau 379 – Index (spécifique à l'emplacement)	1106

Tableau 380 – Index (spécifique à une relation entre applications)	1106
Tableau 381 – Index (spécifique à l'API)	1107
Tableau 382 – Index (spécifique à l'appareil)	1108
Tableau 383 – ARType	1109
Tableau 384 – IOCRMulticastMACAdd utilisant RT_CLASS_UDP	1110
Tableau 385 – IOCRMulticastMACAdd utilisant RT_CLASS_2 ou RT_CLASS_3	1110
Tableau 386 – OUI du Type 10	1111
Tableau 387 – ARProperties.State	1111
Tableau 388 – ARProperties.SupervisorTakeoverAllowed	1112
Tableau 389 – ARProperties.ParameterizationServer	1112
Tableau 390 – ARProperties.DeviceAccess	1112
Tableau 391 – ARProperties.CompanionAR	1112
Tableau 392 – ARProperties.AcknowledgeCompanionAR	1113
Tableau 393 – ARProperties.StartupMode	1113
Tableau 394 – ARProperties.PullModuleAlarmAllowed	1113
Tableau 395 – IOCRProperties.RTClass	1114
Tableau 396 – IOCRTagHeader.IOCRVLANID	1115
Tableau 397 – IOCRTagHeader.IOUserPriority	1115
Tableau 398 – IOCRTType	1115
Tableau 399 – CMInitiatorActivityTimeoutFactor avec ARProperties.DeviceAccess:=0	1116
Tableau 400 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=1 ou ARProperties.StartupMode:=1	1116
Tableau 401 – CMInitiatorTriggerTimeoutFactor	1116
Tableau 402 – LengthIOCS	1117
Tableau 403 – LengthIOPS	1118
Tableau 404 – AlarmCRProperties.Priority	1118
Tableau 405 – AlarmCRProperties.Transport	1118
Tableau 406 – AlarmCRTagHeaderHigh.AlarmCRVLANID	1119
Tableau 407 – AlarmCRTagHeaderHigh.AlarmUserPriority	1119
Tableau 408 – AlarmCRTagHeaderLow.AlarmCRVLANID	1119
Tableau 409 – AlarmCRTagHeaderLow.AlarmUserPriority	1120
Tableau 410 – AlarmSequenceNumber	1120
Tableau 411 – AlarmCRTType	1120
Tableau 412 – RTATimeoutFactor	1121
Tableau 413 – RTARetries	1121
Tableau 414 – AddressResolutionProperties.Protocol	1122
Tableau 415 – AddressResolutionProperties.Factor	1122
Tableau 416 – MCITimeoutFactor	1122
Tableau 417 – VendorIDLow	1123
Tableau 418 – VendorIDHigh	1124
Tableau 419 – ModuleIdentNumber	1124
Tableau 420 – SubmoduleIdentNumber	1124
Tableau 421 – ARUUID	1125

Tableau 422 – ARUUID conjointement avec ARType:= IOCARSR.....	1126
Tableau 423 – TargetARUUID.....	1126
Tableau 424 – ActualLocalTimeStamp	1126
Tableau 425 – LocalTimeStamp.....	1127
Tableau 426 – NumberOfLogEntries	1127
Tableau 427 – EntryDetail.....	1127
Tableau 428 – AdditionalValue1 et AdditionalValue2	1127
Tableau 429 – ControlBlockProperties conjointement avec ControlCommand.ApplicationReady avec ARProperties.StartupMode:=1	1128
Tableau 430 – ControlBlockProperties conjointement avec ControlCommand.ApplicationReady avec ARProperties.StartupMode:=0.....	1128
Tableau 431 – ControlBlockProperties conjointement avec les autres valeurs du champ ControlCommand.....	1128
Tableau 432 – ControlCommand.PrmEnd	1128
Tableau 433 – ControlCommand.ApplicationReady.....	1128
Tableau 434 – ControlCommand.Release	1129
Tableau 435 – ControlCommand.Done	1129
Tableau 436 – ControlCommand.ReadyForCompanion	1129
Tableau 437 – ControlCommand.ReadyForRT_CLASS_3.....	1129
Tableau 438 – ControlCommand.PrmBegin.....	1130
Tableau 439 – DataDescription.Type	1130
Tableau 440 – Valeurs de DataLength	1130
Tableau 441 – Valeurs de SendClockFactor	1131
Tableau 442 – Valeurs de ReductionRatio pour RT_CLASS_1 et RT_CLASS_2.....	1132
Tableau 443 – Valeurs de ReductionRatio pour RT_CLASS_3.....	1132
Tableau 444 – Valeurs de ReductionRatio pour RT_CLASS_UDP.....	1133
Tableau 445 – Valeurs de Phase	1133
Tableau 446 – Valeurs de Sequence.....	1134
Tableau 447 – DataHoldFactor d'une trame	1134
Tableau 448 – DataHoldFactor d'une Sous-trame	1135
Tableau 449 – Valeurs de FrameSendOffset.....	1135
Tableau 450 – ModuleState	1136
Tableau 451 – SubmoduleState.AddInfo	1137
Tableau 452 – SubmoduleState.QualifiedInfo	1137
Tableau 453 – SubmoduleState.MaintenanceRequired	1137
Tableau 454 – SubmoduleState.MaintenanceDemanded.....	1137
Tableau 455 – SubmoduleState.DiagInfo	1138
Tableau 456 – SubmoduleState.ARInfo.....	1138
Tableau 457 – SubmoduleState.IdentInfo	1138
Tableau 458 – SubmoduleState.FormatIndicator	1139
Tableau 459 – SubmoduleState.Detail	1139
Tableau 460 – SubmoduleProperties.Type.....	1140
Tableau 461 – SubmoduleProperties.SharedInput	1140
Tableau 462 – SubmoduleProperties.ReduceInputSubmoduleDataLength	1140

Tableau 463 – SubmoduleProperties.ReduceOutputSubmoduleDataLength	1140
Tableau 464 – SubmoduleProperties.DiscardIOXS	1141
Tableau 465 – SubstitutionMode	1141
Tableau 466 – SubstituteActiveFlag	1142
Tableau 467 – InitiatorUDPRTPort	1142
Tableau 468 – ResponderUDPRTPort	1142
Tableau 469 – InitiatorRPCServerPort	1143
Tableau 470 – ResponderRPCServerPort	1143
Tableau 471 – MaxAlarmDataLength	1143
Tableau 472 – APStructureIdentifier avec API := 0	1144
Tableau 473 – APStructureIdentifier avec API != 0	1144
Tableau 474 – ExtendedIdentificationVersionHigh	1144
Tableau 475 – ExtendedIdentificationVersionLow	1145
Tableau 476 – Valeurs de ErrorCode pour des réponses négatives	1145
Tableau 477 – Valeurs de ErrorDecode	1146
Tableau 478 – Codage d'ErrorCode1 avec ErrorDecode PNIO	1146
Tableau 479 – Codage de ErrorCode2 avec PNIO d'ErrorDecode	1148
Tableau 480 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 1)	1148
Tableau 481 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 2 – acquittement d'alarme)	1152
Tableau 482 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 3 – machines)	1152
Tableau 483 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 4 – appareil de commande E/S)	1154
Tableau 484 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 5 – appareil E/S)	1155
Tableau 485 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 6 – raisons de l'arrêt prématuré)	1156
Tableau 486 – Valeurs d'ErrorCode2 pour ErrorCode1 = RPC	1159
Tableau 487 – Codage d'ErrorCode1 pour ErrorDecode avec la valeur ManufacturerSpecific	1159
Tableau 488 – Codage d'ErrorCode2 pour ErrorDecode avec la valeur ManufacturerSpecific	1160
Tableau 489 – IM_Hardware_Revision	1160
Tableau 490 – IM_SWRevision_Functional_Enhancement	1160
Tableau 491 – IM_SWRevision_Bug_Fix	1161
Tableau 492 – IM_SWRevision_Internal_Change	1161
Tableau 493 – IM_Revision_Counter	1161
Tableau 494 – IM_Profile_ID	1161
Tableau 495 – IM_Profile_Specific_Type conjointement avec IM_Profile_ID:= 0x0000	1162
Tableau 496 – IM_Profile_Specific_Type conjointement avec IM_Profile_ID range 0x0001 – 0xF6FF	1162
Tableau 497 – IM_Version_Major	1162
Tableau 498 – IM_Version_Minor	1162
Tableau 499 – IM_Supported.I&M1	1163

Tableau 500 – IM_Date.....	1164
Tableau 501 – UserStructureIdentifier.....	1165
Tableau 502 – ChannelErrorType	1166
Tableau 503 – ChannelNumber.....	1168
Tableau 504 – ChannelProperties.Type	1168
Tableau 505 – ChannelProperties.Maintenance	1169
Tableau 506 – Combinaisons valides au sein de ChannelProperties	1170
Tableau 507 – Combinaisons valides pour Alarmnotification et RecordDataRead(DiagnosisData)	1170
Tableau 508 – ChannelProperties.Specifier	1171
Tableau 509 – ChannelProperties.Direction	1172
Tableau 510 – ExtChannelErrorType	1172
Tableau 511 – Combinaisons autorisées de ChannelErrorType et d'ExtChannelAddValue	1172
Tableau 512 – ExtChannelErrorType pour ChannelErrorType 0 – 0xFF	1173
Tableau 513 – ExtChannelErrorType pour ChannelErrorType 0x0100 – 0x7FFF	1173
Tableau 514 – ExtChannelErrorType pour ChannelErrorType "Transmission de données impossible"	1173
Tableau 515 – ExtChannelErrorType pour ChannelErrorType "Discordance distante"	1173
Tableau 516 – ExtChannelErrorType pour ChannelErrorType "Discordance de redondance de support"	1174
Tableau 517 – ExtChannelErrorType pour ChannelErrorType "Discordance de Synchronisation" et pour ChannelErrorType "Discordance Temporelle".....	1174
Tableau 518 – ExtChannelErrorType pour ChannelErrorType "Discordance de mode isochrone"	1175
Tableau 519 – ExtChannelErrorType pour ChannelErrorType "Discordance CR multidiffusion"	1175
Tableau 520 – ExtChannelErrorType pour ChannelErrorType "Discordance de fibre optique"	1175
Tableau 521 – ExtChannelErrorType pour ChannelErrorType "Discordance de fonction de composant réseau"	1176
Tableau 522 – ExtChannelErrorType pour ChannelErrorType "Discordance de Fonction de Combinaison de Trame Dynamique"	1176
Tableau 523 – ExtChannelErrorType pour ChannelErrorType "Redondance de Support avec discordance de duplication prévue"	1176
Tableau 524 – ExtChannelErrorType pour ChannelErrorType "Discordance de Redondance du Système"	1177
Tableau 525 – ExtChannelErrorType pour ChannelErrorType "Discordances interfaces multiples"	1177
Tableau 526 – ExtChannelErrorType pour ChannelErrorType "Indication de diagnostic imbriquée"	1178
Tableau 527 – Valeurs pour ExtChannelAddValue	1178
Tableau 528 – Valeurs pour Informations Cumulées	1178
Tableau 529 – Valeurs de "Discordance de fibre optique" – "Bilan de puissance"	1179
Tableau 530 – Valeurs de "Discordance de fonction de composant réseau" – "Trame supprimée"	1179
Tableau 531 – Valeurs de "Discordance distante" – "Discordance Cable Delay homologue"	1179

Tableau 532 – Valeurs pour "Discordance d'interfaces multiples" – "mode MultipleInterfaceMode.NameOfDevice conflictuel"	1180
Tableau 533 – Valeurs pour QualifiedChannelQualifier	1180
Tableau 534 – Valeurs pour MaintenanceStatus	1181
Tableau 535 – URRecordIndex	1182
Tableau 536 – URRecordLength	1182
Tableau 537 – MultipleInterfaceMode.NameOfDevice	1183
Tableau 538 – LineDelay.Value avec LineDelay.FormatIndicator == 0	1184
Tableau 539 – LineDelay.Value avec LineDelay.FormatIndicator == 1	1184
Tableau 540 – LineDelay.FormatIndicator	1185
Tableau 541 – RxPort	1185
Tableau 542 – NumberOfTxPortGroups	1185
Tableau 543 – TxPortEntry	1186
Tableau 544 – FrameDetails.SyncFrame conjointement avec FrameDataProperties.ForwardingMode:= "Absolute mode"	1187
Tableau 545 – FrameDetails.SyncFrame conjointement avec FrameDataProperties.ForwardingMode:= "Relative mode"	1188
Tableau 546 – FrameDetails.MeaningFrameSendOffset	1188
Tableau 547 – FrameDetails.MediaRedundancyWatchDog	1188
Tableau 548 – FrameDataProperties.ForwardingMode	1188
Tableau 549 – FrameDataProperties.FastForwardingMulticastMACAdd	1189
Tableau 550 – FrameDataProperties.FragmentationMode	1189
Tableau 551 – MAUType	1189
Tableau 552 – Valid combinations between MAUType and LinkState	1192
Tableau 553 – CheckSyncMode.CableDelay	1193
Tableau 554 – CheckSyncMode.SyncMaster	1193
Tableau 555 – MAUTypeMode.Check	1193
Tableau 556 – DomainBoundaryIngress	1194
Tableau 557 – DomainBoundaryEgress	1194
Tableau 558 – DomainBoundaryAnnounce	1194
Tableau 559 – MulticastBoundary	1195
Tableau 560 – PeerToPeerBoundary	1195
Tableau 561 – DCPBoundary	1196
Tableau 562 – PreambleLength.Length	1196
Tableau 563 – LinkState.Link	1196
Tableau 564 – LinkState.Port	1197
Tableau 565 – MediaType	1197
Tableau 566 – MaxBridgeDelay	1198
Tableau 567 – NumberOfPorts	1198
Tableau 568 – MaxPortTxDelay	1198
Tableau 569 – MaxPortRxDelay	1198
Tableau 570 – MaxLineRxDelay	1199
Tableau 571 – YellowTime	1200
Tableau 572 – StartOfRedFrameID	1202

Tableau 573 – EndOfRedFrameID	1202
Tableau 574 – Dépendances de StartOfRedFrameID et EndOfRedFrameID	1202
Tableau 575 – NumberOfAssignments	1203
Tableau 576 – NumberOfPhases	1203
Tableau 577 – AssignedValueForReservedBegin	1204
Tableau 578 – AssignedValueForOrangeBegin	1204
Tableau 579 – AssignedValueForReservedEnd	1204
Tableau 580 – Valeurs de RedOrangePeriodBegin	1204
Tableau 581 – Dépendances de RedOrangePeriodBegin, OrangePeriodBegin et GreenPeriodBegin	1205
Tableau 582 – Valeurs d'OrangePeriodBegin avec ARProperties.StartupMode == Legacy	1205
Tableau 583 – Valeurs d'OrangePeriodBegin avec ARProperties.StartupMode == Advanced	1205
Tableau 584 – Valeurs de GreenPeriodBegin	1205
Tableau 585 – EtherType	1206
Tableau 586 – SyncProperties.Role	1206
Tableau 587 – SyncProperties.SyncID	1206
Tableau 588 – ReservedIntervalBegin avec ARProperties.StartupMode == Legacy	1207
Tableau 589 – ReservedIntervalBegin avec ARProperties.StartupMode == Advanced	1207
Tableau 590 – ReservedIntervalEnd avec ARProperties.StartupMode == Legacy	1207
Tableau 591 – ReservedIntervalEnd avec ARProperties.StartupMode == Advanced	1207
Tableau 592 – Dépendances de ReservedIntervalBegin et ReservedIntervalEnd	1207
Tableau 593 – SyncSendFactor	1208
Tableau 594 – PTCPTimeoutFactor	1209
Tableau 595 – PTCPTakeoverTimeoutFactor	1210
Tableau 596 – PTCPMasterStartupTime	1210
Tableau 597 – PLLWindow	1211
Tableau 598 – TimeIObase	1212
Tableau 599 – TimeDataCycle	1213
Tableau 600 – TimeIOInput	1213
Tableau 601 – TimeIOOutput	1213
Tableau 602 – TimeIOInputValid	1214
Tableau 603 – TimeIOOutputValid	1214
Tableau 604 – ControllerApplicationCycleFactor	1214
Tableau 605 – TimePLLWindow	1214
Tableau 606 – TimeMasterPriority1	1216
Tableau 607 – TimeMasterPriority2	1216
Tableau 608 – MRP_RingState	1217
Tableau 609 – MRP_DomainUUID	1217
Tableau 610 – MRP_LengthDomainName	1217
Tableau 611 – MRP_Role	1218
Tableau 612 – MRP_Version	1218
Tableau 613 – MRP_Prio	1218

Tableau 614 – MRP_TOPchgT.....	1218
Tableau 615 – MRP_TOPNRmax.....	1219
Tableau 616 – MRP_TSTshortT.....	1219
Tableau 617 – MRP_TSTdefaultT.....	1219
Tableau 618 – MRP_TSTNRmax.....	1220
Tableau 619 – MRP_LNKdownT.....	1220
Tableau 620 – MRP_LNKupT.....	1220
Tableau 621 – MRP_LNKNRmax.....	1221
Tableau 622 – MRP_Check.MediaRedundancyManager.....	1221
Tableau 623 – MRP_Check.MRP_DomainUUID.....	1221
Tableau 624 – VendorBlockType.....	1222
Tableau 625 – FiberOpticType.....	1222
Tableau 626 – FiberOpticCableType.....	1223
Tableau 627 – FiberOpticPowerBudgetType.Value.....	1223
Tableau 628 – FiberOpticPowerBudgetType.CheckEnable.....	1223
Tableau 629 – NCDropBudgetType.Value.....	1224
Tableau 630 – NCDropBudgetType.CheckEnable.....	1224
Tableau 631 – FSHelloMode.Mode.....	1226
Tableau 632 – FSHelloInterval.....	1226
Tableau 633 – FSHelloRetry.....	1227
Tableau 634 – FSHelloDelay.....	1227
Tableau 635 – FSPParameterMode.Mode.....	1227
Tableau 636 – FSPParameterUUID.....	1228
Tableau 637 – NumberOfSubframeBlocks.....	1228
Tableau 638 – SFIOCRProperties.DistributedWatchDogFactor.....	1228
Tableau 639 – SFIOCRProperties.RestartFactorForDistributedWD.....	1229
Tableau 640 – SFIOCRProperties.DFPMMode.....	1230
Tableau 641 – SFIOCRProperties.DFPDirection.....	1230
Tableau 642 – SFIOCRProperties.DFPRedundantPathLayout.....	1230
Tableau 643 – SFIOCRProperties.SFCRC16.....	1230
Tableau 644 – SubframeData.Position.....	1231
Tableau 645 – SubframeData.DataLength.....	1231
Tableau 646 – Table de fonction d'événement.....	1232
Tableau 647 – SubframeOffset.....	1232
Tableau 648 – Table de fonction d'événement.....	1233
Tableau 649 – SCFEntry.....	1235
Tableau 650 – ACCommunicationProperties.DFP.....	1235
Tableau 651 – ACCommunicationProperties.RTC3.....	1236
Tableau 652 – ACCommunicationProperties.RTCUDP.....	1236
Tableau 653 – ACMinDeviceInterval.....	1236
Tableau 654 – FromOffsetData.....	1237
Tableau 655 – NextOffsetData.....	1237
Tableau 656 – TotalSize.....	1237

Tableau 657 – RedundancyInfo.EndPoint1.....	1237
Tableau 658 – RedundancyInfo.EndPoint2.....	1238
Tableau 659 – Combinaison valide de RedundancyInfo.EndPoint1 et RedundancyInfo.EndPoint2.....	1238
Tableau 660 – SRProperties.InputValidOnBackupAR.....	1238
Tableau 661 – SRProperties.ActivateRedundancyAlarm	1238
Tableau 662 – RedundancyDataHoldFactor	1239
Tableau 663 – NumberOfEntries	1239
Tableau 664 – Vérification d'ArgsLength.....	1240
Tableau 665 – ARBlockReq – vérification de demande	1240
Tableau 666 – IOCRBlockReq – vérification de demande	1241
Tableau 667 – AlarmCRBlockReq – vérification de demande	1245
Tableau 668 – ExpectedSubmoduleBlockReq – vérification de demande	1245
Tableau 669 – PrmServerBlock – vérification de demande.....	1247
Tableau 670 – MCRBlockReq – vérification de demande	1247
Tableau 671 – ARRPCBlockReq – vérification de demande.....	1248
Tableau 672 – IRInfoBlock – vérification de demande.....	1248
Tableau 673 – SRInfoBlock – vérification de demande.....	1249
Tableau 674 – Vérification d'ArgsLength.....	1249
Tableau 675 – ARBlockRes – vérification de réponse.....	1250
Tableau 676 – IOCRBlockRes – vérification de réponse.....	1251
Tableau 677 – AlarmCRBlockRes – vérification de réponse.....	1251
Tableau 678 – ModuleDiffBlock – vérification de réponse	1252
Tableau 679 – ARServerBlockRes – vérification de réponse.....	1253
Tableau 680 – Vérification d'ArgsLength.....	1254
Tableau 681 – ControlBlockConnect(PrmEnd) – vérification de demande	1254
Tableau 682 – ControlBlockPlug(PrmEnd) – vérification de demande	1255
Tableau 683 – ControlBlockConnect(PrmBegin) – vérification de demande.....	1255
Tableau 684 – SubmoduleListBlock – vérification de demande	1256
Tableau 685 – Vérification d'ArgsLength.....	1256
Tableau 686 – ControlBlockConnect – vérification de réponse.....	1257
Tableau 687 – ControlBlockPlug – vérification de réponse.....	1257
Tableau 688 – ControlBlockConnect(PrmBegin) – vérification de réponse	1258
Tableau 689 – Vérification d'ArgsLength.....	1259
Tableau 690 – ControlBlockConnect(AppIRdy) – vérification de demande.....	1259
Tableau 691 – ControlBlockPlug(AppIRdy) – vérification de demande.....	1260
Tableau 692 – Vérification d'ArgsLength.....	1260
Tableau 693 – ControlBlockConnect – vérification de réponse.....	1261
Tableau 694 – ControlBlockPlug – vérification de réponse.....	1261
Tableau 695 – Vérification d'ArgsLength.....	1262
Tableau 696 – ReleaseBlock – vérification de demande.....	1262
Tableau 697 – Vérification d'ArgsLength.....	1263
Tableau 698 – ReleaseBlock – vérification de réponse	1263

Tableau 699 – Vérification d'ArgsLength	1264
Tableau 700 – IODWriteReqHeader – vérification de demande	1265
Tableau 701 – Vérification d'ArgsLength	1265
Tableau 702 – IODWriteResHeader – vérification de réponse	1265
Tableau 703 – Vérification d'ArgsLength	1266
Tableau 704 – Vérification d'ArgsLength	1267
Tableau 705 – Vérification d'ArgsLength	1268
Tableau 706 – IODReadReqHeader – vérification de demande	1268
Tableau 707 – RecordDataReadQuery – vérification de demande	1269
Tableau 708 – Vérification d'ArgsLength	1269
Tableau 709 – IODReadResHeader – vérification de réponse	1270
Tableau 710 – Primitives émises par le Contexte AP (utilisateur FAL) à FSPMDEV	1273
Tableau 711 – Primitives émises par FSPMDEV au Contexte AP (utilisateur FAL)	1275
Tableau 712 – Fonctions, Macros, Temporisateurs et Variables utilisés par le Contexte AP (Utilisateur FAL) vers FSPMDEV	1279
Tableau 713 – Fonctions, Macros, Temporisateurs et Variables utilisés par le FSPMDEV au Contexte AP (utilisateur FAL)	1280
Tableau 714 – Primitives émises par le Contexte AP (utilisateur FAL) à FSPMCTL	1282
Tableau 715 – Primitives émises par le FSPMCTL au contexte AP (utilisateur FAL)	1285
Tableau 716 – Fonctions, Macros, Temporisateurs et Variables utilisés par Contexte AP (utilisateur FAL) au FSPMCTL	1288
Tableau 717 – Fonctions, Macros, Temporisateurs et Variables utilisés par le FSPMCTL au Contexte AP (utilisateur FAL)	1290
Tableau 718 – Primitives distantes émises ou reçues par ALPMI	1293
Tableau 719 – Primitives locales émises ou reçues par ALPMI	1293
Tableau 720 – Table d'états ALPMI	1295
Tableau 721 – Fonctions, Macros, Temporisateurs et Variables utilisés par l'ALPMI	1296
Tableau 722 – Primitives distantes émises ou reçues par ALPMR	1296
Tableau 723 – Primitives locales émises ou reçues par ALPMR	1297
Tableau 724 – Table d'états pour ALPMR	1299
Tableau 725 – Fonctions, Macros, Temporisateurs et Variables utilisés par l'ALPMR	1301
Tableau 726 – Primitives distantes émises ou reçues par CMDEV	1304
Tableau 727 – Primitives locales émises ou reçues par CMDEV	1306
Tableau 728 – Table d'états CMDEV	1308
Tableau 729 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMDEV	1311
Tableau 730 – Primitives distantes émises ou reçues par CMDEV_DA	1313
Tableau 731 – Primitives locales émises ou reçues par CMDEV_DA	1313
Tableau 732 – Table d'états CMDEV_DA	1315
Tableau 733 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMDEV(DA)	1315
Tableau 734 – Primitives distantes émises ou reçues par CMSU	1316
Tableau 735 – Primitives locales émises ou reçues par CMSU	1316
Tableau 736 – Table d'états CMSU	1319
Tableau 737 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMSU	1321
Tableau 738 – Primitives distantes émises ou reçues par CMIO	1322

Tableau 739 – Primitives locales émises ou reçues par CMIO	1322
Tableau 740 – Table d'états CMIO	1323
Tableau 741 – Fonctions utilisées par le CMIO	1325
Tableau 742 – Primitives distantes émises ou reçues par CMWRR	1325
Tableau 743 – Primitives locales émises ou reçues par CMWRR	1327
Tableau 744 – Table d'états CMWRR	1328
Tableau 745 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMWRR....	1330
Tableau 746 – Primitives distantes émises ou reçues par CMRDR.....	1330
Tableau 747 – Primitives locales émises ou reçues par CMRDR.....	1331
Tableau 748 – Table d'états CMRDR	1331
Tableau 749 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMRDR	1332
Tableau 750 – Primitives distantes émises ou reçues par CMSM	1332
Tableau 751 – Primitives locales émises ou reçues par CMSM	1333
Tableau 752 – Table d'états CMSM	1334
Tableau 753 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMSM.....	1335
Tableau 754 – Primitives distantes reçues par CMPBE	1336
Tableau 755 – Primitives locales émises ou reçues par CMPBE	1336
Tableau 756 – Table d'états CMPBE.....	1338
Tableau 757 – Fonctions, macros, temporisateurs et variables utilisés par le CMPBE.....	1340
Tableau 758 – Primitives distantes émises ou reçues par CMDMC	1340
Tableau 759 – Primitives locales émises ou reçues par CMDMC	1340
Tableau 760 – Table d'états CMDMC.....	1342
Tableau 761 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMDMC	1345
Tableau 762 – Primitives distantes émises ou reçues par CMINA	1345
Tableau 763 – Primitives locales émises ou reçues par CMINA	1346
Tableau 764 – Table d'états CMINA.....	1347
Tableau 765 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMINA	1352
Tableau 766 – Valeurs de retour de CheckRPC	1352
Tableau 767 – Primitives distantes émises ou reçues par CMRPC	1353
Tableau 768 – Primitives locales émises ou reçues par CMRPC	1355
Tableau 769 – Table d'états CMRPC	1355
Tableau 770 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMRPC.....	1359
Tableau 771 – Valeurs de retour de CheckRPC	1360
Tableau 772 – Primitives distantes émises ou reçues par CMCTL	1363
Tableau 773 – Primitives locales émises ou reçues par CMCTL.....	1364
Tableau 774 – Table d'états CMCTL	1367
Tableau 775 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CMCTL	1370
Tableau 776 – Primitives distantes émises ou reçues par CTLSM.....	1371
Tableau 777 – Primitives locales émises ou reçues par CTLSM.....	1372
Tableau 778 – Table d'états CTLSM	1372
Tableau 779 – Fonctions, Macros, Temporisateurs et Variables utilisés par CTLSM	1373
Tableau 780 – Primitives distantes émises ou reçues par CTLIO	1374
Tableau 781 – Primitives locales émises ou reçues par CTLIO	1374

Tableau 782 – Table d'états CTLIO.....	1375
Tableau 783 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLIO	1377
Tableau 784 – Primitives distantes reçues par CTLRDI.....	1377
Tableau 785 – Primitives locales émises ou reçues par CTLRDI	1378
Tableau 786 – Table d'états CTLRDI	1379
Tableau 787 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLRDI.....	1380
Tableau 788 – Primitives distantes reçues par CTLRDR	1380
Tableau 789 – Primitives locales émises ou reçues par CTLRDR.....	1381
Tableau 790 – Table d'états CTLRDR.....	1381
Tableau 791 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLRDR	1382
Tableau 792 – Primitives distantes reçues par CTLRPC	1382
Tableau 793 – Primitives locales émises ou reçues par CTLRPC.....	1385
Tableau 794 – Table d'états CTLRPC	1386
Tableau 795 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLRPC	1388
Tableau 796 – Primitives distantes émises ou reçues par CTLSU	1388
Tableau 797 – Primitives locales émises ou reçues par CTLSU.....	1389
Tableau 798 – Table d'états CTLSU.....	1391
Tableau 799 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLSU	1393
Tableau 800 – Primitives distantes émises ou reçues par CTLWRI	1394
Tableau 801 – Primitives locales émises ou reçues par CTLWRI	1394
Tableau 802 – Table d'états CTLWRI.....	1396
Tableau 803 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLWRI	1397
Tableau 804 – Primitives distantes émises ou reçues par CTLWRR.....	1398
Tableau 805 – Primitives locales émises ou reçues par CTLWRR.....	1398
Tableau 806 – Table d'états CTLWRR	1399
Tableau 807 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLWRR...	1400
Tableau 808 – Primitives distantes émises ou reçues par CTLPBE	1401
Tableau 809 – Primitives locales émises ou reçues par CTLPBE	1401
Tableau 810 – Table d'états CTLPBE	1403
Tableau 811 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLPBE....	1405
Tableau 812 – Primitives distantes émises ou reçues par CTLDINA.....	1405
Tableau 813 – Primitives locales émises ou reçues par CTLDINA.....	1406
Tableau 814 – Table d'états CTLDINA	1408
Tableau 815 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CTLDINA ..	1411
Tableau A.1 – Exemples d'établissement de l'AR.....	1414
Tableau B.1 – Exemples d'établissements d'AR compatibles	1423
Tableau I.1 – Régénération de priorité et utilisation de la file d'attente.....	1442
Tableau I.2 – Régénération de priorité – variante 1.....	1442
Tableau I.3 – Régénération de priorité – variante 2.....	1443
Tableau R.1 – Table de vérité.....	1467
Tableau S.1 – Erreur TX	1468
Tableau U.1 – Liste des MIB prises en charge	1470
Tableau V.1 – Contenu des archives.....	1471

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

**RÉSEAUX DE COMMUNICATION INDUSTRIELS –
SPÉCIFICATIONS DES BUS DE TERRAIN –****Partie 6-10: Spécification du protocole de la couche application –
Eléments de type 10****AVANT-PROPOS**

- 1) La Commission Electrotechnique Internationale (CEI) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de la CEI). La CEI a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. A cet effet, la CEI – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de la CEI"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec la CEI, participent également aux travaux. La CEI collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
- 2) Les décisions ou accords officiels de la CEI concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de la CEI intéressés sont représentés dans chaque comité d'études.
- 3) Les Publications de la CEI se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de la CEI. Tous les efforts raisonnables sont entrepris afin que la CEI s'assure de l'exactitude du contenu technique de ses publications; la CEI ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
- 4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de la CEI s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de la CEI dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de la CEI et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
- 5) La CEI elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de la CEI. La CEI n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
- 6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
- 7) Aucune responsabilité ne doit être imputée à la CEI, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de la CEI, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de la CEI ou de toute autre Publication de la CEI, ou au crédit qui lui est accordé.
- 8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.

L'attention est attirée sur le fait que l'utilisation du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle. En tout état de cause, l'engagement de renonciation partielle aux droits de propriété intellectuelle pris par les détenteurs de ces droits autorise l'utilisation d'un type de protocole de couche avec les autres protocoles de couche du même type, ou dans des combinaisons avec d'autres types autorisées explicitement par les détenteurs des droits de propriété intellectuelle pour ce type.

NOTE Les combinaisons de types de protocoles sont spécifiées dans la CEI 61784-1 et la CEI 61784-2.

La Norme internationale CEI 61158-6-10 a été établie par le sous-comité 65C: Réseaux industriels, du comité d'études 65 de la CEI: Mesure, commande et automation dans les processus industriels.

Cette troisième édition annule et remplace la deuxième édition publiée en 2010. Cette édition constitue une révision technique.

Cette édition inclut les modifications techniques majeures suivantes par rapport à l'édition précédente:

- Corrections et améliorations
- Changement de l'intégration MRP en référence MRP
- Intégration de la combinaison de trame dynamique
- Intégration de la fragmentation entre pairs
- Intégration du transfert rapide
- Intégration d'AR de RT_CLASS_3 partagées
- Intégration de blocs spécifiques au vendeur pour la connexion
- Intégration du diagnostic de POF générique
- Intégration de l'autoconfiguration
- Intégration du MRPD de redondance de support sans coupure
- Intégration de la fonction de base de redondance du Système
- Intégration de la fonction de base de Configuration en cours d'exécution
- Intégration de la prise en charge d'interfaces multiples
- Intégration de statistique de port pour le suivi des erreurs
- Intégration de la fonction de base de communication entre contrôleurs
- Optimisation du démarrage et du transfert de RT_CLASS_3
- Optimisation du démarrage isochrone
- Optimisation du temps de démarrage depuis une mise hors tension
- Suppression de MRRT
- Suppression de l'automatisation distribuée
- Mise à jour de LLDP-EXT-MIB

Le texte de la présente norme est issu des documents suivants:

FDIS	Rapport de vote
65C/764/FDIS	65C/774/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de la présente norme.

Cette publication a été rédigée selon les Directives ISO/CEI, Partie 2.

Une liste de toutes les parties de la série CEI 61158, publiées sous le titre général *Réseaux de communication industriels – Spécifications de bus de terrain* peut être consultée sur le site web de la CEI.

Le comité a décidé que le contenu de cette publication ne sera pas modifié avant la date de stabilité indiquée sur le site Web de la CEI sous <http://webstore.iec.ch> dans les données relatives à la publication recherchée. A cette date, la publication sera

- reconduite,
- supprimée,
- remplacée par une édition révisée, ou
- amendée.

IMPORTANT – Le logo "*colour inside*" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

IECNORM.COM : Click to view the full PDF of IEC 61158-6-10:2014

Withdrawing

INTRODUCTION

La présente partie de la CEI 61158 appartient à la série de normes visant à faciliter l'interconnexion des composants du système d'automatisation. Elle est liée aux autres normes de la série telle que définie par le modèle de référence de bus de terrain "à trois couches" décrits dans la CEI 61158-1.

Le protocole d'application assure le service d'application à l'aide des services disponibles dans la liaison de données ou une autre couche immédiatement inférieure. La présente norme a pour principal objet de préciser un ensemble de règles de communication, exprimées sous la forme de modes opératoires que doivent réaliser des entités d'application homologues au moment de la communication. Ces règles de communication ont pour vocation de fournir une base de développement stable visant à atteindre différents objectifs:

- guider les implémenteurs et les concepteurs;
- réaliser les essais et acquérir l'équipement;
- dans le cadre d'un accord d'intégration des systèmes dans l'environnement de systèmes ouverts;
- dans le cadre d'une meilleure compréhension des communications à temps critique au sein de l'OSI.

La présente norme porte en particulier sur la communication et l'interfonctionnement des capteurs, des effecteurs et d'autres appareils d'automatisation. Grâce à la présente norme associée à d'autres normes des modèles de référence OSI ou de bus de terrain, des systèmes par ailleurs incompatibles peuvent fonctionner ensemble, quelle que soit leur combinaison.

NOTE L'attention est attirée sur le fait que l'utilisation du type de protocole associé est restreinte par les détenteurs des droits de propriété intellectuelle correspondants. Dans tous les cas, l'engagement de renonciation partielle aux droits de propriété intellectuelle, pris par le(s) détenteur(s) de ces droits, autorise l'utilisation d'un type de protocole de couche Liaison de données particulier avec des protocoles de couche physique et de couche Application dans les combinaisons de Types explicitement spécifiées dans la série CEI 61784. L'utilisation du ou des types de protocole(s) dans d'autres combinaisons peut impliquer d'obtenir l'autorisation de leurs détenteurs de droit de propriété intellectuelle respectifs.

La Commission Electrotechnique Internationale (CEI) attire l'attention sur le fait qu'il est déclaré que la conformité avec le présent document peut impliquer l'utilisation d'un brevet intéressant les éléments de Type 10 et éventuellement d'autres types comme suit:

Les droits de propriété intellectuelle du Type 10 ont été annoncés par [SI]:

Publication WO	Titre (WO)
WO 02/043336 EP 1388238	System and method for parallel transfer of real-time critical and non-real-time critical data via switchable data networks, particularly Ethernet
WO 02/076033 EP 1368935	Synchronous clocked communication system with decentralized input/output modules and methods for integrating decentralized input/output modules in such a system
WO 03/028258 EP 1430628	Method for synchronizing nodes of a communication system
WO 03/028259 EP 1430627	Communications system and method for synchronizing a communications cycle
WO 04/030284 EP 1540895	Method for permanent redundant transmission of data telegrams in communication systems
EP 1558002	Method for assigning an IP address to a device

La CEI ne prend pas position quant à la preuve, à la validité et à la portée de ces droits de propriété.

Les détenteurs de ces droits de propriété ont donné l'assurance à la CEI qu'ils consentent à négocier des licences avec des demandeurs du monde entier, soit sans frais soit à des termes et conditions raisonnables et non discriminatoires. À ce propos, la déclaration des détenteurs des droits de propriété est enregistrée à la CEI. Des informations peuvent être demandées à:

[SI]: Siemens AG
CT IP L&T
Otto-Hahn-Ring 6
D-81739 Munich
Allemagne

L'attention est d'autre part attirée sur le fait que certains des éléments du présent document peuvent faire l'objet de droits de propriété autres que ceux qui ont été mentionnés ci-dessus. La CEI ne saurait être tenue pour responsable de l'identification de ces droits de propriété en tout ou partie.

L'ISO (www.iso.org/patents) et la CEI (<http://patents.iec.ch>) maintiennent des bases de données, consultables en ligne, des droits de propriété pertinents à leurs normes. Les utilisateurs sont encouragés à consulter ces bases de données pour obtenir l'information la plus récente concernant les droits de propriété.

RÉSEAUX DE COMMUNICATION INDUSTRIELS – SPÉCIFICATIONS DES BUS DE TERRAIN –

Partie 6-10: Spécification du protocole de la couche application – Éléments de type 10

1 Domaine d'application

1.1 Généralités

La couche application de bus de terrain (Fieldbus Application Layer - FAL) permet aux programmes utilisateur d'accéder à l'environnement de communication de bus de terrain. A cet égard, la FAL peut être perçue comme une "fenêtre entre des programmes d'application correspondants".

La présente norme fournit des éléments communs pour les communications de messagerie en temps critique ou non entre des programmes d'application dans un environnement et avec un matériel d'automation spécifiques aux bus de terrain de Type 10. Le terme "en temps critique" signale l'existence d'une fenêtre temporelle dans laquelle des actions spécifiées doivent être exécutées, avec un niveau de certitude défini. Si les actions spécifiées ne sont pas réalisées dans la fenêtre temporelle, les applications demandant les actions risquent de connaître une défaillance, avec les risques que cela comporte pour les équipements, les usines et éventuellement la vie humaine.

La présente norme définit de manière abstraite les caractéristiques visibles en externe offertes par la couche application de bus de terrain de Type 10 en termes

- a) de syntaxe abstraite définissant les unités de données du protocole de couche application transmises entre les entités d'application de communication,
- b) de syntaxe de transfert définissant les unités de données du protocole de couche application transmises entre les entités d'application de communication,
- c) de diagramme d'états de contexte d'application définissant les caractéristiques du service d'application visibles entre les entités d'application de communication, et
- d) de diagrammes d'états de relations d'applications définissant le comportement de communication observable entre les entités d'application en communication.

La présente norme a pour objet de définir le protocole permettant de

- a) définir la représentation filaire des primitives de service définies dans la CEI 61158-5-10 et
- b) définir les caractéristiques visibles en externe liées à leur transfert.

La présente norme spécifie le protocole de la couche application de bus de terrain de Type 10, conformément au modèle de référence de base OSI (ISO/CEI 7498-1) et à la structure de couche application OSI (ISO/CEI 9545).

1.2 Spécifications

La présente norme a pour principal objectif de préciser la syntaxe et les caractéristiques du protocole de couche application qui transmet les services de couche application définis dans la CEI 61158-5-10.

Elle a également pour objectif de fournir les chemins de migration à partir des protocoles de communication déjà existants. C'est ce dernier objectif qui donne lieu à la diversité des protocoles normalisés de la CEI 61158-6.

1.3 Conformité

La présente norme ne spécifie pas de mises en œuvre individuelles ou des produits. Elle n'impose pas non plus la mise en œuvre d'entités de couche application dans les systèmes d'automatisation industrielle. La conformité est assurée par la mise en œuvre de la présente spécification du protocole de couche application.

2 Références normatives

Les documents suivants sont cités en référence de manière normative, en intégralité ou en partie, dans le présent document et sont indispensables pour son application. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

NOTE Toutes les parties de la série CEI 61158, ainsi que la CEI 61784-1 et la CEI 61784-2 font l'objet d'une maintenance simultanée. Les références croisées à ces documents dans le texte se rapportent par conséquent aux éditions datées dans la présente liste de références normatives.

CEI 61158-1:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 1: Présentation et lignes directrices des séries CEI 61158 et CEI 61784*

CEI 61158-2:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 2: Spécification et définition des services de la couche physique*

CEI 61158-5-10:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Définition des services de la couche application – Éléments de type 10*

CEI 61158-6-3:2014, *Réseaux de communication industriels – Spécifications des bus de terrain – Partie 6-3: Spécification du protocole de la couche application – Éléments de type 3*

IEC 61588, *Precision clock synchronization protocol for networked measurement and control systems* (disponible en anglais seulement)

CEI 62439-2, *Réseaux industriels de communication – Réseaux de haute disponibilité pour l'automatisation – Partie 2: Protocole de redondance du support (MRP)*

ISO/CEI 646:1991, *Technologies de l'information – Jeu ISO de caractères codés à 7 éléments pour l'échange d'information*

ISO/CEI 7498-1, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Modèle de référence de base: Le modèle de base*

ISO/CEI 8822, *Technologies de l'information – Interconnexion de systèmes ouverts – Définition du service de présentation*

ISO/IEC 8824-1, *Information technology – Abstract Syntax Notation One (ASN.1): Specification of basic notation* (disponible en anglais seulement)

ISO/CEI 9545, *Technologies de l'information – Interconnexion de systèmes ouverts (OSI) – Structure de la couche Application*

ISO/CEI 10731, *Technologies de l'information – Interconnexion de systèmes ouverts – Modèle de référence de base – Conventions pour la définition des services OSI*

ISO 8601, *Éléments de données et formats d'échange – Échange d'information – Représentation de la date et de l'heure*

IEEE 754, *IEEE Standard for Floating-Point Arithmetic*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802, *IEEE Standard for Local and metropolitan area networks: Overview and Architecture*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.1AB-2005, *IEEE Standard for Local and metropolitan area networks: Station and Media Access Control Connectivity Discovery*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.1AS, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – IEEE standard for Local and metropolitan area networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.1D, *IEEE Standard for Local and metropolitan area networks – Media access control (MAC) Bridges*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.1Q, *IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.3, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.11, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*, disponible à l'adresse <<http://www.ieee.org>>

IEEE 802.15.1-2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.1: Wireless medium access control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*, disponible à l'adresse <<http://www.ieee.org>>

IETF RFC 768, *User Datagram Protocol*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 791, *Internet Protocol*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 792, *Internet Control Message Protocol*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 826, *An Ethernet Address Resolution Protocol or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 1034, *Domain names – concepts and facilities*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2131, *Dynamic Host Configuration Protocol*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2132, *DHCP Options and BOOTP Vendor Extensions*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2236, *Internet Group Management Protocol, Version 2*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2365, *Administratively Scoped IP Multicast*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2674, *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 2863, *The Interfaces Group MIB*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 3621, *Power Ethernet MIB*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 4361, *Node-specific Client Identifiers for Dynamic Host Configuration Protocol Version Four (DHCPv4)*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 4363, *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering, and Virtual LAN Extensions*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 4836, *Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 5735, *Special-Use IPv4 Addresses*, disponible à l'adresse <<http://www.ietf.org>>

IETF RFC 5890, *Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework*, disponible à l'adresse <<http://www.ietf.org>>

The Open Group — Publication C706, *Technical standard DCE1.1: Remote Procedure Call*, disponible à l'adresse <<http://www.opengroup.org/onlinepubs/9629399/toc.htm>>

3 Termes, définitions, abréviations, symboles et conventions

Pour les besoins du présent document, les termes, définitions, symboles, abréviations et conventions suivants s'appliquent.

3.1 Termes et définitions référencés

3.1.1 Termes de l'ISO/CEI 7498-1

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 7498-1 s'appliquent:

- a) entité d'application
- b) processus d'application
- c) unité de données de protocole d'application

- d) élément de service d'application
- e) appel d'entité d'application
- f) appel de processus d'application
- g) transaction d'application
- h) système ouvert réel
- i) syntaxe de transfert

3.1.2 Termes de l'ISO/CEI 8822

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 8822 s'appliquent:

- a) syntaxe abstraite
- b) contexte de présentation

3.1.3 Termes de l'ISO/CEI 8824-1

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 8824 s'appliquent:

- a) identifiant d'objet
- b) type

3.1.4 Termes de l'ISO/CEI 9545

Pour les besoins du présent document, les termes suivants définis dans l'ISO/CEI 9545 s'appliquent:

- a) association d'applications
- b) contexte d'application
- c) nom de contexte d'application
- d) appel d'entité d'application
- e) type d'entité d'application
- f) appel de processus d'application
- g) type de processus d'application
- h) élément de service d'application
- i) élément de service de contrôle d'application

3.2 Termes et définitions supplémentaires pour la périphérie décentralisée

3.2.1

alarme

activation d'un événement présentant un état critique

3.2.2

acquittement d'alarme

acquittement d'un événement présentant un état critique

3.2.3

objet de données d'alarme

objet(s) représentant des états critiques référencés par l'appareil, l'API, l'emplacement, le sous-emplacement et le type d'alarme

3.2.4

allouer

prendre une ressource dans une zone commune et l'attribuer à l'utilisation exclusive d'une entité particulière

3.2.5

application

fonction ou structure de données pour lesquelles les données sont utilisées ou générées

3.2.6

interopérabilité de couche application

capacité des entités d'application à exécuter des opérations coordonnées et coopératives à l'aide des services de la FAL

3.2.7

objets d'application

classes contenant plusieurs objets permettant de gérer et d'assurer l'échange dynamique des PDU sur le réseau et au sein de l'appareil de réseau

3.2.8

processus d'application

partie d'une application répartie sur un réseau, se trouvant sur un appareil et adressée de manière univoque

3.2.9

identificateur de processus d'application

distingue plusieurs processus d'application utilisés dans un appareil

3.2.10

objet de processus d'application

composant d'un processus d'application identifiable et accessible par une relation entre applications FAL

Note 1 à l'article: Les définitions d'objets de processus d'application se composent d'un ensemble de valeurs destinées aux attributs de leur classe (reportez-vous à la définition "classe d'objets de processus d'application"). L'accès aux définitions d'objet de processus d'application peut se faire à distance à l'aide des services de l'ASE de gestion d'objet FAL. Les services de gestion d'objet FAL peuvent être utilisés pour charger ou mettre à jour des définitions, lire des définitions d'objet ou créer et supprimer de manière dynamique les objets d'application et leurs définitions correspondantes.

3.2.11

classe d'objet de processus d'application

classe d'objets de processus d'application définie en termes d'ensemble de leurs attributs et services accessibles sur le réseau

3.2.12

relation entre applications

association coopérative entre plusieurs appels d'entité d'application pour les besoins de l'échange d'informations et la coordination de leur opération conjointe. Cette relation est activée par l'échange d'unités de données de protocole d'application ou à la suite d'activités de configuration préalable

3.2.13

élément de service d'application de relation entre applications

élément de service d'application offrant des moyens exclusifs d'établissement et de terminaison de toutes les relations entre applications

3.2.14**point d'extrémité de la relation entre applications**

contexte et comportement d'une relation entre applications tels que perçus et gérés par l'un des processus d'application impliqués dans la relation entre applications

Note 1 à l'article: Chaque processus d'application impliqué dans la relation d'applications maintient son propre point d'extrémité de relation d'applications.

3.2.15**attribut**

description d'une caractéristique visible en externe ou d'une fonction d'un objet

Note 1 à l'article: Les attributs d'un objet contiennent des informations sur les portions variables d'un objet. En règle générale, ils donnent des informations d'état ou gouvernent le fonctionnement d'un objet. Les attributs peuvent également avoir un impact sur le comportement d'un objet. Les attributs sont divisés en attributs de classe et en attributs d'instance.

3.2.16**sauvegarde**

état de la relation entre applications E/S, qui indique qu'il est à l'état de veille

3.2.17**comportement**

indication de la réponse d'un objet à des événements particuliers

3.2.18**canal**

représentation d'une liaison physique ou logique unique d'un objet d'application en entrée ou sortie d'un serveur vers le processus afin de prendre en charge l'adressage des informations de diagnostic

Note 1 à l'article: En général, le canal représente un seul connecteur ou pince faisant office d'interface réelle d'un module ou sous-module. Cette référence est utilisée pour identifier les points de défaillance des PDU de diagnostic.

3.2.19**diagnostic du canal**

informations relatives à un élément particulier d'un objet d'application en entrée ou sortie, fournies aux fins de maintenance

EXEMPLE boucle ouverte

3.2.20**classe**

ensemble d'objets représentant le même type de composant système

Note 1 à l'article: Une classe est une généralisation d'un objet, un modèle qui permet de définir des variables et des méthodes. Tous les objets d'une classe sont identiques en forme et en comportement, mais leurs attributs contiennent en général des données différentes.

3.2.21**attributs de classe**

attribut partagé par tous les objets de la même classe

3.2.22**code de classe**

identifiant unique attribué à chaque objet de classe

3.2.23**service spécifique à la classe**

service défini par une classe d'objets particulière permettant d'exécuter une fonction requise qu'un service classique n'assure pas

Note 1 à l'article: Tout objet spécifique à une classe est réservé à l'usage exclusif de la classe d'objets qui le définit.

3.2.24

clair (dégagé)

état de l'appareil de commande E/S, indiquant que l'algorithme de contrôle n'est pas en cours d'exécution

3.2.25

client

- a) objet qui utilise les services d'un autre objet (serveur) pour accomplir une tâche
- b) émetteur d'un PDU auquel un serveur réagit

3.2.26

profil commun

ensemble d'informations et de fonctionnalités indépendantes de l'appareil assurant la cohérence entre tous les appareils

3.2.27

objet de données de communication

objet qui est le paramètre des relations de communication et référencé par l'indice d'appareil, d'emplacement ou de sous-emplacement

3.2.28

contrôle de configuration

comparaison de la structure de l'objet de données E/S prévue du client à la structure de données E/S réelle du serveur lors de la phase de démarrage

3.2.29

défaut de configuration

différence inacceptable entre la structure de l'objet de données E/S prévue et la structure de l'objet de données E/S réelle détectée par le serveur

3.2.30

identifiant de configuration

représentation d'une partie des données E/S d'un module d'entrée et/ou de sortie d'un serveur

3.2.31

consommer

acte de recevoir des données d'un fournisseur

3.2.32

consommateur

nœud ou puits qui reçoit des données d'un fournisseur

3.2.33

gestion de contexte

informations accessibles sur le réseau (objets de communication) prenant en charge la gestion du fonctionnement du système de bus de terrain, y compris la couche application

Note 1 à l'article: La gestion inclut des fonctions telles que la commande, la surveillance et le diagnostic.

3.2.34

chemin de transport

flux unidirectionnel d'APDU dans une relation entre applications

3.2.35**cyclique**

répétitif de manière régulière

3.2.36**cohérence des données**

moyen cohérent de transmission et d'accès de l'objet de données d'entrée ou de sortie entre un client et un serveur et à l'intérieur de ceux-là

3.2.37**appareil**

matériel physique connecté à la liaison

Note 1 à l'article: Un appareil peut contenir plusieurs nœuds.

3.2.38**ID d'appareil**

identification du type d'appareil attribué au fournisseur

3.2.39**profil d'appareil**

ensemble d'informations et de fonctionnalités dépendantes de l'appareil assurant la cohérence entre les appareils analogues de même type

3.2.40**objet de données de diagnostic**

objet(s) contenant des informations de diagnostic référencées par appareil, emplacement, sous-emplacement et indice

3.2.41**informations de diagnostic**

toutes les données disponibles sur le serveur aux fins de maintenance

3.2.42**reconfiguration dynamique**

modification des objets de données E/S sans interruption d'une relation entre applications établie et mise à jour continue des objets de données E/S non modifiés

3.2.43**point d'extrémité**

une des entités de communication impliquée dans une connexion

3.2.44**ingénierie**

terme abstrait caractérisant l'application ou l'appareil client chargé de configurer le système d'automatisation

3.2.45**erreur**

écart ou discordance entre une valeur ou une condition calculée, observée ou mesurée et la valeur ou la condition prescrite ou théoriquement correcte

3.2.46**classe d'erreur**

groupement général des définitions d'erreur connexes et des codes d'erreur correspondants

3.2.47

code d'erreur

identification d'un type particulier d'erreur dans une classe d'erreur

3.2.48

événement

instance d'un changement des conditions

3.2.49

diagnostic du canal étendu

informations relatives à un élément particulier d'un objet d'application spécifique, fournies aux fins de maintenance

EXEMPLE Echec de liaison

3.2.50

trame

unité de transfert de données sur une liaison

3.2.51

objet de données d'identification

objet(s) contenant des informations relatives au fabricant et au type de l'appareil, du module et du sous-module, référencées par l'appareil, l'API, l'emplacement, le sous-emplacement et l'indice

3.2.52

point d'extrémité de relation entre applications implicite

point d'extrémité de relation entre applications défini en local dans un appareil sans l'aide du service create

3.2.53

appel entrant

relation de communication entrante d'un appareil E/S à un appareil de commande E/S

3.2.54

index

adresse d'un objet de données d'enregistrement dans un processus d'application

3.2.55

instance

occurrence physique réelle d'un objet dans une classe, identifiant l'un des objets dans la même classe d'objets

3.2.56

attributs d'instance

attribut unique d'une instance d'objet, qui n'est pas partagé par la classe d'objets

3.2.57

instancié

objet créé dans un appareil

3.2.58

appel

acte d'utiliser un service ou une autre ressource d'un processus d'application

Note 1 à l'article: Chaque invocation représente un fil de commande distinct qui peut être décrit par son contexte. A l'issue du service ou lorsque la ressource est libérée, l'appel cesse d'exister. Un service initié mais qui n'est pas encore terminé est appelé "appel de service en attente". De même, pour les appels de service, un ID d'appel peut

être utilisé pour identifier de manière univoque l'appel de service et le distinguer des autres appels de service en attente.

3.2.59

appareil de commande E/S

appareil de contrôle faisant office de client pour plusieurs appareils E/S (appareils de terrain)

Note 1 à l'article: En général, il s'agit d'un automate programmable ou d'un système à commande distribuée.

3.2.60

objet de données E/S

objet conçu pour être transféré de manière cyclique afin d'être traité et référencé par l'appareil, l'API, l'emplacement et le sous-emplacement

3.2.61

appareil E/S

appareil de terrain qui fait office de serveur pour les opérations d'entrée/sortie (E/S)

3.2.62

serveur de paramètre E/S

serveur de paramètre d'application des appareils E/S (client)

Note 1 à l'article: En général, il s'agit d'un appareil de sauvegarde des données de paramètre et de consignation des modifications en ligne des paramètres de l'appareil.

3.2.63

sous-système E/S

sous-système composé d'un appareil de commande E/S et de tous ses appareils E/S associés

3.2.64

superviseur E/S

appareil technique qui gère la mise en service et le diagnostic d'un système E/S

3.2.65

système E/S

système composé de tous ses sous-systèmes E/S

Note 1 à l'article: Par exemple, un Contrôleur Logique Programmable comprenant plusieurs contrôleurs E/S (interface réseau) contrôle un système E/S composé d'un sous-système E/S pour chaque contrôleur E/S.

3.2.66

mode isochrone

système E/S synchronisé présentant une gigue de moins de 1 μ s

3.2.67

membre

élément d'un attribut structuré comme un élément de matrice

3.2.68

message

synonyme de trame

3.2.69

méthode

synonyme de service opérationnel fourni par l'ASE du serveur et appelé par un client

3.2.70

module

composant matériel ou logique d'un appareil physique

3.2.71

réseau

ensemble de nœuds connectés par un certain type de support de communication, y compris les répéteurs intercalés, les ponts, les routeurs et les passerelles de couche inférieure

3.2.72

objet

représentation abstraite d'un composant particulier dans un appareil, en général un ensemble de données (se présentant sous la forme de variables) et de méthodes (modes opératoires) connexes d'utilisation de ces données qui définit clairement l'interface et le comportement

3.2.73

service spécifique à un objet

service unique à la classe d'objet qui le définit

3.2.74

fonctionnement

état de l'appareil de commande E/S, indiquant que l'algorithme de contrôle fonctionne

3.2.75

appel sortant

relation de communication sortante d'un appareil de commande E/S à un appareil E/S

3.2.76

paquet

trame

3.2.77

homologue

rôle d'un point d'extrémité de relation entre applications dans lequel il est en mesure d'agir à la fois en tant que client et serveur

3.2.78

appareil physique

appareil d'automatisation ou autre appareil de réseau

3.2.79

connexion point à point

connexion entre exactement deux objets d'application

3.2.80

primaire

état de la relation entre applications E/S, qui indique qu'il est à l'état de fonctionnement

Note 1 à l'article: Outre une relation AR d'entrée/sortie primaire, il peut exister une relation AR d'entrée/sortie de secours, par exemple, pour la redondance et la reconfiguration dynamique des données d'entrée/sortie.

3.2.81

fournisseur

nœud ou source qui envoie des données à un ou plusieurs consommateurs

3.2.82

sous-domaine PTCP

certaines quantités de DTE dotés d'horloges synchronisées

3.2.83**diagnostic de canal qualifié**

informations relatives à un élément particulier d'un objet d'application spécifique, fournies aux fins de maintenance avec une sévérité paramétrée

EXEMPLE Surcharge

3.2.84**objet de données d'enregistrement**

objet déjà prétraité et transféré de manière acyclique pour les besoins des informations ou pour un traitement supplémentaire, et référencé par l'appareil, l'API, l'emplacement, le sous-emplacement et l'index

3.2.85**ressource**

capacité de traitement ou d'information

3.2.86**marche**

état de l'appareil de commande E/S, indiquant que l'algorithme de commande est en cours d'exécution

3.2.87**serveur**

- a) rôle d'un AREP (point d'extrémité de relation d'applications) dans lequel il retourne au client qui a lancé la demande une APDU confirmée de réponse de service
- b) objet qui fournit des services à un autre objet (client)

3.2.88**service**

opération ou fonction qu'un objet et/ou une classe d'objets exécutent à la demande d'un autre objet et/ou une autre classe d'objets

3.2.89**emplacement**

adresse d'une unité structurelle à l'intérieur d'un appareil E/S

Note 1 à l'article: Dans un dispositif modulaire, un connecteur fournit généralement une adresse à un module physique. Dans des appareils compacts, un emplacement adresse en général une fonction logique ou un module virtuel.

3.2.90**arrêt**

état de l'appareil de commande E/S, indiquant que l'algorithme de contrôle n'est pas en cours d'exécution

3.2.91**sous-module**

composant matériel ou logique d'un module

3.2.92**sous-emplacement**

adresse d'une unité structurelle à l'intérieur d'un emplacement

Note 1 à l'article: Un sous-emplacement peut adresser une interface physique pour les sous-modules d'un module. En règle générale, un sous-emplacement est le deuxième niveau des données de structure d'un appareil.

3.2.93

ID fournisseur

numéro administratif central permettant d'identifier le fabricant

3.3 Abréviations et symboles

3.3.1 Abréviations et symboles pour la redondance de support

MRC	Media Redundancy Client (Client de redondance de support)
MRM	Media Redundancy Manager (Gestionnaire de redondance de support)
MRP	Media Redundancy Protocol (Protocole de redondance de support)
MRPD	Media Redundancy with Planned Duplication of frames (Redondance de support avec duplication planifiée des trames)

NOTE La Redondance de support avec duplication planifiée de trames est effectuée par l'émetteur à l'aide de différents chemins pour les trames supplémentaires.

3.3.2 Abréviations et symboles pour la périphérie décentralisée

AE	Application Entity (Entité d'application)
AL	Application Layer (Couche application)
ALME	Application Layer Management Entity (Entité de gestion de couche application)
ALP	Application Layer Protocol (Protocole de couche application)
ALPMI	Alarm Protocol Machine Initiator (Initiateur de la machine de protocole d'alarme)
ALPMR	Alarm Protocol Machine Responder (Répondeur de la machine de protocole d'alarme)
AP	Application Process (Processus d'application)
APDU	Application Protocol Data Unit (Unité de données de protocole d'application)
API	Application Process Identifier (Identifiant de processus d'application)
APM	Acyclic Protocol Machine (Machine de protocole acyclique)
APMR	Acyclic Protocol Machine Receiver (Destinataire de machine de protocole acyclique)
APMS	Acyclic Protocol Machine Sender (Expéditeur de machine de protocole acyclique)
AO	Application Object (Objet d'application)
AR	Application Relationship (Relation entre applications)
AREP	Application Relationship End Point (Point d'extrémité de relation entre applications)
ARP	Address Resolution Protocol (Protocole de résolution d'adresse)
ARPM	Address Resolution Protocol Machine (Machine de protocole de résolution d'adresse)
ASCII	American Standard Code for Information Interchange (Code standard américain pour l'échange d'informations)
ASE	Application Service Element (Élément de service d'application)
BLOB	Binary Large Object
BMA	Machine de Protocole BMA (Best-Master-Algorithm)
BMC	Best Master Clock (Horloge meilleur maître)
CLRPC	Connectionless Remote Procedure Call (Appel de procédure distante sans connexion)
CL-RPC	Connectionless Remote Procedure Call (Appel de procédure distante sans connexion)
CM	Context Management (Gestion de Contexte)
CMCTL	Context Management Protocol Machine Controller

	(Contrôleur de machine de protocole de gestion de contexte)
CMDEV	Context Management Protocol Machine Device (Appareil de machine de protocole de gestion de contexte)
CMDEV_DA	Context Management Device Access Protocol Machine Device (Appareil de machine de protocole d'accès à l'appareil de gestion de contexte)
CMDMC	Discovery Multicast Communication Protocol Machine (Machine de protocole de communication multidiffusion de découverte)
CMINA	Context Management IP and Name Assignment Protocol Machine (Machine de protocole d'attribution d'IP et de nom de gestion de contexte)
CMIO	Context Management Input Output Protocol Machine Device (Appareil de machine de protocole d'entrée-sortie de gestion de contexte)
CMPBE	Context Management Prm Begin End Protocol Machine Device (Appareil de machine de protocole des primitives de début et de fin de gestion de contexte)
CMRDR	Context Management Read Record Responder Protocol Machine Device (Appareil de machine de protocole du répondeur de l'enregistrement de la lecture de gestion de contexte)
CMRPC	Context Management RPC Device Protocol Machine (Machine de protocole d'appareil de RPC de gestion de contexte)
CMSM	Context Management Surveillance Protocol Machine Device (Appareil de machine de protocole de surveillance de gestion de contexte)
CMSU	Context Management Startup Protocol Machine Device (Appareil de machine de protocole de démarrage de gestion de contexte)
CMWRR	Context Management Write Record Responder Protocol Machine Device (Appareil de machine de protocole du répondeur de l'enregistrement de l'écriture de gestion de contexte)
CPM	Consumer protocol machine for cyclic transmitted data (Machine de protocole de consommateur pour les données transmises cycliques)
CR	Communication Relationship (Relation de communication)
CREP	Communication Relationship End Point (Point d'extrémité de relation de communication)
TC	Cut Through switching
CTLDINA	Context Management Discovery, IP and Name Assignment (Attribution de découverte, d'IP et de nom de gestion de contexte)
CTLIO	Context Management Input Output Protocol Machine Controller (appareil de commande de machine de protocole d'entrée-sortie de gestion de contexte)
CTLPBE	Context Management Prm Begin End Protocol Machine Controller (appareil de commande de machine de protocole des primitives de début et de fin de gestion de contexte)
CTLRDI	Context Management Read Record Initiator Protocol Machine Controller (appareil de commande de Machine de Protocole de l'initiateur de l'enregistrement de la lecture de gestion de contexte)
CTLRDR	Context Management Read Record Responder Protocol Machine Controller (appareil de commande de machine de protocole du répondeur de l'enregistrement de la lecture de gestion de contexte)
CTLRPC	Context Management RPC Protocol Machine Controller

	(appareil de commande de machine de protocole de RPC de gestion de contexte)
CTLSM	Context Management Surveillance Protocol Machine Controller (appareil de commande de machine de protocole de surveillance de gestion de contexte)
CTLSU	Context Management Startup Protocol Machine Controller (appareil de commande de machine de protocole de démarrage de gestion de contexte)
CTLWRI	Context Management Write Record Initiator Protocol Machine Controller (appareil de commande de machine de protocole de l'initiateur de l'enregistrement de l'écriture de gestion de contexte)
CTLWRR	Context Management Write Record Responder Protocol Machine Controller (appareil de commande de machine de protocole du répondeur de l'enregistrement de l'écriture de gestion de contexte)
DCE	OSF Distributed Computing Environment (Environnement d'informatique distribuée OSF)
DCP	Protocole DCP (Discovery and basic Configuration Protocol) (Protocole de découverte et de configuration de base)
DCPHMCS	DCP Hello Multicast Sender Protocol Machine (Machine de protocole d'émetteur multidiffusion Hello de DCP)
DCPHMRC	DCP Hello Multicast Receiver Protocol Machine (Machine de protocole de destinataire multidiffusion Hello de DCP)
DCPMCR	DCP Multicast Receiver Protocol Machine (Machine de protocole de destinataire multidiffusion de DCP)
DCPMCS	DCP Multicast Sender Protocol Machine (Machine de protocole d'émetteur multidiffusion de DCP)
DCPUCR	DCP Unicast Receiver Protocol Machine (Machine de protocole de destinataire diffusion unique de DCP)
DCPUCS	DCP Unicast Sender Protocol Machine (Machine de protocole d'émetteur diffusion unique de DCP)
DEFRAG	Defragmentation Protocol Machine (Machine de protocole de défragmentation)
DELAY_REQ	Line Delay Request Protocol Machine (Machine de protocole de demande de retard de ligne)
DELAY_RSP	Line Delay Response Protocol Machine (Machine de protocole de réponse de retard de ligne)
DEMUX	Network Access Receiver (Destinataire de l'accès au réseau)
DFP	Dynamic Frame Packing (Combinaison de Trame Dynamique)
DFP_RELAY	DFP Protocol Machine (Machine de Protocole de DFP)
DFP_RELAY_IN_STORAGE	DFP Inbound Storage Protocol Machine (Machine de protocole de stockage à l'entrée de DFP)
DFP_RELAY_INBOUND	DFP Inbound Protocol Machine (Machine de protocole à l'entrée de DFP)
DFP_RELAY_OUTBOUND	DFP Outbound Protocol Machine (Machine de protocole à la sortie de DFP)
DHCP	Dynamic Host Configuration Protocol (protocole DHCP)
DIM	Device Interface Module (Module d'interface d'appareil)
DLC	Data Link Connection (Connexion de liaison de données)
DLL	Data Link Layer (Couche de liaison de données)
DLPDU	Data Link Protocol Data Unit (Unité de données de protocole de liaison de données)
DLSDU	Data Link-service-data-unit (Unité de données de service de liaison de données)
DMPM	DLL Mapping Protocol Machines (Machines de protocole de mapping DLL)
DNS	Domain Name Service (Service de nom de domaine)
DTE	Date Terminal Equipment

FAL	Fieldbus Application Layer (Couche application de bus de terrain)
FDB	Filtering Data Base (Base de données de filtrage)
FIFO	First In First Out (Premier entré, premier sorti)
FRAG	Fragmentation Protocol Machine (Machine de protocole de fragmentation)
FSPM	FAL Service Protocol Machines (Machine de protocole de service FAL)
FSPMCTL	FSPM Controller (appareil de commande FSPM)
FSPMDEV	FSPM Device (Appareil FSPM)
GBSM	Get Best Sync Master Protocol Machine (Machine de protocole de récupération du meilleur maître de synchronisation)
GSDML	General Station Description Markup Language (Langage GSDML)
I&M	Identification and Maintenance Profile (Profil d'identification et de maintenance)
IANA	Internet Assigned Numbers Authority (Autorité chargée de l'assignation des numéros Internet)
ICMP	Protocole ICMP (Internet Control Message Protocol)
ID	Identificateur
CEI	Commission Electrotechnique Internationale
IFW	RT_CLASS_3 Forwarding Protocol Machine (Machine de protocole de transfert de RT_CLASS_3)
IOC	Input Output Controller (appareil de commande d'Entrée-Sortie)
IOCR	IO Communication Relation (Relation de communication Entrée-Sortie)
IOCS	Input Output Object Consumer Status (Statut du consommateur d'objet d'entrée/sortie)
IOD	Input Output Device (Appareil Entrée-Sortie)
IOMCR	IO Multicast Communication Relation (Relation de communication multidiffusion Entrée-Sortie)
IOPS	Input Output Object Provider Status (Etat du fournisseur d'objet d'entrée/sortie)
IOS	Input Output Supervisor (Superviseur d'Entrée-Sortie)
IP	Internet Protocol (Protocole Internet)
IR	Isochronous Relay (Relais isochrone)
IRT	Isochronous Real Time Protocol (Protocole IRT)
ISO	International Organization for Standardization (Organisation Internationale de Normalisation)
IsoM	Isochronous Relay (Mode isochrone)
DEL	Diode électroluminescente
LLDP	Link Layer Discovery Protocol (Protocole LLDP de découverte de couche liaison de données)
LME	Layer Management Entity (Entité de gestion de couche)
LMPPM	Data Link Layer Mapping Protocol Machine (Machine de protocole de mapping de la couche de liaison de données)
lsb	Least Significant Bit (Bit de poids faible)
LT	Longueur/Type
MAC	Media Access Control (Contrôle d'accès aux supports)
MAC_RELAY	Forwarding Protocol Machine (Machine de protocole de transfert)
MCPM	Multicast Consumer protocol machine for cyclic transmitted data (Machine de protocole de client multidiffusion pour des données transmises cycliques)
MCR	Multicast communication relation (Relation de communication multidiffusion)
MPPM	Provider protocol machine for cyclic transmitted data (PPM) using Multicast communication relation (Machine de protocole de fournisseur pour des données transmises

	cycliques (PPM) à l'aide de la relation de communication multidiffusion)
msb	most significant bit (Bit de poids fort)
MUX	Network Access Sender (Emetteur de l'accès au réseau)
NAP	Network Access Point (Point d'accès au réseau)
NCA	Network Computing Architecture (Architecture NCA)
OSF	Open Software Foundation
OSI	Open Systems Interconnect (Interconnexion de systèmes ouverts)
PDU	Protocol Data Unit (Unité de données de protocole)
PL	Physical Layer (Couche physique)
PPM	Provider protocol machine for cyclic transmitted data (Machine de protocole de fournisseur pour les données transmises cycliques)
PTCP	Protocole PTCP (Precision Transparent Clock Protocol)
QoS	Quality of Service (Qualité de service)
RED_RELAY	Machine de protocole de transfert de la Classe 3 en temps réel
RPC	Remote Procedure Call (Appel de procédure distant)
RSM (Reference Site Method)	Best Remote Sync Master Protocol Machine (Machine de Protocole du meilleur maître distant de synchronisation)
RT	Protocole RT ("Real Time Protocol" "Protocole temps réel")
RT_CLASS_UDP	UDP de classe 3 en temps réel (prend en charge le transport de couche 3)
RT_CLASS_1	Classe 1 en temps réel (transport de couche 2)
RT_CLASS_2	Classe 2 en temps réel (transport de couche 2)
RT_CLASS_3	Classe 3 en temps réel (transport de couche 2)
RTA	Protocole RTA (Real Time Protocol Acyclic)
RTC	Protocole RTC (Real Time Protocol Cyclic)
RTC3PSM	Realtime Class 3 Port State Machine (Diagramme d'états du port de classe 3 en temps réel)
S&F	Stocker et transférer
SCHEDULER (PROGRAMMATEUR)	Scheduler Protocol Machine (Machine de protocole de programmeur)
SDU	Service Data Unit (Unité de données de service)
SYN_BMA	Best-Master-Algorithm Protocol Machine (Machine de protocole de l'algorithme de meilleur maître)
SYN_BMA_GBSM	Get Best Sync Master (Récupération du meilleur maître de synchronisation)
SYN_BMA_RSM	Best Remote Sync Master (Meilleur maître distant de synchronisation)
SYN_MPSM	PTCP Master Protocol Machine (Machine de protocole de maître de PTCP)
SYN_SPSM	PTCP Slave Protocol Machine (Machine de protocole d'esclave de PTCP)
SYNC_RELAY	Sync Relay Protocol Machine (Machine de protocole de relais de synchronisation)
TLV	Type/ Longueur/Valeur (règle de codage)
UDP	User Datagram Protocol (Protocole de datagramme utilisateur)
UUID	Universal Unique Identifier (Identifiant unique universel)
VLAN	Virtual Local Area Network (Réseau local virtuel)

3.3.3 Abréviations et symboles pour les services

cnf	Confirmation
ind	Indication
req	Demande
rsp	Réponse

3.4 Conventions

3.4.1 Concept général

La FAL est définie comme un ensemble d'ASE orientés objet. Chaque ASE est spécifiée dans un article distinct. Chaque spécification d'ASE est constituée de trois parties, à savoir sa définition de classe, son service et sa spécification de protocole. Les deux premiers éléments sont contenus dans la CEI 61158-5-10. La spécification des protocoles pour chacun des éléments ASE est définie dans la présente norme.

Les définitions de classe présentent les attributs des classes prises en charge par chaque ASE. Les attributs sont accessibles à partir des instances de la classe qui utilise les services ASE de gestion spécifiés dans la norme CEI 61158-5-10. La spécification de service définit les services fournis par l'élément ASE.

La présente norme utilise les conventions de description données dans l'ISO/CEI 10731.

3.4.2 Conventions pour la périphérie décentralisée

3.4.2.1 Conventions de syntaxe abstraite

3.4.2.1.1 Les PDU sont décrites sous la forme d'octets ou de groupes d'octets

- a) Les groupes d'octets séparés par une virgule apparaissent dans l'ordre dans lequel ils sont transférés
En l'absence d'octets facultatifs, les octets suivants apparaissent sans espace
- b) Si les octets ou groupes d'octets sont placés entre "{ }", l'ordre est aléatoire
- c) Si les octets ou groupes d'octets sont marqués d'un "*", ils peuvent apparaître plus d'une fois.
S'ils sont placés dans une section entre "{ }", ils peuvent apparaître mélangés avec d'autres octets ou groupes d'octets de ladite section.
- d) Les octets peuvent être groupés ou les valeurs attribuées entre "()"
- e) Si les octets ou groupes d'octets sont groupés entre "[]", le groupe peut être ignoré. Cela n'implique pas que la prise en charge de ces octets est facultative.
- f) Des APDU complexes peuvent être générées au moyen de substitutions (sous-structures)
- g) Les sélections exclusives d'octets ou de groupes d'octets sont séparées par "^"

NOTE 1 Exemple de PDU formel

AP_PDU = Octet1, OctetGroup1, [Octet2], [Octet3], {[OctGroup2*], OctetGroup3 ^ Octet4}

A cet égard, les variantes suivantes sont valides (liste non exhaustive):

Variante 1: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2, OctetGroup3

Variante 2: Octet1, OctetGroup1, Octet2, Octet3, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup3

Variante 3: Octet1, OctetGroup1, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup3, OctetGroup2

Variante 4: Octet1, OctetGroup1, OctetGroup2, OctetGroup3, OctetGroup2, OctetGroup2, OctetGroup2, OctetGroup2

Variante 5: Octet1, OctetGroup1, Octet3, Octet4

NOTE 2 L'ordre aléatoire implique de caractériser les groupes d'octets par un en-tête particulier décrit dans les règles de codage.

NOTE 3 La syntaxe APDU de RTA-PDU et RTC-PDU implique qu'une APDU ne contienne pas plus de 1 440 octets au total, conformément à la DLSDU maximale.

NOTE 4 La syntaxe APDU de CL-RPC implique qu'un appareil de commande E/S prenne en charge une taille minimale d'ASDU de 4 096 octets au total et ne dépasse pas $2^{32}-64$ octets en tout. La taille ASDU minimale est déduite de la taille prévue des données de configuration, de paramètre et de diagnostic d'un appareil E/S avancé.

3.4.2.2 Convention de codage des bits et octets réservés

Le terme "réserve" peut être utilisé pour décrire les bits dans des octets ou des octets complets. Il convient de mettre à zéro à tous les bits ou octets réservés côté expéditeur. Ils ne doivent pas être soumis à essai côté destinataire, sauf si cela a été explicitement énoncé ou si les bits ou octets réservés sont vérifiés par un diagramme d'états.

Le terme "réserve" peut également être utilisé pour indiquer que certaines valeurs dans la plage d'un paramètre sont réservées à des extensions ultérieures. Dans ce cas, il convient de ne pas utiliser les valeurs réservées côté expéditeur. Elles ne doivent pas non plus être soumises à essai côté destinataire, sauf si cela a été explicitement énoncé ou si les valeurs réservées sont vérifiées par un diagramme d'états.

3.4.2.3 Conventions de codage commun des octets de champs spécifiques

Les APDU peuvent comporter des champs spécifiques acheminant des informations de manière primitive et condensée. Ces champs doivent être codés dans l'ordre conformément à la Figure 1.

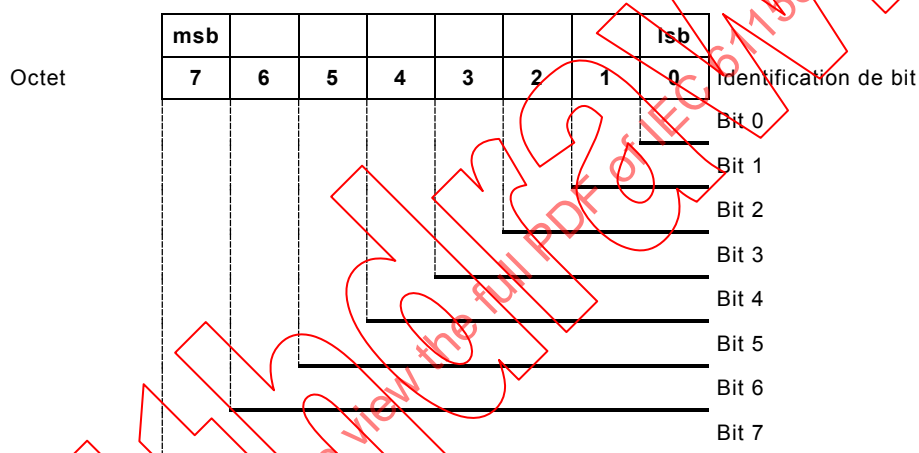


Figure 1 – Structure commune des champs spécifiques

Plusieurs bits peuvent être rassemblés en groupe de bits. Chaque bit ou groupe de bits doit être adressé par son Identification de bit (Bit 0, Bits 1 à 4, par exemple). La position à l'intérieur de l'octet doit être conforme à la Figure ci-dessus. Les noms d'alias peuvent être utilisés pour chaque bit ou groupe de bits ou être marqués comme étant réservés. Le regroupement de bits individuels doit être réalisé dans l'ordre croissant, sans trou. Les valeurs d'un groupe de bits peuvent être représentées sous forme binaire, décimale ou hexadécimale. Cette valeur doit uniquement être valide pour les bits groupés et ne peut représenter que l'ensemble de l'octet si les 8 bits sont regroupés. Les valeurs décimales et hexadécimales doivent être transférées en valeurs binaires, de sorte que le bit portant le numéro le plus élevé du groupe représente le bit de poids fort du bit groupé.

EXEMPLE 1 Description et relation pour l'octet de champ spécifique

Bit 0 : réservé

Bit 1-3: Reason_Code La valeur décimale 2 de Reason_Code indique une erreur générale.

Bit 4-7: doit toujours être défini sur un

L'octet construit conformément à la description ci-dessus se présente comme suit:

(msb) Bit 7 = 1,
 Bit 6 = 1,
 Bit 5 = 1,
 Bit 4 = 1,
 Bit 3 = 0,
 Bit 2 = 1,
 Bit 1 = 0,

(lsb) Bit 0 = 0.

La combinaison de bits "0-1-0" pour les Bits 1-3 équivaut à la valeur décimale 2.

3.4.2.4 Conventions de codage commun d'un champ spécifique composé de deux octets consécutifs

Les APDU peuvent comporter des champs spécifiques acheminant des informations de manière primitive et condensée. Ces champs doivent être codés dans l'ordre conformément à la Figure 2 et à la Figure 3.

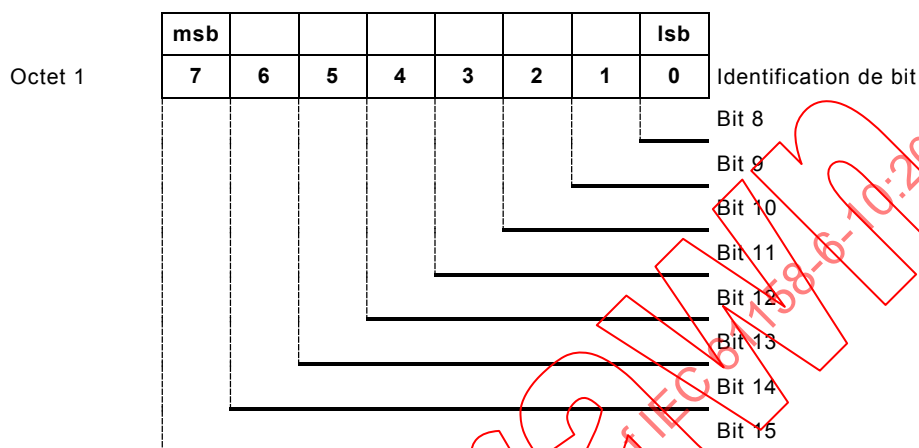


Figure 2 – Structure commune de champs spécifiques pour l'octet 1 (élevé)

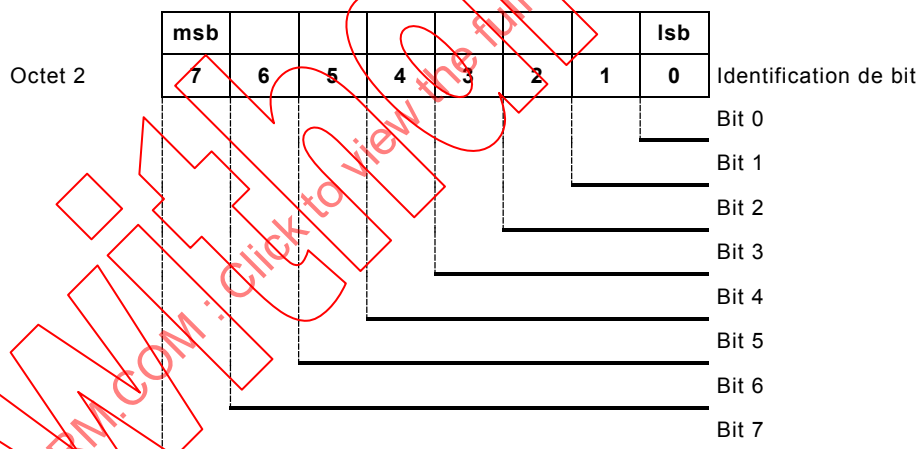
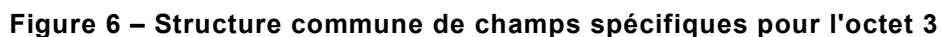
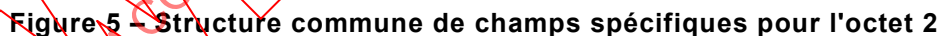


Figure 3 – Structure commune de champs spécifiques pour l'octet 2 (bas)

Plusieurs bits peuvent être rassemblés en groupe de bits. Chaque bit ou groupe de bits doit être adressé par son Identification de bit (Bit 0, Bits 1 à 4, par exemple). La position à l'intérieur de l'octet doit être conforme à la figure ci-dessus. Les noms d'alias peuvent être utilisés pour chaque bit ou groupe de bits ou être marqués comme étant réservés. Le regroupement de bits individuels doit être réalisé dans l'ordre croissant, sans trou. Les valeurs d'un groupe de bits peuvent être représentées sous forme binaire, décimale ou hexadécimale. Cette valeur doit uniquement être valide pour les bits groupés et ne peut représenter que l'ensemble de l'octet si les 16 bits sont regroupés. Les valeurs décimales et hexadécimales doivent être transférées en valeurs binaires, de sorte que le bit portant le numéro le plus élevé du groupe représente le bit de poids fort du bit groupé.

Les APDU peuvent comporter des champs spécifiques acheminant des informations de manière primitive et condensée. Ces champs doivent être codés dans l'ordre conformément à la Figure 4, Figure 5, Figure 6 et à la Figure 7.



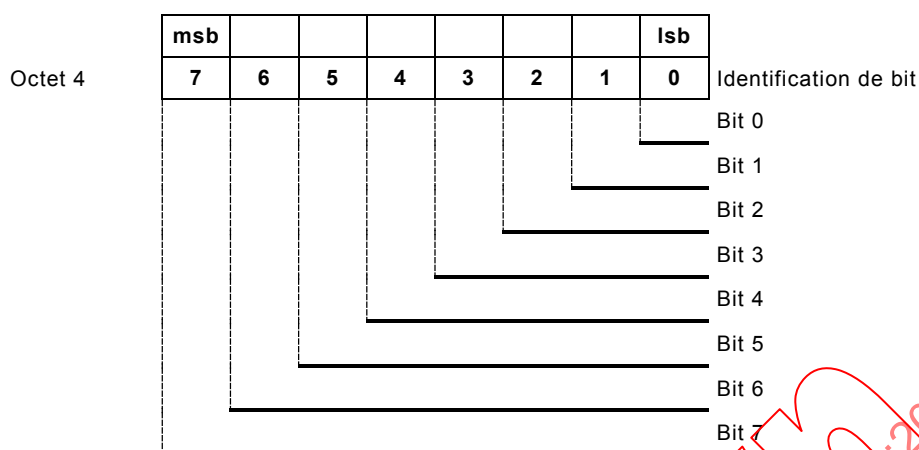


Figure 7 – Structure commune de champs spécifiques pour l'octet 4 (bas)

Les bits peuvent être rassemblés en groupe de bits. Chaque bit ou groupe de bits doit être adressé par son Identification de bit (Bit 0, Bits 1 à 4, par exemple). La position à l'intérieur de l'octet doit être conforme à la figure ci-dessus. Les noms d'alias peuvent être utilisés pour chaque bit ou groupe de bits ou être marqués comme étant réservés. Le regroupement de bits individuels doit être réalisé dans l'ordre croissant, sans trou. Les valeurs d'un groupe de bits peuvent être représentées sous forme binaire, décimale ou hexadécimale. Cette valeur doit uniquement être valide pour les bits groupés et ne peut représenter que l'ensemble de l'octet si les 32 bits sont regroupés. Les valeurs décimales et hexadécimales doivent être transférées en valeurs binaires, de sorte que le bit portant le numéro le plus élevé du groupe représente le bit de poids fort des bits groupés.

3.4.3 Conventions utilisées dans les diagrammes d'états

Les séquences de protocole sont décrites au moyen de diagrammes d'états.

Dans les schémas d'états, les états sont représentés par des cases et les transitions d'états par des flèches. Les noms d'états et de transitions du schéma d'états correspondent aux noms indiqués dans la liste des transitions d'états.

L'initialisation de tous les paramètres (par exemple les valeurs par défaut) d'un diagramme d'états sera effectuée automatiquement lors de l'instanciation du diagramme (nouvelle instance). Transition implicite de POWER-ON (MISE SOUS TENSION) vers le premier état appelant le service INIT. S'il est nécessaire d'avoir différentes instances sur un type de diagramme, différents paramètres sont configurés par le service INIT.

La liste des changements d'état est structurée comme suit (voir aussi le Tableau 1).

- La première ligne contient le nom de la transition.
- La deuxième ligne définit l'état en cours.
- La troisième ligne contient un événement facultatif suivi des conditions commençant par une barre oblique (/) et se terminant par les actions commençant par =>.
- La dernière ligne contient l'état suivant.

Si l'événement se produit et que les conditions sont remplies, la transition est lancée (les actions sont exécutées et le passage à l'état suivant a lieu, par exemple).

La présentation d'une description de diagramme est illustrée dans le Tableau 1. La signification des éléments d'une description de diagramme d'états est présentée dans le Tableau 2.

Tableau 1 – Éléments de description d'un diagramme d'états

#	Etat en cours	Événement ou condition => action	Etat suivant

Tableau 2 – Description des éléments d'un diagramme d'états

Élément de description	Signification
Etat en cours	Nom des états donnés.
Etat suivant	
#	Nom ou numéro de la transition d'état.
Événement	Nom ou description de l'événement.
/Condition	Expression booléenne. Le signe \ qui précède ne fait pas partie de la condition.
=> Action	Liste des affectations et des appels de service ou de fonction. Le signe => qui précède ne fait pas partie de l'action.

Les conventions utilisées dans les diagrammes d'états sont indiquées dans le Tableau 3.

Tableau 3 – Conventions utilisées dans les diagrammes d'états

Convention	Signification
:=	La valeur d'une entité de gauche est remplacée par celle d'une entité de droite. Si une entité de droite est un paramètre, il provient de la primitive identifiée comme un événement d'entrée.
xxx	Nom d'un paramètre. Exemple: Identifieur:= reason signifie que la valeur d'un paramètre reason est attribuée à un paramètre appelé Identifieur.
"xxx"	Indique une valeur fixe. Exemple: Identifieur := "abc" signifie que la valeur "abc" est attribuée à un paramètre appelé Identifieur.
= ou ==	Condition logique indiquant qu'une entité de gauche est égale à une entité de droite.
<	Condition logique indiquant qu'une entité de gauche est inférieure à une entité de droite.
>	Condition logique indiquant qu'une entité de gauche est supérieure à une entité de droite.
<> ou !=	Condition logique indiquant qu'une entité de gauche n'est pas égale à une entité de droite.
>>	Condition sémantique signifiant "le plus récent"
<<	Condition sémantique signifiant "le plus ancien"
&&	Logique AND
	Logique OR
<=	Une condition logique pour indiquer qu'une entité de gauche est inférieure ou égale à une entité de droite.
>=	Une condition logique pour indiquer qu'une entité de gauche est supérieure ou égale à une entité de droite.

Convention	Signification
for (Identifiant := start_value to end_value) actions endfor	Cette construction permet d'exécuter une séquence d'actions dans une boucle d'une transition. La boucle est exécutée pour toutes les valeurs comprises entre start_value et end_value.
If (condition) actions else actions	Cette construction permet d'exécuter d'autres actions en fonction de certaines conditions (qui pourrait être la valeur d'un identifiant ou le résultat d'une action précédente) dans une transition. La partie commençant par "else" peut être ignorée en l'absence d'action et si la condition n'est pas remplie.

Les conventions utilisées pour le service dans les diagrammes d'états sont présentées dans le Tableau 4.

Tableau 4 – Conventions pour les services utilisées dans les diagrammes d'états

Convention	Signification
xxx.req ()	demande conformément à l'ISO/CEI 7498-1. invoque le service et passe tout paramètre requis les paramètres sont omis et énumérés au sein de la description des diagrammes d'états
xxx.ind ()	demande conformément à l'ISO/CEI 7498-1 conseille l'activation d'un service demandé
xxx.rsp (+), xxx.cnf (+)	réponse conformément à l'ISO/CEI 7498-1. peut acquitter positivement ou terminer une action précédemment invoquée par une primitive "request" confirmation conformément à l'ISO/CEI 7498-1 primitive retournée au demandeur pour acquitter positivement ou parachever une action antérieurement invoquée par une primitive "request" les paramètres sont omis et énumérés au sein de la description des diagrammes d'états
xxx.rsp (-), xxx.cnf (-)	réponse conformément à l'ISO/CEI 7498-1. peut acquitter positivement ou terminer une action précédemment invoquée par une primitive "request" confirmation conformément à l'ISO/CEI 7498-1 primitive retournée au demandeur pour acquitter négativement ou parachever une action antérieurement invoquée par une primitive "request" les paramètres sont omis et énumérés au sein de la description des diagrammes d'états
xxx_req (), xxx_cnf (+), xxx_cnf (-)	cette construction est utilisée pour la description des services locaux et confirmés, qui sont transférés entre des diagrammes d'états sont énumérés au sein de la description du diagramme d'états et peuvent être délivrés entre parenthèses de manière facultative
xxx_ind ()	cette construction est utilisée pour la description de services locaux et indique la réception de ce service sont énumérés au sein de la description du diagramme d'états et peuvent être délivrés entre parenthèses de manière facultative
xxx ()	cette construction est utilisée pour la description des fonctions internes du diagramme d'états (locales) sont énumérés au sein de la description du diagramme d'états et peuvent être délivrés entre parenthèses de manière facultative

Convention	Signification
(), (+), (-)	remplacement de la liste des paramètres associés, décrite dans la définition de primitives sauf les services locaux et les fonctions peut délivrer les paramètres entre parenthèses Si le résultat n'est pas nécessaire pour d'autres exécutions, les signes + et – peuvent être ignorés

Il est vivement recommandé aux lecteurs de se reporter aux articles relatifs aux définitions d'attribut AREP et CREP, aux fonctions locales et aux définitions de FAL-PDU pour comprendre les machines de protocole. Les lecteurs sont censés avoir une connaissance suffisante de ces définitions pour les utiliser sans explications complémentaires.

De plus, les éléments de description suivants sont utilisés:

Caractères génériques dans les noms

nom_XXX: "XXX" fait office de chaîne à caractère générique pour tous les noms commençant par "name".

autres: est utilisé en tant que chaîne de caractères génériques pour tous les événements pouvant apparaître dans cet état à l'exception des événements indiqués de façon implicite.

L'utilisation classique d'un caractère générique est un Événement. Dans ce contexte, il existe autant de transitions d'état que d'événements possibles correspondant à ce caractère générique.

Caractères génériques dans les noms d'état

"ANY" est utilisé en tant que chaîne de caractères génériques pour un état en cours afin d'indiquer tous les états de cette table d'états.

"SAME" est utilisé en tant que chaîne de caractères génériques pour un état suivant afin d'indiquer que la transition reste dans le même état. Il ne doit être utilisé qu'en combinaison avec "ANY".

Macro conditionnelle

<CONDITIONAL-MACRO-NAME>

<

CONDITION1: macrobody1

CONDITION2: macrobody2

...

>

CONDITION1, CONDITION2, etc. définit tous les cas possibles pour la macro conditionnelle. CONDITIONAL-MACRO-NAME sert de place réservée pour le "macrocode" qui dépend du résultat de la condition.

Macro de remplacement

<REPLACEMENT-MACRO-NAME>

<

XXX= Name1 : macrobody1

XXX= Name2 : macrobody2

...

>

Name1, Name2, etc. définit tous les cas possibles pour la macro de remplacement. REPLACEMENT-MACRO-NAME sert de place réservée pour le "macrocode" qui dépend de la valeur de l'utilisation en cours du caractère générique XXX.

EXEMPLE

<SERVICE_REQ_PARA>

<

XXX=Read: Para1, Para2

XXX=Write: Para1, Para2, Para3

>

lorsqu'elle est appelée

MSAB_XXX.req(<SERVICE_REQ_PARA>)

donnera

MSAB_Read.req(Para1, Para2) or MSAB_Write.req(Para1, Para2, Para3)

4 Spécification du protocole de couche application pour les protocoles communs

4.1 Description de la syntaxe FAL

4.1.1 Référence de syntaxe abstraite DLPDU

4.1.1.1 Généralités

Le Tableau 5, le Tableau 6 et le Tableau 7 présentent la syntaxe abstraite du DLPDU conformément à l'IEEE 802.3, l'IEEE 802.11 et l'IEEE 802.15.1.

4.1.1.2 IEEE 802.3

Le codage et le décodage des champs du Tableau 5 doivent être conformes à l'IEEE 802.3 pour la DLPDU.

Tableau 5 – Syntaxe IEEE 802.3 pour la DLPDU

Nom du DLPDU	Structure DLPDU
DLPDU	Préambule ^a , StartFrameDelimiter, DestinationAddress, SourceAddress, DLSDU ^b , DLPDU_Padding* ^c , FrameCheckSequence
DLSDU	[VLAN] ^d , LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^e ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU ^e ^ FRAG-PDU
VLAN	LT(=0x8100), TagControlInformation
NOTE Conformément à l'IEEE 802.3, les DLPDU présentent une longueur minimale de 64 octets (autre le préambule et le délimiteur de début de trame).	

- a Le champ contient 7 octets, mais il convient de le diminuer à au moins 1 octet conformément à la présente norme.
- b La taille minimale du DLSDU est de 2 octets.
- c Le nombre d'octets de bourrage doit être compris entre 0 et 46, selon la taille du DLSDU. Voir la taille minimale de la trame dans IEEE 802.3
- d Il est recommandé que le champ VLAN d'un DLPDU RTC soit codé par l'émetteur mais il doit être omis pour RT_CLASS_3.
Il est admis de l'omettre lors de l'acheminement de la trame.
Il est admis que RED_RELAY accepte des trames RT_CLASS_3 avec VLAN.
Le décodeur doit accepter des DLPDU avec ou sans champ VLAN.
Un seul VLAN doit être pris en charge, il est admis de prendre en charge plusieurs VLAN.
- e Le champ VLAN doit être ignoré, sauf pour PTCP-AnnouncePDU.

4.1.1.3 IEEE 802.11

Le codage et le décodage des champs du Tableau 6 doivent être conformes à l'IEEE 802.11 pour la DLPDU.

Tableau 6 – Syntaxe IEEE 802.11 pour la DLPDU

Nom du DLPDU	Structure DLPDU
DLPDU	DLPDU_1 ^ DLPDU_2 ^ DLPDU_3 ^ DLPDU_4
DLPDU_1	FrameControl, DestinationAddress, SourceAddress, BSSID, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_2	FrameControl, DestinationAddress, BSSID, SourceAddress, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_3	FrameControl, BSSID, SourceAddress, DestinationAddress, SequenceControl, [QoSControl], DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_4	FrameControl, ReceiverAddress, TransmitterAddress, DestinationAddress, SequenceControl, SourceAddress, [QoSControl], DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLSDU	DSAP(=0xaa), SSAP(=0xaa), CTRL(=0x03), OUI(=0), [VLAN] ^b , LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU
NOTE 1 Voir l'IEEE 802.11 pour obtenir la définition de FrameControl NOTE 2 Voir l'IEEE 802.11 pour obtenir la définition de BSSID NOTE 3 Voir l'IEEE 802.11 pour obtenir la définition de SequenceControl NOTE 4 Voir l'IEEE 802.11 pour obtenir la définition de QoSControl NOTE 5 Voir l'IEEE 802.11 pour obtenir la définition de ReceiverAddress NOTE 6 Voir l'IEEE 802.11 pour obtenir la définition de TransmitterAddress NOTE 7 L'IEEE 802.11 prend en charge l'utilisation de VLAN conjointement avec FrameClassifier Type 2. Voir IEEE 802.11 pour la définition de FrameClassifier NOTE 8 Voir IEEE 802.11 pour obtenir la définition de DSAP NOTE 9 Voir IEEE 802.11 pour obtenir la définition de SSAP NOTE 10 Voir IEEE 802.11 pour obtenir la définition de CTRL NOTE 11 Voir IEEE 802.11 pour la définition de OUI dans ce contexte	
a	Le nombre d'octets de bourrage doit être compris entre 0 et 46, selon la taille du DLSDU. Voir la taille minimale de la trame dans IEEE 802.3.
d	Il convient que le champ VLAN d'un DLPDU RTC soit codé par l'émetteur mais qu'il soit omis pour RT_CLASS_3. Il est admis qu'il soit omis lors de l'acheminement de la trame mais il doit être omis par le RED_RELAY. Le décodeur doit accepter des DLPDU avec ou sans champs VLAN.

4.1.1.4 IEEE 802.15.1

Le codage et le décodage des champs du Tableau 7 doivent être conformes à l'IEEE 802.15.1 pour la DLPDU.

Tableau 7 – Syntaxe IEEE 802.15.1 pour une DLPDU

Nom du DLPDU	Structure DLPDU
DLPDUHEADER	Access Code, Packet Header, Payload Header, L2CAP Header, BNEPType(0)
DLPDU	DLPDUHEADER, DestinationAddress, SourceAddress, DLSDU ^a , FrameCheckSequence
DLSDU	[VLAN] ^b , LT, SAPDU ^c FIDAPDU
SAPDU	UDP-RTC-PDU ^c UDP-RTA-PDU ^c CL-RPC-PDU ^c LLDP-PDU ^c ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^c RTA-PDU ^c DCP-PDU ^c PTCP-PDU ^c
VLAN	LT(=0x8100), TagControlInformation
NOTE 1 Voir l'IEEE 802.15.1 pour obtenir la définition d'Access Code NOTE 2 Voir l'IEEE 802.15.1 pour obtenir la définition de Packet Header NOTE 3 Voir l'IEEE 802.15.1 pour obtenir la définition de Payload Header NOTE 4 Voir l'IEEE 802.15.1 pour obtenir la définition de L2CAP Header NOTE 5 Voir l'IEEE 802.15.1 pour obtenir la définition de BNEPType	
^a La taille minimale du DLSDU est de 2 octets. ^b Il convient que le champ VLAN d'un DLPDU RTC soit codé par l'émetteur mais qu'il soit omis pour RT_CLASS_3. Il est admis qu'il soit omis lors de l'acheminement de la trame mais il doit être omis par le RED_RELAY. Le décodeur doit accepter des DLPDU avec ou sans champs VLAN. ^c Il convient d'ignorer le champ VLAN.	

4.1.2 Types de données

4.1.2.1 Notation du type Boolean

Boolean ::= BOOLEAN

-- TRUE si la valeur n'est pas nulle.
 -- FALSE si la valeur est nulle.

4.1.2.2 Notation du type Integer

Integer8 ::= INTEGER (-128..+127) -- plage $-2^7 \leq I \leq 2^7-1$

Integer16 ::= INTEGER (-32 768..+32 767) -- plage $-2^{15} \leq I \leq 2^{15}-1$

Integer32 ::= INTEGER -- plage $-2^{31} \leq I \leq 2^{31}-1$

Integer64 ::= INTEGER -- plage $-2^{63} \leq I \leq 2^{63}-1$

4.1.2.3 Notation for the Unsigned type

Unsigned8 ::= INTEGER (0..255) -- plage $0 \leq I \leq 2^8-1$

Unsigned16 ::= INTEGER (0..65 535) -- plage $0 \leq I \leq 2^{16}-1$

Unsigned32 ::= INTEGER -- plage $0 \leq I \leq 2^{32}-1$

Unsigned64 ::= INTEGER -- plage $0 \leq I \leq 2^{64}-1$

4.1.2.4 Notation du type Floating Point

Float32 ::= BIT STRING SIZE (4) -- IEEE 754 Précision simple

Float64 ::= BIT STRING SIZE (8) -- IEEE 754 Précision double

4.1.2.5 Notation du type OctetString

OctetString ::= OCTET STRING -- Pour utilisation générique

4.1.2.6 Notation du type VisibleString

VisibleString ::= VISIBLE STRING -- ISO/CEI 646 – Version de référence internationale sans le caractère "del" (codage 0x7F)

4.1.2.7 Notation du type BinaryDate

BinaryDate ::= OCTET STRING SIZE (7) -- Les formats de date et d'heure ISO 8601 sont recommandés.

4.1.2.8 Notation du type TimeOfDay

TimeOfDay avec indication de date ::= OCTET STRING SIZE (6) -- Les formats de date et d'heure ISO 8601 sont recommandés.

TimeOfDay sans indication de date ::= OCTET STRING SIZE (4) -- Les formats de date et d'heure ISO 8601 sont recommandés.

4.1.2.9 Notation du type TimeDifference

TimeDifference avec indication de date ::= OCTET STRING SIZE (6) -- Les formats de date et d'heure ISO 8601 sont recommandés.

TimeDifference sans indication de date ::= OCTET STRING SIZE (4) -- Les formats de date et d'heure ISO 8601 sont recommandés.

4.2 Syntaxe de transfert

4.2.1 Codage des types de données de base

4.2.1.1 Codage général

- Sauf indication contraire explicite, le codage des valeurs doit être au format grossier.

4.2.1.2 Codage d'une valeur Boolean

- Le codage d'une valeur Boolean doit être primitif. ContentsOctets doit être composé d'un seul octet.
- Si la valeur Boolean est FALSE, ContentsOctets doit être 0 (zéro). Si la valeur Boolean est TRUE, ContentsOctets doit être 0xff.

4.2.1.3 Codage d'une valeur Integer

- Le codage d'une valeur Integer de longueur fixe de type Integer8, Integer16, Integer32 et Integer64 doit être primitif, et ContentsOctets doit être respectivement composé d'exactly un, deux, quatre ou huit octets.
- ContentsOctets doit être un nombre binaire en complément à deux égal à la valeur entière et composé des bits 7 à 0 du premier octet, suivis des bits 7 à 0 du deuxième octet, puis des bits 7 à 0 de chaque octet en augmentant et incluant le dernier octet de ContentsOctets.

NOTE La valeur d'un nombre binaire en complément à deux est déduite de la numérotation des bits de ContentsOctets, en commençant par le bit 0 du dernier octet (bit zéro) et en terminant par le bit 7 du premier. Une valeur numérique 2^N est attribuée à chaque bit, N étant sa position dans la séquence de numérotation ci-dessus. La valeur du nombre binaire en complément à deux est obtenue en ajoutant les valeurs numériques attribuées à chaque bit correspondant aux bits ayant pris la valeur un, à l'exclusion du bit 7 du premier octet, puis en réduisant cette valeur de la valeur numérique attribuée au bit 7 du premier octet, si la valeur un lui a été attribuée.

4.2.1.4 Codage d'une valeur Unsigned

- Le codage d'une valeur Unsigned de longueur fixe de type Unsigned8, Unsigned16, Unsigned32 et Unsigned64 doit être primitif et ContentsOctets doit être composé d'exactly un, deux, quatre ou huit octets, respectivement.
- ContentsOctets doit être un nombre binaire égal à la valeur Unsigned et composé des bits 7 à 0 du premier octet, suivis des bits 7 à 0 du deuxième octet, puis des bits 7 à 0 de chaque octet en augmentant et incluant le dernier octet de ContentsOctets.

NOTE La valeur d'un nombre binaire est déduite de la numérotation des bits de ContentsOctets, en commençant par le bit 0 du dernier octet (bit zéro) et en terminant par le bit 7 du premier. Une valeur numérique 2^N est attribuée à chaque bit, N étant sa position dans la séquence de numérotation ci-dessus. La valeur du nombre binaire est obtenue en ajoutant les valeurs numériques attribuées à chaque bit pour les bits qui sont mis à un.

4.2.1.5 Codage d'une valeur à Virgule Flottante

- Le codage d'une valeur à virgule flottante de longueur fixe de type Float32 et Float64 doit être primitif, et ContentsOctets doit être respectivement composé d'exactly quatre ou huit octets.
- Les ContentsOctets doivent comporter les valeurs à virgule flottante définies conformément à IEEE 754.
Pour Float32, le signe est codé dans le bit 7 du premier octet. Il est suivi de l'exposant, en commençant par le bit 6 du premier octet puis de la mantisse, en commençant par le bit 6 du deuxième octet.
Pour Float64, le signe est codé dans le bit 7 du premier octet. Il est suivi de l'exposant, en commençant par le bit 6 du premier octet puis de la mantisse, en commençant par le bit 3 du deuxième octet.

4.2.1.6 Encodage d'une valeur VisibleString

- Le codage d'une valeur VisibleString de longueur variable doit être primitif.
- Il n'existe pas de champ Length, la longueur étant codée de manière implicite.
- ContentsOctets doit être une séquence d'octets. L'élément le plus à gauche de la chaîne est codé dans le premier octet, suivi du deuxième octet, puis de chaque octet en augmentant et incluant le dernier octet le plus à droite de ContentsOctets.

4.2.1.7 Codage d'une valeur OctetString

- Le codage d'une valeur OctetString de longueur variable doit être primitif.
- Il n'existe pas de champ Length, la longueur étant codée de manière implicite.
- ContentsOctets doit être une séquence d'octets. L'élément le plus à gauche de la chaîne est codé dans le premier octet, suivi du deuxième octet, puis de chaque octet en augmentant et incluant le dernier octet le plus à droite de ContentsOctets.

4.2.1.8 Codage d'une valeur BinaryDate

- Le codage d'une valeur BinaryDate doit être primitif.

- Le champ Length doit indiquer comme nombre binaire le nombre d'octets de la valeur BinaryDate.
- La valeur de la chaîne ContentsOctets doit être égale aux octets de la valeur des données, comme indiqué à la Figure 8.

bits octets	7	6	5	4	3	2	1	0	
1	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	0...59 999 ms
2	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
3	0	0	2^5	2^4	2^3	2^2	2^1	2^0	0...59 min
4	DST	0	0	2^4	2^3	2^2	2^1	2^0	0...23 heures DST: heures d'économie d'énergie
	jour de la semaine			jour du mois			jour de la semaine		0...7 jour de la semaine
5	2^2	2^1	2^0	2^4	2^3	2^2	2^1	2^0	1...31 jour du mois
6	0	0	2^5	2^4	2^3	2^2	2^1	2^0	1...12 mois
7	0	2^6	2^5	2^4	2^3	2^2	2^1	2^0	0 ... 50 années 2000 à 2050 51 ... 99 années 1951 à 1999
msb									lsb

Figure 8 – Codage du type de données BinaryDate

4.2.1.9 Codage d'une valeur TimeOfDay avec et sans indication de date

- L'encodage d'une valeur TimeOfDay, avec et sans valeur d'indication de date, doit être de type primitif.
- Le champ Length doit indiquer comme nombre binaire le nombre d'octets dans la valeur de TimeOfDay with et without date indication (avec et sans indication de date).
- La valeur de ContentsOctets doit être égale à celle des octets dans la valeur de données (voir Figure 9 et Figure 10):

bits octets	7	6	5	4	3	2	1	0	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Nombre de millisecondes depuis minuit
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	nombre de jours depuis le 1984-01-01 T 00:00 Z seulement avec indication de date
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
msb									lsb

Figure 9 – Codage d'une valeur TimeOfDay avec indication de date

bits octets	7	6	5	4	3	2	1	0	
1	0	0	0	0	2^{27}	2^{26}	2^{25}	2^{24}	Nombre de millisecondes depuis minuit
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	msb				lsb				

Figure 10 – Codage d'une valeur TimeOfDay sans indication de date

4.2.1.10 Codage de TimeDifference avec et sans indication de date

- L'encodage d'une valeur TimeDifference, avec et sans valeur d'indication de date, doit être de type primitif.
- Le champ Length doit indiquer comme nombre binaire le nombre d'octets dans la valeur de TimeDifference with et without date indication (avec et sans indication de date).
- La valeur de ContentsOctets doit être égale à celle des octets dans la valeur de données (voir Figure 11 et Figure 12):

bits octets	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Millisecondes
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
5	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	Jours
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	msb				lsb				

Figure 11 – Encodage d'une valeur TimeDifference avec indication de date

bits octets	7	6	5	4	3	2	1	0	
1	2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Millisecondes
2	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
3	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
4	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	msb				lsb				

Figure 12 – Encodage d'une valeur TimeDifference sans indication de date

4.2.1.11 Codage d'une valeur NULL

- Le codage d'une valeur NULL doit être primitif.
- ContentsOctet doit être ignoré.

4.2.1.12 Codage d'une valeur NIL

- Le codage d'une valeur NIL doit être primitif.

- Une valeur nulle doit être attribuée à chaque ContentsOctet.

4.2.1.13 Codage d'un UUID

- Le codage de ce type de données doit être primitif et les ContentOctet doivent être composés exactement de seize octets.
- Pour DCE RPC V1.1, le codage des ContentOctet doit être conforme à la Publication C706 de The Open Group.

4.2.2 Section de codage relative aux champs de base communs

4.2.2.1 Vue d'ensemble

Les champs communs de 4.1.1 font partie de RTC-PDU et de RTA-PDU.

4.2.2.2 Codage du champ SourceAddress du DLPDU

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ SourceAddress doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802, de l'IEEE 802.1D et au Tableau 8.

Tableau 8 – SourceAddress

PDU	Signification
RTC-PDU, RTA-PDU, UDP-RTC-PDU, UDP-RTA-PDU, CL-RPC-PDU, ICMP-PDU, DCP-PDU	L'adresse MAC de l'interface est utilisée
LLDP-PDU, PTCP-PDU	L'adresse MAC du port est utilisée

NOTE 1 L'adresse MAC du port permet d'éviter l'apprentissage erroné de la base de données de filtrage des appareils connectés lorsque l'état sont à l'état BLOCKED.

NOTE 2 L'adresse MAC du port est appelée l'adresse MAC individuelle Bridge Port dans l'IEEE 802.1D.

EXEMPLE Un appareil doté de n ports a besoin d'une adresse MAC d'interface et de n adresses MAC de port.

4.2.2.3 Codage du champ DestinationAddress du DLPDU

4.2.2.3.1 Vue d'ensemble

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ DestinationAddress doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

NOTE L'octet 1 contient le bit d'adresse individuelle/de groupe (bit de poids faible).

4.2.2.3.2 DCP-PDU

Pour les DCP-Multicast-PDU, ce champ doit être codé conformément au Tableau 9, au Tableau 10 et au Tableau 11. Les DCP-Unicast-PDU ne doivent pas utiliser des adresses de multidiffusion ou de diffusion.

Tableau 9 – DCP_MulticastMACAdd pour Identify

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifieur (hexadécimale)	Signification

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-00-00	Avec FrameID=0xFEFE utilisé pour DCP-Identify-ReqPDU

Tableau 10 – DCP_MulticastMACAdd pour Hello

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-00-01	Avec FrameID=0xFEFC utilisé pour DCP-Hello-ReqPDU

Tableau 11 – DCP_MulticastMACAdd

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-00-02 – 00-00-FF	Réservé pour d'autres applications

4.2.2.3.3 PTCP-PDU

Pour les PTCP-PDU, la valeur doit être définie conformément au Tableau 12, au Tableau 13, au Tableau 14, au Tableau 15, au Tableau 16, au Tableau 17, au Tableau 18 et au Tableau 19.

Tableau 12 – PTCP_MulticastMACAdd Page 1

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-01-00 – 00-03-FF	Réservé pour d'autres applications

Tableau 13 – PTCP_MulticastMACAdd Page 2

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-04-00	Conjointement avec PTCP-AnnouncePDU et FrameID (=0xFF00) utilisé pour la synchronisation d'horloge
01-0E-CF	00-04-01 – 00-04-1E	Réservé
01-0E-CF	00-04-1F	Réservé

Tableau 14 – PTCP_MulticastMACAdd Page 3

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-04-20	Conjointement avec PTCP-RTSyncPDU avec suivi et FrameID(=0x0020) utilisé pour la synchronisation de l'horloge de travail
01-0E-CF	00-04-22 – 00-04-3E	Réservé
01-0E-CF	00-04-3F	Réservé

Tableau 15 – PTCP_MulticastMACAdd Page 4

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-04-40	Conjointement avec PTCP-FollowUpPDU et FrameID (=0xFF20) utilisé pour la synchronisation de l'horloge de travail
01-0E-CF	00-04-41 – 00-04-5E	Réservé
01-0E-CF	00-04-5F	Réservé

Tableau 16 – PTCP_MulticastMACAdd Page 5

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-04-60 – 00-04-7F	Réservé pour d'autres applications

Tableau 17 – PTCP_MulticastMACAdd Page 6

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-04-80	Conjointement avec PTCP-RTSyncPDU et FrameID (=0x0080) utilisés pour la synchronisation de l'horloge
01-0E-CF	00-04-81 – 00-04-9E	Réservé
01-0E-CF	00-04-9F	Réservé

Tableau 18 – PTCP_MulticastMACAdd Page 7

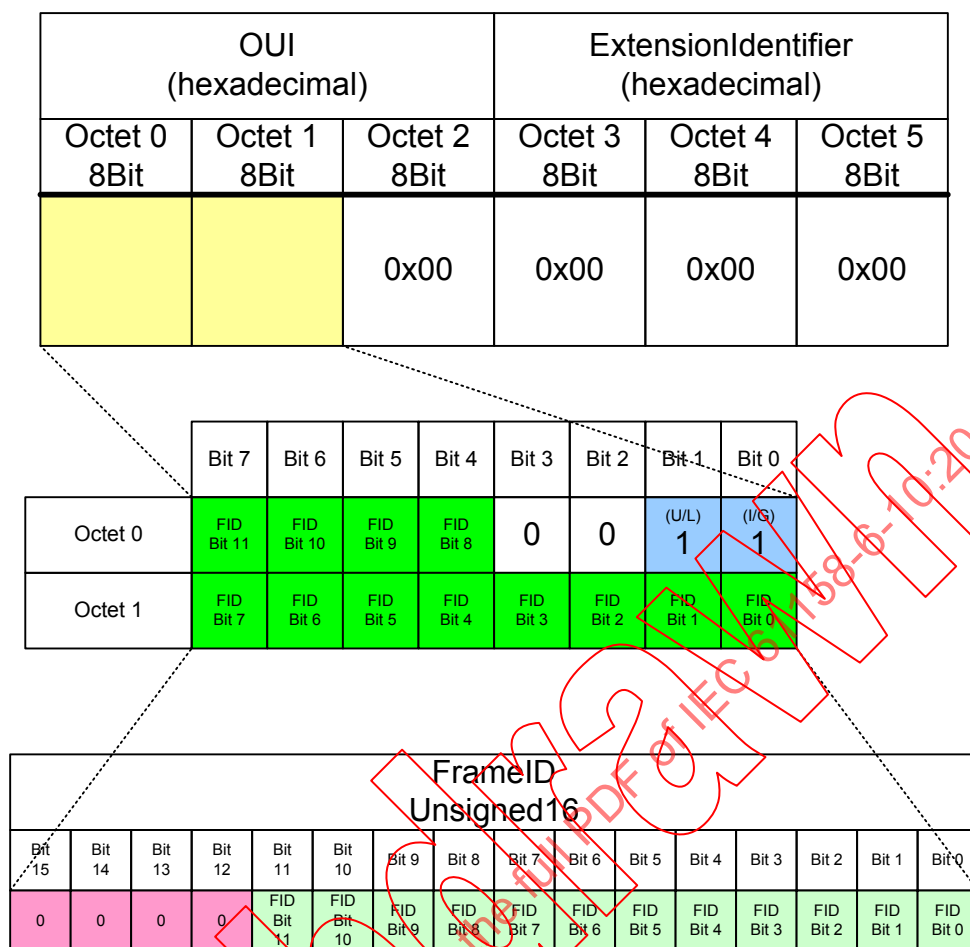
Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-A0	00-04-A0 – FF-FF-FF	Réservé pour d'autres applications

Tableau 19 – PTCP_MulticastMACAdd Page 8

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-80-C2	00-00-0E	Conjointement avec PTCP-DelayReqPDU et FrameID(=0xFF40), PTCP-DelayResPDU avec suivi et FrameID(=0xFF41), PTCP-DelayFuResPDU et FrameID(=0xFF42) et PTCP-DelayResPDU sans suivi et FrameID (=0xFF43) utilisés pour la mesure du délai

4.2.2.3.4 RTC-PDU avec FastForwarding

Pour les RTC-PDU avec FastForwarding, ce champ doit être codé conformément à la Figure 13 comme une "Adresse de groupe administrée localement" conformément aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.



where

(U/L) means „Universally or Locally administered address“

(I/G) means „Individual/Group address“

FID means FrameID

Légende

Anglais	Français
OUI (hexadecimal)	OUI (valeur hexadécimale)
ExtensionIdentifier (hexadecimal)	ExtensionIdentifier (valeur hexadécimale)
Where	Où
(U/L) means "Universally or Locally administered address"	(U/L) signifie "Adresse administrée de façon universelle ou locale"
(I/G) means "Individual/Group address"	(I/G) signifie "Adresse Individuelle/de Groupe"
FID means FrameID	FID signifie FrameID (ID de trame)

Figure 13 – FastForwardingMulticastMACAdd

Le champ DA est utilisé pour la décision de transfert et le champ FrameID est utilisé pour la décision CPM.

4.2.2.3.5 RTC-PDU avec adresse de multidiffusion de destination RT_CLASS_3

Pour les RTC-PDU avec adresse de multidiffusion de destination RT_CLASS_3, ce champ doit être codé conformément au Tableau 20.

Tableau 20 – Adresse de multidiffusion de destination RT_CLASS_3

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifieur (hexadécimale)	Signification
01-0E-CF	00-01-01	Adresse multidiffusion de destination RT_CLASS_3

4.2.2.3.6 RTC-PDU avec adresse de multidiffusion de trame non valide RT_CLASS_3

Pour les RTC-PDU avec adresse de multidiffusion de trame non valide RT_CLASS_3, ce champ doit être codé conformément au Tableau 21.

Tableau 21 – Adresse de multidiffusion de trame non valide RT_CLASS_3

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifieur (hexadécimale)	Signification
01-0E-CF	00-01-02	Adresse multidiffusion de trame non valide RT_CLASS_3

4.2.2.4 Codage du champ LT

Ce champ doit être codé en type de données Unsigned16 avec des valeurs conformes à l'IEEE 802.3. La présente spécification utilise les valeurs conformes au Tableau 22.

Tableau 22 – LT (Longueur/Type)

Valeur (hexadécimale)	Signification
0x0800	IP (UDP, RPC, SNMP, ICMP)
0x0806	ARP
0x8100	Tag Control Information
0x8892	RTC, RTA, DCP, PTCP
0x8817	IEEE 802.1AS
0x88CC	LLDP

4.2.2.5 Codage du champ TagControlInformation

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 11: TagControlInformation.VLAN_Id

Ce champ doit être codé conformément à l'IEEE 802.1Q.

NOTE La valeur VLAN_Id à zéro signifie qu'aucun VLAN n'est utilisé.

Bit 12: TagControlInformation.CanonicalFormatIdentificator

Ce champ doit être codé conformément à l'IEEE 802.1Q.

NOTE CFI est une constante 0.

Bit 13 – 15: TagControlInformation.Priority

Ce champ doit être codé conformément à l'IEEE 802.1Q.

Ce champ doit être codé avec les valeurs conformes au Tableau 23.

Tableau 23 – TagControllInformation.Priority

Valeur (hexadécimale)	Signification
0x00	IP, DCP
0x01	Réservé
0x02	Réservé
0x03	Réservé
0x04	Réservé
0x05	Bas avant RTA_CLASS_1 ou RTA_CLASS_UDP
0x06	RT_CLASS_UDP, RT_CLASS_1, RT_CLASS_2, RT_CLASS_3 ^a élevé avant RTA_CLASS_1 or RTA_CLASS_UDP
0x07	PTCP-AnnouncePDU
^a Les trames RT_CLASS_3 utilisent TagControllInformation.Priority uniquement en dehors de la période ROUGE.	

NOTE Si la mise en œuvre IP ne donne pas Tag Control Information, les trames RTA_CLASS_UDP et RT_CLASS_UDP peuvent être sans Tag Control Information.

4.2.2.6 Codage du champ FrameID

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 24, au Tableau 25, au Tableau 26, au Tableau 27, au Tableau 28, au Tableau 29, au Tableau 30, au Tableau 31, au Tableau 32, au Tableau 33, au Tableau 34, au Tableau 35 et au Tableau 36. Ce champ identifie la structure et le type de l'APDU.

Tableau 24 – FrameID– Plage 1

Valeur (hexadécimale)	Signification	Utilisation
0x0000 – 0x001F	Réservé	Réservé
0x0020	PTCP-RTSyncPDU (avec suivi)	Synchronisation PTCP avec suivi Horloge de travail pour la synchronisation d'horloge d'envoi et de phase
0x0021 – 0x007F	Réservé	Réservé

Tableau 25 – FrameID– Plage 2

Valeur (hexadécimale)	Signification	Utilisation
0x0080	PTCP-RTSyncPDU	Synchronisation PTCP sans suivi Horloge de travail pour la synchronisation d'horloge d'envoi et de phase
0x0081 – 0x009F	Réservé	—
0x00A0 – 0x00FF	Réservé	—

Tableau 26 – FrameID– Page 3

Valeur (hexadécimale)	Signification	Utilisation
0x0100 – 0x06FF	Dédié à la diffusion unique et à la multidiffusion de la classe de communication RT_CLASS_3 (ROUGE)	ROUGE, non redondant, normal ou DFP ^a
0x0700 – 0x0FFF	Dédié à la diffusion unique et à la multidiffusion de la classe de communication RT_CLASS_3 (ROUGE)	ROUGE, redondant ^b , normal ou DFP
0x1000 – 0x47FF	Réservé	—
^a Un surveillant de réseau peut classer les trames "normales" et "DFP" à l'aide du premier SFCRC16 de la trame. ^b Dans ce cas, deux FrameIDs (par exemple, 0xFFFF0 et 0xFFFF1) sont utilisés pour une CR. Un pour chaque sens dans lequel la trame traverse l'anneau. La paire doit être identifiée sans tenir compte du bit de poids faible de la FrameID.		

Tableau 27 – FrameID page 4

Valeur (hexadécimale)	Signification	Utilisation
0x4800 – 0x4FFF	Réservé	—
0x5000 – 0x57FF	Réservé	—
0x5800 – 0x5FFF	Réservé	—
0x6000 – 0x67FF	Réservé	—

Tableau 28 – FrameID page 5

Valeur (hexadécimale)	Signification	Utilisation
0x6800 – 0x6FFF	Réserve	—
0x7000 – 0x77FF	Réservé	—
0x7800 – 0x7FFF	Réservé	—

Tableau 29 – FrameID page 6

Valeur (hexadécimale)	Signification	Utilisation
0x8000 – 0xBBFF	Dédiée à la diffusion unique de la classe de communication RT_CLASS_1 (VERTE)	VERTE, non redondant, normal ^{a b}
0xBC00 – 0xBFFF	Dédiée à la multidiffusion de la classe de communication RT_CLASS_1 (VERTE)	VERTE, non redondant, normal ^{a b}
^a Utilisé à la place de l'ancienne classe de communication RT_CLASS_1 (héritée) présentée au Tableau 30 Tableau 30. ^b Peut être utilisé en tant que RT_CLASS_2 (ORANGE) seulement en conjonction avec ARProperties.StartupMode == Legacy.		

Tableau 30 – FrameID page 7

Valeur (hexadécimale)	Signification	Utilisation
0xC000 – 0xF7FF	Dédiée à la diffusion unique de la classe de communication RT_CLASS_UDP	RT_CLASS_UDP (recommandée) et RT_CLASS_1 (originelle) ^a ; L'utilisation concomitante de cette plage pour RT_CLASS_UDP et RT_CLASS_1 doit être rejetée.
0xF800 – 0xFBFF	Dédiée à la multidiffusion de la classe de communication RT_CLASS_UDP	RT_CLASS_UDP (recommandée) et RT_CLASS_1 (originelle) ^a ; L'utilisation concomitante de cette plage pour RT_CLASS_UDP et RT_CLASS_1 doit être rejetée.
^a RT_CLASS_1 (originelle) doit être prise en charge par appareil de commande E/S et superviseur d'E/S. Il convient qu'elle ne soit pas générée par un appareil E/S.		

Tableau 31 – FrameID page 8

Valeur (hexadécimale)	Signification	Utilisation
0xFC00	Réservé	—
0xFC01	Alarme haute	RTA_CLASS_1 et RTA_CLASS_UDP
0xFC02 – 0xFDFF	Réservé	—
0xFE00	Réservé	—
0xFE01	Alarme basse	RTA_CLASS_1 et RTA_CLASS_UDP
0xFE02 – 0xFEFB	Réservé	—
0xFEFC	DCP-Hello-ReqPDU	DCP
0xFEFD	DCP-Get-ReqPDU, DCP-Get-ResPDU, DCP-Set-ReqPDU, DCP-Set-ResPDU	DCP
0xFEFE	DCP-Identify-ReqPDU	DCP
0xFEFF	DCP-Identify-ResPDU	DCP

Tableau 32 – FrameID page 9

Valeur (hexadécimale)	Signification	Utilisation
0xFF00	PTCP-AnnouncePDU (horloge de travail)	Protocole PTCP Application isochrone, de synchronisation d'horloge et de phase d'envoi
0xFF01 – 0xFF1F	Réservé	—
0xFF20	PTCP-FollowUpPDU (horloge de travail)	Synchronisation d'horloge et de phase d'envoi (PTCP-FollowUpPDU)
0xFF21 – 0xFF3F	Réservé	—
0xFF40	PTCP-DelayReqPDU	Pour la mesure du délai
0xFF41	PTCP-DelayResPDU	Pour la mesure du délai avec suivi
0xFF42	PTCP-DelayFuResPDU	Pour la mesure du délai avec suivi
0xFF43	PTCP-DelayResPDU	Pour la mesure du délai sans suivi
0xFF44 – 0xFF5F	Réservé	—

Tableau 33 – FrameID page 10

Valeur (hexadécimale)	Signification	Utilisation
0xFF60 – 0xFF6F	Réservé	—

Tableau 34 – FrameID page 11

Valeur (hexadécimale)	Signification	Utilisation
0xFF70 – 0xFF7F	Réservé	—

Tableau 35 – FrameID page 12

Valeur (hexadécimale)	Signification	Utilisation
0xFF80 – 0xFF8F	FragmentationFrameID Voir 4.2.2.7	Protocole entre homologues pour activer des SendClockFactors inférieurs à 5 sans tenir compte de la taille des trames des protocoles utilisés de manière concomitante.

NOTE Le principe de fragmentation est utilisable avec tous les SendClockFactors.

Tableau 36 – FrameID page 13

Valeur (hexadécimale)	Signification	Utilisation
0xFF90 – 0xFFFF	Réservé	—

4.2.2.7 Codage du champ FragmentationFrameID

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 3: FragmentationFrameID.FragSequence

Ce champ doit être utilisé pour identifier la trame fragmentée par le destinataire et doit être encodé conformément au Tableau 37.

Tableau 37 – FragmentationFrameID.FragSequence

Valeur (hexadécimale)	Signification
0x00 – 0x0F	Nombre de trames incrémenté par l'émetteur pour chacune des trames fragmentées.

Bit 4 – 15: FragmentationFrameID.Constant

Ce champ doit être codé avec les valeurs conformes au Tableau 38.

Tableau 38 – FragmentationFrameID.Constant

Valeur (hexadécimale)	Signification
0xFF8	Partie supérieure de FrameID conformément au Tableau 35

4.3 Reconnaissance et configuration de base

4.3.1 Description de la syntaxe DCP

4.3.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.3.1.2 Syntaxe abstraite DCP APDU

Le Tableau 39 définit la syntaxe abstraite des DCP-PDU appelés APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU.

Tableau 39 – Syntaxe DCP APDU

Nom de l'APDU	Structure de l'APDU
DCP-PDU	DCP-Multicast-PDU ^g ^ DCP-Unicast-PDU
DCP-Multicast-PDU	DCP-Identify-ReqPDU ^g ^ DCP-Hello-ReqPDU
DCP-Unicast-PDU	DCP-Get-ReqPDU ^ DCP-Set-ReqPDU ^ DCP-Get-ResPDU ^ DCP-Set-ResPDU ^ DCP-Identify-ResPDU
DCP-Identify-ReqPDU	DCP-IdentifyFilter-ReqPDU ^ DCP-IdentifyAll-ReqPDU
DCP-IdentifyFilter-ReqPDU	DCP-MC-Header, [NameOfStationBlock] ^ [AliasNameBlock], IdentifyReqBlock* ^a
DCP-IdentifyAll-ReqPDU	DCP-MC-Header, AllSelectorBlock
DCP-Hello-ReqPDU	DCP-UC-Header, NameOfStationBlockRes, { IPParameterBlockRes, DeviceIDBlockRes, [DeviceOptionsBlockRes] ^e , DeviceRoleBlockRes, DeviceInitiativeBlockRes, [DeviceInstanceBlockRes] } ^f
DCP-Identify-ResPDU	DCP-UC-Header, { IdentifyResBlock* ^b , NameOfStationBlockRes ^c , IPParameterBlockRes ^c , DeviceIDBlockRes ^c , DeviceVendorBlockRes ^c , DeviceOptionsBlockRes ^c , DeviceRoleBlockRes ^c , [DeviceInitiativeBlockRes] ^d , [DeviceInstanceBlockRes] } ^f
DCP-Get-ReqPDU	DCP-UC-Header, GetReqBlock*
DCP-Get-ResPDU	DCP-UC-Header, (GetResBlock ^ GetNegResBlock)*
DCP-Set-ReqPDU	DCP-UC-Header, [StartTransactionBlock, BlockQualifier], [FactoryResetBlock, BlockQualifier] ^ [ResetToFactoryBlock, BlockQualifier] ^ SetReqBlock*, [StopTransactionBlock, BlockQualifier]
DCP-Set-ResPDU	DCP-UC-Header, (SetResBlock ^ SetNegResBlock)*
^a Le contenu de la valeur de champ d'IdentifyReqBlock du PDU doit être interprété comme un filtre au niveau de l'instance destinataire. Tous les DataBlocks inclus doivent être utilisés avec un opérateur logique AND et doivent uniquement obtenir une réponse DCP-Identify-ResPDU si tous les critères de recherche correspondent. ^b Le contenu de la valeur de champ d'IdentifyResBlock du PDU doit contenir l'option et la sous-option demandée dans IdentifyReqBlock. IdentifyResBlock ne doit contenir que l'élément AliasNameBlock utilisé dans IdentifyReqBlock. ^c Ce champ doit uniquement être présent s'il ne fait pas déjà partie d'IdentifyResBlock. ^d Ce champ doit uniquement être présent si l'utilisation de DCP-Hello-ReqPDU est activée. ^e Il convient que ce champ soit ignoré. ^f Des blocs supplémentaires peuvent être accolés par l'émetteur. ^g Peut être utilisé en conjonction avec une adresse MAC d'interface.	

Le Tableau 40 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 39.

Tableau 40 – Substitutions DCP

Nom de substitution	Structure
DCP-MC-Header	ServiceID, ServiceType, Xid, ResponseDelayFactor, DCPDataLength
DCP-UC-Header	ServiceID, ServiceType, Xid, Padding* ^a , DCPDataLength ^a Le nombre d'octets de bourrage doit être 2.
IdentifyReqBlock	DeviceRoleBlock ^ DeviceVendorBlock ^ DeviceIDBlock ^ DeviceOptionsBlock ^ MACAddressBlock ^ IPPParameterBlock ^ DHCPParameterBlock ^ ManufacturerSpecificParameterBlock
IdentifyResBlock	NameOfStationBlockRes ^ DeviceRoleBlockRes ^ DeviceVendorBlockRes ^ DeviceIDBlockRes ^ DeviceOptionsBlockRes ^ MACAddressBlockRes ^ IPPParameterBlockRes ^ DHCPParameterBlockRes ^ ManufacturerSpecificParameterBlockRes ^ AliasNameBlockRes
GetReqBlock	NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ MACAddressType ^ IPPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ DeviceInstanceType ^ AllSelectorType
GetResBlock	NameOfStationBlockRes ^ DeviceRoleBlockRes ^ DeviceVendorBlockRes ^ DeviceIDBlockRes ^ DeviceOptionsBlockRes ^ MACAddressBlockRes ^ IPPParameterBlockRes ^ FullIPSuiteBlockRes ^ DHCPParameterBlockRes ^ ManufacturerSpecificParameterBlockRes ^ DeviceInstanceBlockRes
GetNegResBlock	ControlOption, SuboptionResponse, DCPBlockLength, NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ MACAddressType ^ IPPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ DeviceInstanceType, BlockError, [AddDataValue]
SetReqBlock	SignalType ^ NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ IPPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType, DCPBlockLength, BlockQualifier, SignalValue ^ NameOfStationValue ^ DeviceRoleValue ^ DeviceVendorValue ^ DeviceIDValue ^ DeviceOptionsValue ^ IPPParameterValue ^ FullIPSuiteValue ^ DHCPParameterValue ^ ManufacturerSpecificParameterValue Seuls les blocs supplémentaires doivent être combinés à SetReqBlock. Exemple SignalType, DCPBlockLength, BlockQualifier, SignalValue
SetResBlock	ControlOption, SuboptionResponse, DCPBlockLength, NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ IPPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ SignalType ^ FactoryResetType ^ ResetToFactoryType ^ StartTransactionType ^ StopTransactionType, BlockError(=0), [AddDataValue]
SetNegResBlock	ControlOption, SuboptionResponse, DCPBlockLength, NameOfStationType ^ DeviceRoleType ^ DeviceVendorType ^ DeviceIDType ^ DeviceOptionsType ^ IPPParameterType ^ FullIPSuiteType ^ DHCPParameterType ^ ManufacturerSpecificParameterType ^ SignalType ^ FactoryResetType ^ ResetToFactoryType ^ StartTransactionType ^ StopTransactionType, BlockError(<=0), [AddDataValue]
AllSelectorType	AllSelectorOption, SuboptionAllSelector
AllSelectorBlock	AllSelectorType, DCPBlockLength

Nom de substitution	Structure
DeviceInitiativeType	DeviceInitiativeOption, DeviceInitiativeSuboption Utilisé seulement pour DCP-Hello-ReqPDU et pour DCP-Identify-ResPDU
DeviceInitiativeBlockRes	DeviceInitiativeType, DCPBlockLength, BlockInfo, DeviceInitiativeValue
StartTransactionType	ControlOption, SuboptionStart
StartTransactionBlock	StartTransactionType, DCPBlockLength
StopTransactionType	ControlOption, SuboptionStop
StopTransactionBlock	StopTransactionType, DCPBlockLength
SignalType	ControlOption, SuboptionSignal
FactoryResetType	ControlOption, SuboptionFactoryReset
FactoryResetBlock	FactoryResetType, DCPBlockLength
ResetToFactoryType	ControlOption, SuboptionResetToFactory
ResetToFactoryBlock	ResetToFactoryType, DCPBlockLength
NameOfStationType	DevicePropertiesOption, SuboptionNameOfStation
NameOfStationBlock	NameOfStationType, DCPBlockLength, NameOfStationValue
NameOfStationBlockRes	NameOfStationType, DCPBlockLength, BlockInfo, NameOfStationValue
AliasNameType	DevicePropertiesOption, SuboptionAliasName
AliasNameBlock	AliasNameType, DCPBlockLength, AliasNameValue
AliasNameBlockRes	AliasNameType, DCPBlockLength, BlockInfo, AliasNameValue
DeviceRoleType	DevicePropertiesOption, SuboptionDeviceRole
DeviceRoleBlock	DeviceRoleType, DCPBlockLength, DeviceRoleValue
DeviceRoleBlockRes	DeviceRoleType, DCPBlockLength, BlockInfo, DeviceRoleValue
DeviceRoleValue	DeviceRoleDetails, Padding
DeviceVendorType	DevicePropertiesOption, SuboptionDeviceVendor
DeviceVendorBlock	DeviceVendorType, DCPBlockLength, DeviceVendorValue
DeviceVendorBlockRes	DeviceVendorType, DCPBlockLength, BlockInfo, DeviceVendorValue
DeviceIDType	DevicePropertiesOption, SuboptionDeviceID
DeviceIDBlock	DeviceIDType, DCPBlockLength, DeviceIDValue
DeviceIDBlockRes	DeviceIDType, DCPBlockLength, BlockInfo, DeviceIDValue
DeviceIDValue	VendorIDHigh, VendorIDLow, DeviceIDHigh, DeviceIDLow
DeviceOptionsType	DevicePropertiesOption, SuboptionDeviceOptions
DeviceOptionsBlock	DeviceOptionsType, DCPBlocklength, DeviceOptionsValue
DeviceOptionsBlockRes	DeviceOptionsType, DCPBlocklength, BlockInfo, DeviceOptionsValue
DeviceOptionsValue	(Option ^a , Sous-option ^b)* ^c ^a Voir les valeurs au Tableau 49. ^b Voir Tableau 50, Tableau 51, Tableau 52, Tableau 53, Tableau 54, Tableau 55, Tableau 56, et Tableau 57 pour les valeurs. ^c Si aucune entrée n'existe, ignorer alors DeviceOptionsBlockRes dans la DCP-Identify-Res-PDU
DeviceInstanceType	DevicePropertiesOption, SuboptionDeviceInstance Non utilisable conjointement avec DCP-Set-ReqPDU
DeviceInstanceBlockRes	DeviceInstanceType, DCPBlockLength, BlockInfo, DeviceInstanceValue

Nom de substitution	Structure
DeviceInstanceValue	InstanceHigh, InstanceLow
MACAddressType	IPOption, SuboptionMACAddress
MACAddressBlock	MACAddressType, DCPBlockLength, MACAddressValue
MACAddressBlockRes	MACAddressType, DCPBlockLength, BlockInfo, MACAddressValue
IPParameterType	IPOption, SuboptionIPParameter
IPParameterBlock	IPParameterType, DCPBlockLength, IPParameterValue
IPParameterBlockRes	IPParameterType, DCPBlockLength, BlockInfo, IPParameterValue
IPParameterValue	IPAddress, Subnetmask, StandardGateway
FullIPSuiteType	IPOption, SuboptionFullIPSuite
FullIPSuiteBlockRes	FullIPSuiteType, DCPBlockLength, BlockInfo, FullIPSuiteValue
FullIPSuiteValue	IPAddress, Subnetmask, StandardGateway, IPAddress[4] ^a ^a Cette matrice contient jusqu'à quatre adresses IP de serveur DNS. La valeur "0.0.0.0" indique "aucune adresse".
DHCPPParameterType	DHCPOption, SuboptionDHCP
DHCPPParameterBlock	DHCPPParameterType, DCPBlockLength, DHCPPParameterValue
DHCPPParameterBlockRes	DHCPPParameterType, DCPBlockLength, BlockInfo, DHCPPParameterValue
DHCPPParameterValue	SuboptionDHCP, DHCPPParameterLength, DHCPPParameterData
ManufacturerSpecificParameterType	ManufacturerSpecificOption, SuboptionManufacturerSpecific
ManufacturerSpecificParameterBlock	ManufacturerSpecificParameterType, DCPBlockLength, ManufacturerSpecificParameterValue
ManufacturerSpecificParameterBlockRes	ManufacturerSpecificParameterType, DCPBlockLength, BlockInfo, ManufacturerSpecificParameterValue
ManufacturerSpecificParameterValue	ManufacturerOUI, ManufacturerSpecificString

4.3.1.3 Section de codage liée aux champs d'en-tête

4.3.1.3.1 Codage du champ ServiceID

Ce champ doit être codé en type de données Unsigned8 et doit contenir les valeurs indiquées au Tableau 41. Elles doivent être définies dans le PDU approprié.

Tableau 41 – ServiceID

Valeur (hexadécimale)	Signification
0x00 – 0x02	Réservé
0x03	Récupère
0x04	Met en place
0x05	Identification
0x06	Hello
0x07 – 0xFF	Réservé

4.3.1.3.2 Codage du champ **ServiceType**

4.3.1.3.2.1 Codage du champ **ServiceType** pour la demande

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0: ServiceType.Selection

Ce champ doit être codé avec les valeurs conformes au Tableau 42.

Tableau 42 – ServiceType.Selection

Valeur (hexadécimale)	Signification
0x00	Demande
0x01	Réservé

Bit 1 – 7: ServiceType.Reserved

Ce champ doit être codé avec les valeurs conformes au Tableau 43.

Tableau 43 – ServiceType.Reserved

Valeur (hexadécimale)	Signification
0x00	Par défaut
0x01 – 0x7F	Réserve

4.3.1.3.2.2 Codage du champ **ServiceType** pour la réponse

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0: ServiceType.Selection

Ce champ doit être codé avec les valeurs conformes au Tableau 44.

Tableau 44 – ServiceType.Selection

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Réponse

Bit 1: ServiceType.Reserved_1

Ce champ doit être codé avec les valeurs conformes au Tableau 45.

Tableau 45 – ServiceType.Reserved_1

Valeur (hexadécimale)	Signification
0x00	Par défaut
0x01	Réservé

Bit 2: ServiceType.Response

Ce champ doit être codé avec les valeurs conformes au Tableau 46.

Tableau 46 – ServiceType.Response

Valeur (hexadécimale)	Signification
0x00	Succès Par défaut
0x01	ServiceID non pris en charge Facultatif sinon, la demande peut être ignorée.

Bit 3 – 7: ServiceType.Reserved_2

Ce champ doit être codé avec les valeurs conformes au Tableau 47.

Tableau 47 – ServiceType.Reserved_2

Valeur (hexadécimale)	Signification
0x00	Par défaut
0x01 – 0x1F	Réservé

4.3.1.3.3 Codage du champ Xid

Ce champ doit être codé en type de données Unsigned32. Il doit contenir une identification de transaction choisie par le client afin d'associer les demandes et les réponses entre un client et un serveur.

4.3.1.3.4 Codage du champ DCPDataLength

Ce champ doit être codé en type de données Unsigned16. Il doit contenir la longueur totale des données suivie de DCP-UC-Header ou de DCP-MC-Header, en octets. La longueur maximale des données DCP est de 1 432 octets.

4.3.1.3.5 Codage du champ ResponseDelayFactor

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 48. Il doit contenir un facteur de délai qui doit être utilisé par le serveur pour calculer son délai de réponse individuel.

Tableau 48 – ResponseDelayFactor

Valeur (hexadécimale)	Signification
0x0000	Réservé
0x0001	Valeur admise sans propagation
0x0002 – 0x1900	Valeur admise avec propagation
0x1901 – 0xFFFF	Réservé

Le serveur doit calculer le délai de réponse (valeurs admises entre 0 ms et 64 s) conformément à la formule suivante.

Les deux derniers octets de l'adresse MAC (voir les adresses LAN MAC universelles en 48 bits de l'IEEE 802) doivent être utilisés comme nombres aléatoires K. Dans l'adresse MAC, l'octet 6 représente l'octet de poids faible et l'octet 5 représente l'octet de poids fort.

$$\text{Propagation} = K \text{ MOD } \text{ResponseDelayFactor} \quad (1)$$

où

Propagation	est le facteur de propagation spécifique
K	est le nombre aléatoire, les deux derniers octets de l'adresse MAC en tant que nombre aléatoire
ResponseDelayFactor	est le facteur pour le calcul d'un délai de réponse individuel

Propagation différente de zéro:

$$\text{Délai Minimal de Réponse} = 10 \text{ ms} \times \text{Propagation} \quad (2)$$

où

Délai Minimal de Réponse	est le délai de réponse minimal
Propagation	est le facteur de propagation spécifique

Propagation égale à zéro:

$$\text{Délai Minimal de Réponse} = 0 \text{ ms} \quad (3)$$

où

Délai Minimal de Réponse	est le délai de réponse minimal

Le client doit calculer le délai de réponse conformément à la formule suivante.

ResponseDelayFactor égal à un:

$$\text{Délai de réponse} = 400 \text{ ms} \quad (4)$$

où

Délai de réponse	est le délai de réponse

ResponseDelayFactor supérieur à un:

$$\text{Valeur intermédiaire} = 1 \text{ s} + \text{ResponseDelayFactor} \times 10 \text{ ms} \quad (5)$$

où

Valeur intermédiaire	est la valeur intermédiaire du délai de réponse
ResponseDelayFactor	est le facteur pour le calcul d'un délai de réponse individuel

$$\text{Délai de réponse} = \text{Arrondi à la seconde supérieure (Valeur intermédiaire)} \quad (6)$$

où

Délai de réponse	est le délai de réponse
Valeur intermédiaire	est la valeur intermédiaire du délai de réponse

4.3.1.4 Section de codage des champs de bloc

4.3.1.4.1 Généralités

Les champs de bloc sont répartis en option, sous-option, longueur de bloc, informations sur le bloc et valeur. Chaque bloc doit assurer un alignement Unsigned16. Les octets de bourrage ajoutés dont la valeur est zéro doivent être comptés pour DCPDataLength et ne doivent pas être comptés pour DCPBlockLength. Le Tableau 49 donne la liste des options disponibles et le Tableau 50, Tableau 51, le Tableau 52, le Tableau 53, Tableau 54, le Tableau 55 et le Tableau 56 donnent la liste des sous-options disponibles.

Tableau 49 – Liste des options

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	IPOption
0x02	DevicePropertiesOption
0x03	DHCPOption
0x04	Réservé
0x05	ControlOption
0x06	DeviceInitiativeOption
0x07 – 0x7F	Réservé
0x80 – 0xFE	ManufacturerSpecificOption
0xFF	AllSelectorOption

Tableau 50 – Liste des sous-options d'IPOption

Valeur (hexadécimale)	Signification	Accessible
0x01	SuboptionMACAddress	Lecture
0x02	SuboptionIPPParameter	Lecture / Ecriture
0x03	SuboptionFullIPSuite	Lecture / Ecriture Facultatif
Autre	Réservé	—

Tableau 51 – Liste des sous-options de DevicePropertiesOption

Valeur (hexadécimale)	Signification	Accessible
0x01	SuboptionDeviceVendor	Lecture
0x02	SuboptionNameOfStation	Lecture / Ecriture
0x03	SuboptionDeviceID	Lecture
0x04	SuboptionDeviceRole	Lecture
0x05	SuboptionDeviceOptions	Lecture
0x06	SuboptionAliasName	Utilisé uniquement en tant que filtre
0x07	SuboptionDeviceInstance	Lecture Facultatif
Autre	Réservé	—

Tableau 52 – Liste des sous-options de DHCPOption

Valeur (hexadécimale)	Signification	Accessible
Voir Tableau 57	SuboptionDHCP	Lecture / Ecriture

Tableau 53 – Liste des sous-options de ControlOption

Valeur (hexadécimale)	Signification	Accessible
0x01	SuboptionStart	Ecriture
0x02	SuboptionStop	Ecriture
0x03	SuboptionSignal	Ecriture
0x04	SuboptionResponse	—
0x05	SuboptionFactoryReset	Ecriture
0x06	SuboptionResetToFactory	Ecriture
Autre	Réservé	—

Tableau 54 – Liste des sous-options de DeviceInitiativeOption

Valeur (hexadécimale)	Signification	Accessible
0x01	SuboptionDeviceInitiative	Lecture
Autre	Réservé	—

Tableau 55 – Liste des sous-options de AllSelectorOption

Valeur (hexadécimale)	Signification	Accessible
0xFF	SuboptionAllSelector	Utilisé uniquement en tant que filtre
Autre	Réservé	—

Tableau 56 – Liste des sous-options de ManufacturerSpecificOption

Valeur (hexadécimale)	Signification	Accessible
0x00 – 0xFF	SuboptionManufacturerSpecific	Spécifique à un fabricant

4.3.1.4.2 Section de codage relative à IPOption

4.3.1.4.2.1 Codage du champ IPOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.2.2 Codage du champ SuboptionMACAddress

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.2.3 Codage du champ SuboptionIPParameter

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.3 Section de codage relative à DevicePropertiesOption

4.3.1.4.3.1 Codage du champ DevicePropertiesOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.3.2 Codage du champ SuboptionDeviceVendor

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

4.3.1.4.3.3 Codage du champ SuboptionNameOfStation

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

4.3.1.4.3.4 Codage du champ SuboptionDeviceID

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

4.3.1.4.3.5 Codage du champ SuboptionDeviceRole

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

4.3.1.4.3.6 Codage du champ SuboptionDeviceOptions

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

Il convient que cette combinaison option/sous-option soit utilisée pour acheminer une liste des options et sous-options prises en charge qui sont définies comme facultatives et qui sont susceptibles d'être utilisées pour toutes les options et sous-options.

4.3.1.4.3.7 Codage du champ SuboptionAliasName

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

4.3.1.4.3.8 Codage du champ SuboptionDeviceInstance

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 51.

4.3.1.4.4 Section de codage relative à DHCPOption

4.3.1.4.4.1 Codage du champ DHCPOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.4.2 Codage du champ SuboptionDHCP

Ce champ doit être codé en type de données Unsigned8. Les valeurs admises doivent être conformes au Tableau 57.

Tableau 57 – SuboptionDHCP

Valeur (décimale)	Longueur de données	Description	Références
0 - 11	—	Réservé ^a	—
12	1 +	Nom d'hôte ^d	RFC 2132
13 - 42	—	Réservé ^a	—
43	1 +	Informations spécifiques au vendeur	RFC 2132
44 - 53	—	Réservé ^a	—

Valeur (décimale)	Longueur de données	Description	Références
54	4	Identifiant du serveur	RFC 2132
55	1 +	Liste de demandes de paramètre ^c	RFC 2132
56 - 59	—	Réservé ^a	—
60	1 +	Identifiant de classe	RFC 2132
61	2 +	Identifiant du client DHCP	RFC 2132, RFC 4361, and RFC 4363
62 - 80	—	Réservé ^a	—
81	1 +	FDQN, Nom de domaine qualifié complet ^d	—
82 - 96	—	Réservé ^a	—
97	Variable	Identifiant du client UUID (= Station adressée)	—
98 - 254	—	Réservé ^a	—
255 ^b	1	Voir 4.3.1.4.24	—
^a Il convient d'utiliser les options réservées dans le sens de leur définition des RFC correspondantes pour DHCP.. ^b Défini comme "END" (FIN) par RFC 2132, mais pris en charge de manière interne. Par conséquent, cette clé est surchargée comme commande de DHCP dans la présente norme. ^c Dans cette option, il convient de demander au moins les paramètres 1 (masque de sous-réseau) et 3 (routeur). ^d L'utilisation de ces sous-options n'est pas encore fixée et fait l'objet de modifications dans les versions ultérieures.			

4.3.1.4.5 Section de codage relative à ControlOption

4.3.1.4.5.1 Codage du champ ControlOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.5.2 Codage du champ SuboptionStart

Ce champ doit être codé en type de données Unsigned8 conformément au Tableau 53 et avec un DCPBlockLength conformément au Tableau 58.

Tableau 58 – Codage de DCPBlockLength conjointement avec SuboptionStart

Valeur (hexadécimale)	Signification
0x0000 – 0x0001	Réservé
0x0002	Par défaut
0x0003 – 0xFFFF	Réservé

Il n'existe aucune donnée supplémentaire pour cette sous-option.

4.3.1.4.5.3 Codage du champ SuboptionStop

Ce champ doit être codé en type de données Unsigned8 conformément au Tableau 53 et avec un DCPBlockLength conformément au Tableau 59.

Tableau 59 – Codage de DCPBlockLength conjointement avec SuboptionStop

Valeur (hexadécimale)	Signification
0x0000 – 0x0001	Réservé
0x0002	Par défaut
0x0003 – 0xFFFF	Réservé

Il n'existe aucune donnée supplémentaire pour cette sous-option.

4.3.1.4.5.4 Codage du champ SuboptionSignal

Ce champ doit être codé en type de données Unsigned8 conformément au Tableau 53 et avec un DCPBlockLength conformément au Tableau 60.

Tableau 60 – Codage de DCPBlockLength conjointement avec SuboptionSignal

Valeur (hexadécimale)	Signification
0x0000 – 0x0003	Réservé
0x0004	Par défaut
0x0005 – 0xFFFF	Réservé

4.3.1.4.5.5 Codage du champ SuboptionResponse

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 53.

4.3.1.4.5.6 Codage du champ SuboptionFactoryReset

Ce champ doit être codé en type de données Unsigned8 conformément au Tableau 53 et avec un DCPBlockLength conformément au Tableau 61.

Tableau 61 – Codage de DCPBlockLength conjointement avec SuboptionFactoryReset

Valeur (hexadécimale)	Signification
0x0000 – 0x0001	Réservé
0x0002	Par défaut
0x0003 – 0xFFFF	Réservé

Il n'existe aucune donnée supplémentaire pour cette sous-option.

Factory Reset doit être en permanence mis

- Le NameOfStation à ""
- L'Adresse IP, le masque de sous-réseau et la passerelle normalisée à 0.0.0.0
- Tous les autres paramètres à la valeur par défaut du fabricant.

NOTE Tous les paramètres stockés de l'appareil physique tels que PDIRData, PDPortDataAdjust, ... sont mis à la valeur par défaut des fabricants.

Il est recommandé que chaque appareil prenne en charge la possibilité d'exécuter la "réinitialisation à l'état d'usine" sans appareil d'ingénierie. Un commutateur ou un autre moyen pourrait permettre d'y arriver.

Il convient que l'exécution de Factory Reset n'influence pas la commutation d'un nœud.

4.3.1.4.5.7 Codage du champ SuboptionResetToFactory

Ce champ doit être codé en type de données Unsigned8 conformément au Tableau 53 et avec un DCPBlockLength conformément au Tableau 62.

Tableau 62 – Codage de DCPBlockLength conjointement avec SuboptionResetToFactory

Valeur (hexadécimale)	Signification
0x0000 – 0x0001	Réservé
0x0002	Par défaut
0x0003 – 0xFFFF	Réservé

Il n'existe aucune donnée supplémentaire pour cette sous-option.

Cette Sous-option offre différents modes codés par le BlockQualifier.

4.3.1.4.6 Section de codage relative à DeviceInitiativeOption

4.3.1.4.6.1 Codage du champ DeviceInitiativeOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.6.2 Codage du champ SuboptionDeviceInitiative

Ce champ doit être codé en type de données Unsigned8 conformément au Tableau 54 et avec un DCPBlockLength conformément au Tableau 63.

Tableau 63 – Codage de DCPBlockLength conjointement avec SuboptionDeviceInitiative

Valeur (hexadécimale)	Signification
0x0000 – 0x0003	Réservé
0x0004	Par défaut
0x0005 – 0xFFFF	Réservé

4.3.1.4.7 Section de codage relative à AllSelectorOption

4.3.1.4.7.1 Codage du champ AllSelectorOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.7.2 Codage du champ SuboptionAllSelector

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 55.

4.3.1.4.8 Section de codage relative à ManufacturerSpecificOption

4.3.1.4.8.1 Codage du champ ManufacturerSpecificOption

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 49.

4.3.1.4.8.2 Codage du champ SuboptionManufacturerSpecific

Ce champ doit être codé en type de données Unsigned8 conformément à Tableau 56.

4.3.1.4.9 Codage du champ DCPBlockLength

Ce champ doit être codé en type de données Unsigned16 et contenir la longueur des données spécifiques de la sous-option, sans DCPBlockLength et sans tenir compte des octets de bourrage.

4.3.1.4.10 Codage du champ BlockQualifier

Ce champ doit être codé en type de données Unsigned16.

Les valeurs admises avec les options IPOption, DevicePropertiesOption, DHCPOption et ManufacturerSpecificOption doivent être conformes au Tableau 64.

Tableau 64 – BlockQualifier avec les options IPOption, DevicePropertiesOption, DHCPOption et ManufacturerSpecificOption

Bit	Valeur (binaire)	Signification
Bit 0	0	Stocke la valeur de l'option/la sous-option et son attribut "temporary" dans l'ASE DCP et utilise la valeur ultérieurement. L'attribut "temporary" signifie que la valeur doit être réinitialisée à sa valeur par défaut après un cycle mise hors tension / mise sous tension.
	1	Stocke la valeur de l'option/la sous-option et son attribut "permanent" dans l'ASE DCP et utilise la valeur ultérieurement. L'attribut "permanent" signifie que la valeur doit être disponible après un cycle mise hors tension / mise sous tension.
Bit 1 – 15	—	Réservé

Les valeurs admises avec l'option ControlOption et la sous-option SuboptionResetToFactory doivent être conformes au Tableau 65.

Tableau 65 – BlockQualifier avec l'option ControlOption et la sous-option SuboptionResetToFactory

Bit	Valeur (décimale)	Signification	Plage
Bit 0	0	Réservé	—
	1		
Bit 1 – 15	0	Réservé	—
	1	Recommandé Réinitialisation des données d'application Réinitialisation des données stockées de façon permanente dans les sous-modules et les modules à leurs valeurs d'usine.	Interface
	2	Obligatoire Réinitialisation du paramètre de communication Tous les paramètres actifs pour l'interface ou les ports et les AR doivent être mis à leurs valeurs par défaut et s'ils sont stockés de façon permanente, réinitialisés. Ce mode est destiné à mettre l'interface de communication adressée d'un appareil dans un état quasiment similaire à l'état "out of the box".	

Bit	Valeur (décimale)	Signification	Plage
	3	Recommandé Réinitialisation des paramètres d'ingénierie Réinitialisation des paramètres d'ingénierie stockés de façon permanente dans l'IOC ou l'IOD à leurs valeurs d'usine.	
	4	Recommandé Réinitialisation de toutes les données stockées Réinitialisation de toutes les données stockées dans l'IOD ou l'IOC à leurs valeurs d'usine par défaut.	
	5 - 7	Réservé	
	8	Recommandé Réinitialisation de toutes les données stockées dans l'IOD ou l'IOC à leurs valeurs d'usine. Ce service doit réinitialiser les paramètres de communication de toutes les interfaces de l'appareil et il convient qu'il réinitialise tous les paramètres de l'appareil. Ce mode est destiné à mettre l'appareil dans un état similaire à l'état "out of the box".	Appareil
	9	Facultatif Réinitialisation et restauration des données Réinitialisation des révisions des logiciels installés à des images d'usine.	
	10 - 15	Réservé	
	autre	Réservé	—
Le comportement à adopter doit être défini par les profils et/ou l'application locale.			

Dans tous les autres cas, le champ BlockQualifier doit être conforme au Tableau 66.

Tableau 66 – BlockQualifier avec toutes les autres options et sous-options

Bit	Valeur (binaire)	Signification
Bit 0 – 15	0	Par défaut
Bit 0 – 15	autre	Réservé

4.3.1.4.11 Codage du champ BlockError

Ce champ doit être codé en type de données Unsigned8 et doit être mis conformément au Tableau 67.

Tableau 67 – BlockError

Valeur (hexadécimale)	Signification	Utilisation
0x00	Pas d'erreur	Réponse positive Paramètre
0x01	Option non prise en charge	Si facultatif, l'option n'est pas prise en charge
0x02	Sous-option non prise en charge ou pas de DataSet disponible ou BlockQualifier non pris en charge	Si facultatif, la sous-option n'est pas prise en charge
0x03	Sous-option non définie	Si la sous-option n'a pas pu être définie pour des raisons locales
0x04	Erreur de ressource	En cas d'erreur de ressource temporaire dans le serveur
0x05	SET impossible pour des raisons locales	Si le service Set est impossible pour des raisons locales
0x06	En fonctionnement, SET impossible	Si le service Set est impossible en raison du fonctionnement de l'application
0x07 – 0xFF	Réservé	Ne doivent pas être utilisées

4.3.1.4.12 Codage du champ BlockInfo

Ce champ doit être codé en type de données Unsigned16. Les valeurs admises doivent être définies conformément au Tableau 68, au Tableau 69, au Tableau 70 et au Tableau 71.

Tableau 68 – BlockInfo pour SuboptionIPParameter

Bit	Signification
Bit 0 – 1	Voir Tableau 69
Bit 2 – 6	Réservé
Bit 7	Voir Tableau 70
Bit 8 – 15	Réservé

Tableau 69 – Bit 1 et Bit 0 de BlockInfo pour SuboptionIPParameter

Bit 1	Bit 0	Signification pour SuboptionIPParameter
0	0	Pas de paramètre IP
0	1	Paramètre IP défini
1	0	Paramètre IP défini par le biais de DHCP
1	1	Réservé

Tableau 70 – Bit 7 de BlockInfo pour SuboptionIPParameter

Bit 7	Signification pour SuboptionIPParameter
0	Aucun conflit d'adresse IP détecté ^a Adresse IP active
1	Conflit d'adresse IF détecté ^a Adresse IP inactive
^a Un essai facultatif défini en local montre que l'adresse IP demandée est déjà utilisée par un autre appareil. L'appareil ne peut pas activer l'adresse IP délivrée.	

Tableau 71 – BlockInfo pour toutes les autres sous-options

Bit	Signification
Bit 0 – 15	Réservé

4.3.1.4.13 Codage du champ DeviceInitiativeValue

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0: DeviceInitiativeValue.Hello

Ce champ doit être mis conformément au Tableau 72.

Tableau 72 – DeviceInitiativeValue

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	L'appareil n'émet pas de DCP-Hello-ReqPDU après la mise sous tension.
0x01	ON	L'appareil émet un DCP-Hello-ReqPDU après la mise sous tension.

Bit 1 – 15: DeviceInitiativeValue.reserved

Ce champ doit être mis conformément à 3.4.2.2.

4.3.1.4.14 Codage du champ SignalValue

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 73.

Tableau 73 – SignalValue

Valeur (hexadécimale)	Signification	Utilisation
0x0100	Clignoter une fois	Clignoter une DEL (par exemple, une Ethernet LINK LED) ou un autre système de signalisation d'une durée de 3 s à une fréquence de 1 Hz (500 ms allumée, 500 ms éteinte).

4.3.1.4.15 Codage du champ NameOfStationValue**4.3.1.4.15.1 Généralités**

Chaque NameOfStation doit être unique, si un appareil est équipé de plusieurs sous-modules d'interface.

4.3.1.4.15.2 Codage

Ce champ doit être codé en type de données OctetString avec 1 à 240 octets. La définition de RFC 5890 et la syntaxe suivante s'appliquent:

- 1 ou plusieurs étiquettes, séparées par [.]
- La longueur totale est comprise entre 1 et 240
- La longueur de l'étiquette est comprise entre 1 et 63
- Les étiquettes sont composées de [a-z0-9-]
- Les étiquettes ne commencent pas par [-]
- Les étiquettes ne se terminent pas par [-]

- La première étiquette ne se présente pas sous la forme "port-xyz" ou "port-xyz-abcde" avec a, b, c, d, e, x, y, z = 0...9, pour éviter une mauvaise similitude avec le champ AliasNameValue
- Les noms de station ne se présentent pas sous la forme n.n.n.n, n = 0...999

NOTE 1 Les étiquettes commencent uniquement par 'xn-' si la chaîne originale contient des caractères autres que [a-z0-9-].

EXEMPLE 1 "device-1.machine-1.plant-1.vendor"

EXEMPLE 2 "mühle1.ölmühle1.plant.com" est codé "xn-mhle1-kva.xn-lmhle1-vxa4c.plant.com"

NOTE 2 Le champ NameOfStationValue ne se termine pas par zéro.

NOTE 3 Les appareils dotés de plus d'une interface Ethernet nécessitent plus d'un NameOfStation unique. Dans ce cas, le nom de station est un nom d'interface. Etant donné que la plupart des appareils ne nécessitent qu'une seule interface, le nom de l'interface est également le nom de station.

4.3.1.4.15.3 Sémantique avec le service Identify

Les règles suivantes doivent être appliquées:

- Nom de la station composé de 1 à 240 octets:
seul l'appareil portant exactement ce nom doit répondre. En cas de comparaison de deux noms, les conventions DNS doivent être prises en compte.
- Nom de station de longueur == 0:
seuls les appareils auxquels n'a pas encore été attribué un nom de station doivent répondre.

La représentation réseau de NameOfStation doit être conforme à 4.3.1.4.15.2.

4.3.1.4.15.4 Sémantique avec le service Set

Les règles suivantes doivent être appliquées:

- Nom de la station composé de 1 à 240 octets:
Le nom de la station doit être défini en fonction de NameOfStationValue.
- Nom de station de longueur == 0:
La station doit supprimer son nom enregistré.

La représentation réseau de NameOfStation doit être conforme à 4.3.1.4.15.2.

4.3.1.4.16 Codage du champ NameOfPort

Ce champ doit être codé en OctetString[8] ou en OctetString[14] comme "port-xyz" ou "port-xyz-rstuv" où x, y, z est dans la plage "0"- "9" de 001 à 255 et r, s, t, u, v est dans la plage "0"- "9" de 00000 à 65535. Les valeurs x, y, z doivent être utilisées pour identifier le numéro de port. Les valeurs r, s, t, u, v doivent être utilisées pour identifier l'emplacement du port.

Les valeurs "port-001-00000" doivent être utilisées pour le premier sous-module de port d'une interface au sein de l'emplacement 0 si tout autre emplacement peut contenir un autre sous-module de port.

Les valeurs "port-001" doivent être utilisées pour le premier sous-module de port d'une interface si aucun autre emplacement ne peut contenir un autre sous-module de port.

De plus, la définition selon la RFC 5890 doit être appliquée.

4.3.1.4.17 Codage du champ AliasNameValue

4.3.1.4.17.1 Codage

Ce champ doit être codé en OctetString. Le contenu doit être la concaténation du contenu des champs NameOfPort et NameOfStation.

$$\text{AliasNameValue} = \text{NameOfPort} + "." + \text{NameOfStation} \quad (7)$$

où

AliasNameValue	est un nom supplémentaire d'un nœud. Il fait référence à une interface particulière de ce nœud.
NameOfPort	est le nom du port du voisin.
NameOfStation	est le nom de la station du voisin.

4.3.1.4.17.2 Sémantique avec le service Identify

Les règles suivantes doivent être appliquées:

- Nom d'alias de longueur != 0:
seul l'appareil portant exactement ce nom d'alias supplémentaire doit répondre. En cas de comparaison de deux noms, les conventions DNS doivent être prises en compte. L'appareil ne doit pas répondre avec AliasName. Il doit plutôt utiliser le NameOfStation en cours (même si NameOfStation n'est pas défini).
- Nom de station de longueur == 0:
Non admis dans une requête.

NOTE AliasName est un autre nom unique d'un appareil ou d'une interface d'un appareil. Le nom d'alias est déduit des informations de topologie ou de voisinage. Le nom d'alias de la station ne peut pas être défini ni extrait au moyen de DCP.

4.3.1.4.18 Codage du champ DeviceRoleDetails

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 74.

Tableau 74 – DeviceRoleDetails

Bit	Signification	Utilisation
Bit 0	PNIO Device	L'appareil contient une interface d'appareil E/S.
Bit 1	PNIO Controller	L'appareil contient une interface de contrôleur E/S.
Bit 2	PNIO Multidevice	L'appareil contient plusieurs interfaces d'appareil E/S.
Bit 3	PNIO Supervisor	L'appareil contient une interface de superviseur E/S.
Bit 4 – 7	Réservé	—

4.3.1.4.19 Codage du champ MACAddressValue

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

4.3.1.4.20 Codage du champ IPAddress

Ce champ doit être codé en type de données OctetString[4]. Le codage des champs doit être conforme au RFC 791, au RFC 5735 et au Tableau 75.

La valeur zéro doit être utilisée dans tous les champs pour supprimer une adresse IP déjà définie.

Tableau 75 – IPAddress

Valeur (CIDR ^a)	Signification
0.0.0.0 / 8	Aucune adresse IP attribuée
127.0.0.0 / 8	Réservé Boucle avec retour
224.0.0.0 / 4	Réservé Multidiffusion
Autre	Adresse IP attribuée
Réservé	Si un sous-réseau est utilisé, son adresse IP supérieure fait office d'adresse de diffusion de sous-réseau. Il convient que l'attribution de cet IPAddress soit rejetée.
^a Notation conforme au routage interdomaine sans classe de RFC 4632	

NOTE L'objet IP suite (IPAddress, Subnetmask and StandardGateway) n'existe qu'une fois. Le fanion "store permanently" est un attribut de l'objet et changeait conjointement avec chaque accès en écriture.

4.3.1.4.21 Codage du champ Subnetmask

Ce champ doit être codé en type de données OctetString[4]. Le codage des champs doit être conforme au RFC 791, au RFC 5735 et au Tableau 76.

Tableau 76 – Subnetmask

Valeur	Nombre d'hôtes	Signification
255.255.255.255	1	Masque de sous-réseau attribué
255.255.255.254	2	
255.255.255.252	4	
255.255.255.248	8	
255.255.255.240	16	
255.255.255.224	32	
255.255.255.192	64	
255.255.255.128	128	
255.255.255.000	256	
255.255.254.000	512	
255.255.252.000	1 024	
255.255.248.000	2 048	
255.255.240.000	4 096	
255.255.224.000	8 192	
255.255.192.000	16 384	
255.255.128.000	32 768	
255.255.000.000	65 536	
255.254.000.000	131 072	
255.252.000.000	262 144	
255.248.000.000	524 288	
255.240.000.000	1 048 576	

Valeur	Nombre d'hôtes	Signification
255.224.000.000	2 097 152	
255.192.000.000	4 194 304	
255.128.000.000	8 388 608	
255.000.000.000	16 777 216	
254.000.000.000	33 554 432	Masque de sous-réseau valide, prise en charge facultative conformément aux règles de l'IANA
252.000.000.000	67 108 864	
248.000.000.000	134 217 728	
240.000.000.000	268 435 456	
224.000.000.000	536 870 912	Masque de sous-réseau valide, mais rejeté conformément aux règles de l'IANA
192.000.000.000	1 073 741 824	
128.000.000.000	2 147 483 648	
000.000.000.000	4 294 967 296	Masque de sous-réseau valide, mais rejeté conformément aux règles de l'IANA si un StandardGateway est attribué. Masque de sous-réseau valide et attribué si aucun StandardGateway n'est attribué.
Autre	Réservé	Masque de sous-réseau non valide

4.3.1.4.22 Codage du champ StandardGateway

Ce champ doit être codé en type de données OctetString[4]. Le codage de ce champ doit être conforme au RFC 791, au RFC 5735, au Tableau 75 et au Tableau 77.

Si aucun IPAddress n'est attribué, les valeurs pour IPAddress, Subnetmask et StandardGateway doivent être nulles.

Tableau 77 – StandardGateway

Valeur (CIDR ^a)	Signification
0.0.0.0 / 32	Aucune passerelle normalisée attribuée L'application doit configurer l'implémentation IP locale afin d'éviter le routage IP
Own IPAddress / 32	Aucune passerelle normalisée attribuée L'application doit configurer l'implémentation IP locale afin d'éviter le routage IP
Autre ^b	Passerelle normalisée attribuée Il convient de vérifier de nouveau le masque de sous-réseau du champ
Réservé ^c	Aucune passerelle normalisée attribuée
^a Notation conforme au routage interdomaine sans classe de RFC 1518 et de RFC 1519	
^b Valide conformément au Tableau 75 et au Tableau 76.	
^c Non valide conformément au Tableau 75 et au Tableau 76.	

Les règles de vérification de la StandardGateway dépendent de l'existence de plus d'une interface. Pour plus d'interfaces, seule une StandardGateway (un seul objet) doit exister. Cela signifie que chaque manipulation de la StandardGateway à partir de n'importe quelle interface a un impact sur toutes les autres interfaces. Par conséquent, dans ce cas, la vérification doit être effectuée par des méthodes et des règles locales.

4.3.1.4.23 Codage des champs DHCPParameterValue à l'aide de l'Option 61 de DHCP

Le client DHCP peut utiliser l'identifiant du client pour demander ses paramètres IP à un serveur DHCP. Un appareil doit pouvoir utiliser l'identifiant du client provenant de l'ensemble de données de la manière suivante:

4.3.1.4.23.1 Codage du champ DHCPParameterLength

Ce champ doit être codé en type de données Unsigned16 et contenir la longueur des données consécutives spécifiques.

4.3.1.4.23.2 Utilisation de MACAddress en tant qu'identifiant de client

Si l'appareil doit obtenir ses paramètres IP à l'aide du protocole DHCP et utiliser l'adresse MAC comme identifiant de client, l'option 61 doit prendre les valeurs suivantes:

SuboptionDHCP = 61

DHCPParameterLength = 1

DHCPParameterData = 1

4.3.1.4.23.3 Utilisation de NameOfStation en tant qu'identifiant de client

Si l'appareil doit obtenir ses paramètres IP à l'aide du protocole DHCP et utiliser NameOfStation comme identifiant de client, l'option 61 doit prendre les valeurs suivantes (en plus de celles du RFC 2132):

SuboptionDHCP = 61

DHCPParameterLength = 1

DHCPParameterData = 0

Si le nom de la station de l'appareil a été modifié, la requête DHCP doit automatiquement utiliser le nouveau NameOfStation dans la requête DHCP suivante.

4.3.1.4.23.4 Utilisation de l'identifiant de client aléatoire

Si l'appareil doit obtenir ses paramètres IP à l'aide du protocole DHCP et utiliser une chaîne aléatoire comme identifiant de client, l'option 61 doit prendre les valeurs suivantes:

SuboptionDHCP = 61

DHCPParameterLength => 1

DHCPParameterData[first octet] = 0

DHCPParameterData[second octet ...n] = soit "NameOfStation" soit "AliasName"

4.3.1.4.24 Codage des champs DHCPParameterValue à l'aide de l'Option 255 de DHCP

L'option DHCP 255 est définie comme END par RFC 2131. Elle est automatiquement utilisée par le client ou le serveur DHCP. Par conséquent, cette option est surchargée par DCP et doit être utilisée de la manière suivante:

SuboptionDHCP	= 255
DHCPPParameterLength	= 1
DHCPPParameterData	= 0: Pas d'utilisation de DHCP (Par défaut)
	= 1: Pas d'utilisation de DHCP mais configuration de toutes les DHCPOptions à leur valeur "Reset to factory" (réinitialisée à la valeur d'usine)
	= 2: Utilisation de DHCP avec l'ensemble donné de DHCPOptions

NOTE 2 Si DHCP est activé, alors toutes les valeurs d'IPAddress, de Subnetmask et de StandardGateway établies précédemment sont supprimées. Ces champs sont désormais contrôlés par DHCP.

Il convient que cette option soit la dernière de la liste des options DHCP dans une demande.

4.3.1.4.25 Codage du champ ManufacturerOUI

Ce champ doit être codé en OctetString[3] avec l'OUI (Organizationally Unique Identifier) défini par l'IEEE 802.

NOTE La valeur de ce numéro administratif central est donnée par l'autorité d'enregistrement IEEE. Disponible à l'adresse suivante: <<http://standards.ieee.org/regauth>>

4.3.1.4.26 Codage du champ ManufacturerSpecificString

Ce champ doit être codé en type de données OctetString et contenir les valeurs de l'option et des sous-options associées. Si la longueur de BlockValue est impaire, un octet de bourrage doit être ajouté pour être aligné à Unsigned16. La valeur de l'octet de bourrage doit être 0.

4.3.1.4.27 Codage du champ DeviceVendorValue

Ce champ doit être codé en type de données VisibleString et contenir le même contenu que le champ DeviceType sans les espaces de fin de champ.

4.3.1.4.28 Codage du champ DHCPPParameterData

Ce champ doit être codé en type de données OctetString et contenir les valeurs de l'option et des sous-options associées. Si la longueur de BlockValue est impaire, un octet de bourrage doit être ajouté pour être aligné à Unsigned16. La valeur de l'octet de bourrage doit être 0.

4.3.1.4.29 Codage du champ AddDataValue

Ce champ doit être codé en type de données OctetString et contenir les valeurs de l'option et des sous-options associées. Si la longueur de BlockValue est impaire, un octet de bourrage doit être ajouté pour être aligné à Unsigned16. La valeur de l'octet de bourrage doit être 0.

4.3.2 Diagrammes d'états de protocole DCP

4.3.2.1 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.3.2.2 Machines de protocole de relation entre applications

4.3.2.2.1 Emetteur de diffusion unique DCP

4.3.2.2.1.1 Définitions de primitives

4.3.2.2.1.1.1 Primitives échangées entre les machines distantes

Les primitives de service distantes, y compris leurs paramètres associés, émises ou reçues par DCPUCS sont décrites dans la définition de service et présentées dans le Tableau 78.

Tableau 78 – Primitives distantes émises ou reçues par DCPUCS

Primitive	Source	Destination	Paramètres associés	Fonctions
DCP_Get.req	Moyens locaux	DCPUCS	CREP, DA, ListOfOptions	Ce service est utilisé pour extraire les données de l'ASE DCP distant. Le paramètre DA contient l'adresse de Layer2 du serveur et le paramètre ListOfOptions, les informations requises.
DCP_Set.req	CTLDINA	DCPUCS	CREP, DA, ListOfData, ListOfControlCommands	Ce service est utilisé pour acheminer les données de l'ASE DCP distant. Le paramètre DA contient l'adresse de Layer2 du serveur et le paramètre ListOfData, les informations requises. Le paramètre ListOfControlCommand contient les informations de direction.
LMPM_A_Data.cnf (-)	LMPM	DCPUCS	CREP, LMPM_status	—
LMPM_A_Data.cnf (+)	LMPM	DCPUCS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPUCS	CREP, DA, SA, Prio, A_SDU	—
DCP_Get.cnf (-)	DCPUCS	Moyens locaux	ERRCLS, ERRCODE	—
DCP_Get.cnf (+)	DCPUCS	Moyens locaux	ListOfData	—
DCP_Set.cnf (-)	DCPUCS	CTLDINA	ERRCLS, ERRCODE	—
DCP_Set.cnf (+)	DCPUCS	CTLDINA	CREP, ListOfResponse	Ce service transporte les données et les informations d'exécution.
LMPM_A_Data.req	DCPUCS	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.1.1.2 Primitives échangées entre les machines locales

Les primitives locales ainsi que leurs paramètres associés émises ou reçues par DCPUCS sont décrites dans la définition de service et présentées dans le Tableau 79.

Tableau 79 – Primitives locales émises ou reçues par DCPUCS

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.3.2.2.1.2 Schéma de transition d'états

Le schéma de transition d'états du DCPUCS est présenté à la Figure 14.

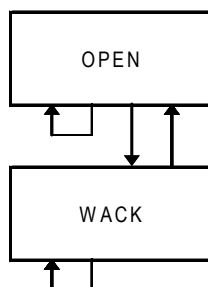


Figure 14 – Schéma de transition d'états de DCPUCS

Etats du DCPUCS

OPEN	Attend une demande de DCP_Get ou de DCP_Set de l'application et l'envoie au serveur. Ensuite, passe à l'état WACK et attend la réponse.
WACK	Attend une réponse de DCP_Get ou de DCP_Set du serveur et génère une confirmation de DCP_Get ou de DCP_Set. Si le serveur ne répond pas, réitérer la demande et une fois que la limite de tentatives est dépassée, créer une confirmation négative.

4.3.2.2.1.3 Description du diagramme d'états

Le diagramme d'états obtient ses valeurs initiales à partir des attributs DCP ASE. A l'état OPEN, les demandes de service DCP_Get et Set émises par l'utilisateur DCP sont transformées en services Data Link Layer. A la suite à la transmission des données, l'état WACK permet d'attendre la Response-PDU générant une primitive de confirmation destinée à l'utilisateur DCP.

4.3.2.2.1.4 Table d'états DCPUCS

Le Tableau 80 contient la description exhaustive du diagramme d'états de DCPUCS.

Tableau 80 – Table d'états DCPUCS

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	OPEN	DCP_Get.req () => Pending:= GET DA := From instance identified by CREP XID := SXID A_SDU := DCP-GetReqPDU with parameters from the request Store A_SDU Retry:= Max Retry Limit StartTimer (UC Client Timeout) LMPM_A_Data.req ()	WACK

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
2	OPEN	DCP_Set.req () => Pending:= SET DA := From instance identified by CREP XID := SXID A_SDU := DCP-SetReqPDU with parameters from the request Store A_SDU Retry:= Max Retry Limit StartTimer (UC Client Timeout) LMPM_A_Data.req ()	WACK
3	OPEN	LMPM_A_Data.ind () => ignore	OPEN
4	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
5	OPEN	TimerExpired (UC Client Timeout) => ignore	OPEN
6	WACK	DCP_Get.req () => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Get.cnf (-)	WACK
7	WACK	DCP_Set.req () => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Set.cnf (-)	WACK
8	WACK	LMPM_A_Data.ind () /DGP-Get-ResPDU && Pending == GET && XID == SXID => SXID := SXID+1 StopTimer (UC Client Timeout) DCP_Get.cnf (+)	OPEN
9	WACK	LMPM_A_Data.ind () /DGP-Get-ResPDU && (Pending != GET XID != SXID) => ignore	WACK
10	WACK	LMPM_A_Data.ind () /DGP-Set-ResPDU && Pending == SET && XID == SXID => SXID:= SXID+1 StopTimer (UC Client Timeout) DCP_Set.cnf (+)	OPEN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
11	WACK	LMPM_A_Data.ind () /DCP-Set-ResPDU && (Pending != SET XID != SXID) => ignore	WACK
12	WACK	LMPM_A_Data.cnf (+) /LMPM_status == OK => ignore	WACK
13	WACK	LMPM_A_Data.cnf (-) /LMPM_status != OK && Pending == GET => ERRCLS := PROTOCOL ERRCODE := LMPM StopTimer (UC Client Timeout) DCP_Get.cnf (-)	OPEN
14	WACK	LMPM_A_Data.cnf (-) /LMPM_status != OK && Pending == SET => ERRCLS := PROTOCOL ERRCODE := LMPM StopTimer (UC Client Timeout) DCP_Set.cnf (-)	OPEN
15	WACK	TimerExpired (UC Client Timeout) /Retry != 0 => A_SDU:=from Stored A_SDU Retry:= Retry-1 StartTimer (UC Client Timeout) LMPM_A_Data.req ()	WACK
16	WACK	TimerExpired (UC Client Timeout) /Retry == 0 && Pending == GET => ERRCLS := PROTOCOL ERRCODE := TIMEOUT DCP_Get.cnf (-)	OPEN
17	WACK	TimerExpired (UC Client Timeout) /Retry == 0 && Pending == SET => ERRCLS := PROTOCOL ERRCODE := TIMEOUT DCP_Set.cnf (-)	OPEN
18	WACK	LMPM_A_Data.ind () /!(DCP-Set-ResPDU DCP-Get-ResPDU) => ignore	WACK

4.3.2.2.1.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 81 contient les fonctions, macros, temporisateurs et variables utilisé par le DCPUCS ainsi que leurs arguments et leurs descriptions.

Tableau 81 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPUCS

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction est utilisée pour lancer ou relancer un temporisateur.
StopTimer	Fonction	Cette fonction est utilisée pour arrêter un temporisateur.
TimerExpired	Fonction	Cette fonction signale qu'un temporisateur a expiré.
UC Client Timer	Temporisateur	Ce temporisateur, relancé sur le demandeur lors de l'envoi d'une demande à diffusion unique, met fin à l'attente d'une réponse.
ERRCLS	Variable	Cette variable locale contient l'entité signalant l'erreur.
ERRCODE	Variable	Cette variable locale contient la raison de l'erreur dans le contexte du ERRCLS.
Max Retry Limit	Variable	Cette variable locale contient la limite de nouvelles tentatives.
Pending	Variable	Cette variable locale contient des informations indiquant si le service Set ou Get est en attente.
Retry	Variable	Cette variable locale contient le nombre de nouvelles tentatives. La valeur maximale doit être issue de l'attribut DCP ASE approprié.
SXID	Variable	Cette variable locale contient le Transaction ID XID utilisé pour identifier les services de manière unique. La valeur initiale doit être déduite d'un générateur de nombres aléatoires.
UC Client Timeout	Variable	La valeur de cette variable locale doit être issue de l'attribut DCP ASE approprié. La valeur par défaut est 1 s.

4.3.2.2.2 Destinataire de diffusion unique DCP

4.3.2.2.2.1 Définitions de primitives

4.3.2.2.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par DCPUCR, sont décrites dans la définition de service et présentées dans le Tableau 82.

Tableau 82 – Primitives distantes émises ou reçues par DCPUCR

Primitive	Source	Destination	Paramètres associés	Fonctions
DCP_Set.rsp	CMINA	DCPUCR	CREP, ListOfResponse	—
LMPM_A_Data.cnf	LMPM	DCPUCR	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPUCR	CREP, DA, SA, Prio, A_SDU	—
DCP_Set.ind	DCPUCR	CMINA	CREP, ListOfData, ListOfControlCommands	—
LMPM_A_Data.req	DCPUCR	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.1.2 Primitives échangées entre les machines locales

Les primitives locales, y compris leurs paramètres associés, émises ou reçues par DCPUCR, sont décrites dans la définition de service et présentées dans le Tableau 83.

Tableau 83 – Primitives locales émises ou reçues par DCPUCR

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.3.2.2.2 Schéma de transition d'états

Le schéma de transition d'états du DCPUCR est présenté à la Figure 15.

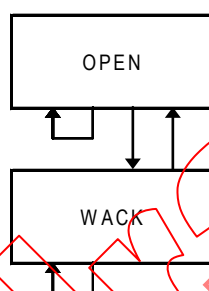


Figure 15 – Schéma de transition d'états de DCPUCR

Etats du DCPUCR

OPEN

Attendre un service DCP Get ou Set et prépare la réponse pour GET

WACK

Attendre la réponse de l'application pour SET pour créer la réponse de DCP Set

4.3.2.2.3 Description du diagramme d'états

A l'état OPEN, le diagramme d'états attend la première indication de service LMPM_A_Data. Il passe à l'état WACK après avoir transmis l'indication à l'utilisateur. Un service de réponse active une requête LMPM_A_Data et remet le diagramme d'états à l'état OPEN.

4.3.2.2.4 Table d'états DCPUCR

Le Tableau 84 contient la description exhaustive du diagramme d'états de DCPUCR. Une primitive LMPM_A_Data.ind est acceptée si le type est DCP-Get-ResPDU ou DCP-Set-ResPDU.

Tableau 84 – Table d'états DCPUCR

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	OPEN	DCP_Set.rsp () => ignore	OPEN
2	OPEN	LMPM_A_Data.ind () /DCP-Get-ReqPDU && (SAM == NIL SAM == SA) && (DA != Multicast) && ValidSelectors => SAM := SA SXID := XID SDA := SA DA := SDA XID := SXID A_SDU := Create DCP-Get-ResPDU with ListOfData taken from the DCP ASE according the ListOfData of the DCP-Get-ReqPDU StartTimer (Client Hold Time) LMPM_A_Data.req ()	OPEN
3	OPEN	LMPM_A_Data.ind () /DCP-Get-ReqPDU && (SAM == NIL SAM == SA) && (DA != Multicast) && !ValidSelectors => SAM := SA SXID := XID SDA := SA DA := SDA XID := SXID ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet A_SDU := Create DCP-Get-ResPDU with error information StartTimer (Client Hold Time) LMPM_A_Data.req ()	OPEN
4	OPEN	LMPM_A_Data.ind () /DCP-Set-ReqPDU && (SAM == NIL SAM == SA) && (DA != Multicast) => SAM := SA SXID := XID SDA := SA DCP_Set.ind (CREP, ListOfData)	WACK
5	OPEN	LMPM_A_Data.ind () /DCP-Get-ReqPDU && (SAM != NIL && SAM != SA) => ignore	OPEN
6	OPEN	LMPM_A_Data.ind () /DCP-Set-ReqPDU && (SAM != NIL && SAM != SA) => ignore	OPEN
7	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
8	OPEN	TimerExpired (Client Hold Time) => SAM:= NIL	OPEN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
9	WACK	DCP_Set.rsp () => DA := SDA XID := SXID A_SDU := DCP-Set-ResPDU containing the response from the application StartTimer (Client Hold Time) LMPM_A_Data.req ()	OPEN
10	WACK	LMPM_A_Data.ind () => ignore	WACK
11	WACK	LMPM_A_Data.cnf () => ignore	WACK

4.3.2.2.2.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 85 contient les fonctions, macros, temporisateurs et variables utilisés par le DCPUCR ainsi que leurs arguments et leurs descriptions.

Tableau 85 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPUCR

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction est utilisée pour lancer ou relancer un temporisateur
StopTimer	Fonction	Cette fonction est utilisée pour arrêter un temporisateur
TimerExpired	Fonction	Cette fonction est émise si un temporisateur expire
ValidSelectors	Macro	Vérifier le PDU par rapport à la définition de DCP et à l'ASE DCP
Client Hold Timer	Temporisateur	Ce temporisateur est utilisé pour protéger le serveur d'un accès par plusieurs clients à l'état OPEN. Il protège le temps de réaction du client pour la prochaine demande et ne dépend pas du temps de réaction du serveur.
Client Hold Time	Variable	La valeur est ce temps est de 3 s par défaut.
SAM (Standard Antenna Method)	Variable	Cette variable locale contient la SA du client qui est à ce moment la "propriétaire" du serveur.
SDA	Variable	Cette variable locale contient la dernière SA valide acheminée avec une indication de LMPM_A_Data avec des données valides et est utilisée pour la génération d'une réponse
SXID	Variable	Cette variable locale contient le Transaction ID XID permettant d'identifier les services de manière unique. La valeur doit être issue du PDU de la demande.

4.3.2.2.3 Emetteur multidiffusion DCP

4.3.2.2.3.1 Définitions de primitives

4.3.2.2.3.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par DCPMCS sont décrites dans la définition de service et présentées dans le Tableau 86.

Tableau 86 – Primitives distances émises ou reçues par DCPMCS

Primitive	Source	Destination	Paramètres associés	Fonctions
DCP_Identify.req	CTLDINA	DCPMCS	ListOfFilter, ResponseDelayFactor	—
LMPM_A_Data.cnf (-)	LMPM	DCPMCS	CREP, LMPM_status	—
LMPM_A_Data.cnf (+)	LMPM	DCPMCS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPMCS	CREP, DA, SA, Prio, A_SDU	—
DCP_Identify.cnf (-)	DCPMCS	CTLDINA	CREP, ERRCLS, ERRCODE	—
DCP_Identify.cnf (+)	DCPMCS	CTLDINA	ListOfDevices	—
LMPM_A_Data.req	DCPMCS	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.3.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DCPMCS sont décrites dans la définition de service et présentées dans le Tableau 87.

Tableau 87 – Primitives locales émises ou reçues par DCPMCS

Primitive	Source	Destination	Paramètres associés	Fonctions
DCP_Identify_ind	DCPMCS	CTLDINA	SA, ListOfDevices	—

4.3.2.2.3.2 Schéma de transition d'états

Le schéma de transition d'états du DCPMCS est présenté à la Figure 16.

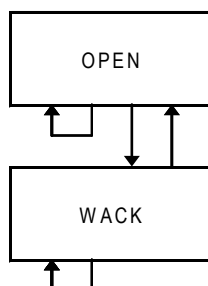


Figure 16 – Schéma de transition d'états de DCPMCS

Etats du DCPMCS

OPEN	Attend une demande du service Identify.
WACK	Attend toutes les réponses jusqu'à une certaine heure.

4.3.2.2.3.3 Description du diagramme d'états

La machine attend à l'état OPEN une demande de service Identify. Suite à la transmission de données, l'état WACK est utilisé pour attendre toutes les réponses pendant un certain temps. Le délai d'attente génère la primitive de service de confirmation Identify avec toutes les réponses reçues destinée à l'utilisateur DCP, et définit le diagramme d'états à l'état OPEN.

4.3.2.2.3.4 Table d'états DCPMCS

Le Tableau 88 contient la description exhaustive du diagramme d'états de DCPMCS. Une primitive LMPM_A_Data.ind doit être acceptée si le type est DCP-IdentifyResPDU.

Tableau 88 – Table d'états DCPMCS

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	OPEN	DCP_Identify.req () => XID := SXID DA := DCP_MulticastMACAdd A_SDU := DCP-Identify-ReqPDU First := TRUE StartTimer (Response delay timeout) LMPM_A_Data.req ()	WACK
2	OPEN	LMPM_A_Data.ind () => ignore	OPEN
3	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
4	OPEN	TimerExpired (Response delay timeout) => ignore	OPEN
5	WACK	DCP_Identify.req () => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Identify.cnf (-)	WACK
6	WACK	LMPM_A_Data.ind () /(DCP-Identify-ResPDU && XID == SXID) && FIRST == FALSE => ListOfDevices.SA += SA ListOfDevices.ListOfData += A_SDU	WACK
7	WACK	LMPM_A_Data.ind () /(DCP-Identify-ResPDU && XID == SXID) && First == TRUE => ListOfDevices.SA += SA ListOfDevices.ListOfData += A_SDU First := FALSE DCP_Identify_ind (SA, ListOfDevices)	WACK

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
8	WACK	LMPM_A_Data.ind () /(DCP-Identify-ResPDU && XID == SXID) => ignore	WACK
9	WACK	LMPM_A_Data.cnf (+) => ignore	WACK
10	WACK	LMPM_A_Data.cnf (-) => ERRCLS := PROTOCOL ERRCODE := LMPM StopTimer (Response delay timeout) DCP_Identify.cnf (-)	OPEN
11	WACK	TimerExpired (Response delay timeout) => ListOfDevices := Stored ListOfDevices SXID := SXID + 1 DCP_Identify.cnf (+)	OPEN

4.3.2.2.3.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 89 contient les fonctions, macros, temporisateurs et variables utilisés par le DCPMCS ainsi que leurs arguments et leurs descriptions.

Tableau 89 – Fonctions utilisées par le DCPMCS

Nom	Type	Fonction/Signification
SXID	Variable	Cette variable locale contient le Transaction ID XID utilisé pour identifier les services de manière unique. Elle doit être modifiée à chaque demande.
First (Première)	Variable	Cette variable est utilisée pour accélérer l'établissement de la connexion.
Response delay timer	Temporisateur	Ce temporisateur est utilisé pour mettre en place un délai d'attente pour la réponse du service Identify de DCP
Délai de réponse	Variable	La valeur est issue du ResponseDelayFactor
TimerExpired	Fonction	Cette fonction est émise si un temporisateur expire
StartTimer	Fonction	Cette fonction est utilisée pour lancer ou relancer un temporisateur
StopTimer	Fonction	Cette fonction est utilisée pour arrêter un temporisateur

4.3.2.2.4 Destinateur multidiffusion DCP

4.3.2.2.4.1 Définitions de primitives

4.3.2.2.4.1.1 Primitives échangées entre les machines distantes

Les primitives de service ainsi que leurs paramètres associés émis ou reçus par DCPMCR sont décrits dans la définition de service et présentés dans le Tableau 90.

Tableau 90 – Primitives distantes émises ou reçues par DCPMCR

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_A_Data.cnf	LMPM	DCPMCR	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPMCR	CREP, DA, SA, Prio, A_SDU	—
LMPM_A_Data.req	DCPMCR	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.4.1.2 Primitives échangées entre les machines locales

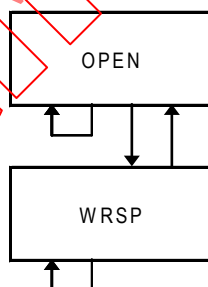
Les primitives locales, y compris leurs paramètres associés, émises ou reçues par DCPMCR sont décrites dans la définition de service et présentées dans le Tableau 91.

Tableau 91 – Primitives locales émises ou reçues par DCPMCR

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.3.2.2.4.2 Schéma de transition d'états

Le schéma de transition d'états du DCPMCR est présenté à la Figure 17.

**Figure 17 – Schéma de transition d'états du DCPMCR**

Etats du DCPMCR

OPEN	Attendre des DCP-IdentifyReqPDU et compare les données locales avec la ListOfFilter. Si une correspondance est détectée, lance le délai de réponse est lancé et le passage à l'état suivant est détecté.
WRSP	Attendre le délai de réponse pour envoyer la réponse lorsque l'utilisateur de DCP a reconnu une correspondance avec les données locales.

4.3.2.2.4.3 Description du diagramme d'états

Le diagramme d'états attend DCP-IdentifyReqPDUs à l'état OPEN. La liste de filtres doit être transmise à l'utilisateur DCP pour vérifier les critères de filtrage. Le diagramme d'états attend à l'état WRSP un certain temps, calculé grâce au facteur de délai de réponse des clients, pour envoyer la réponse si l'utilisateur DCP a reconnu une correspondance avec les données locales. Sinon, aucune donnée ne doit être envoyée.

4.3.2.2.4.4 Table d'états DCPMCR

Le Tableau 92 contient la description exhaustive du diagramme d'états de DCPMCR. Une primitive LMPM_A_Data.ind doit être acceptée si le Type est DCP-DCP-IdentifyReqPDU.

Tableau 92 – Table d'états DCPMCR

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	OPEN	LMPM_A_Data.ind () /DCP-Identify-ReqPDU && Successful comparison with ListOfFilters => SAM := SA SXID := XID ResponseDelay :=CalculateResponseDelay(ResponseDelayFactor, MACAddress) StartTimer (ResponseDelay)	WRSP
2	OPEN	LMPM_A_Data.ind () /DCP-Identify-ReqPDU && ! Successful comparison with ListOfFilters => ignore	OPEN
3	OPEN	LMPM_A_Data.ind () /!DCP-Identify-ReqPDU => ignore	OPEN
4	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
5	WRSP	TimerExpired (ResponseDelay) => DA = SAM XID := SXID A_SDU := Create-DCP-Identify-ResPDU with ListOfData taken from the DCP ASE according the ListOfFilters of the DCP-Identify-ReqPDU LMPM_A_Data.req ()	OPEN
6	WRSP	LMPM_A_Data.ind () => ignore	WRSP
7	WRSP	LMPM_A_Data.cnf () => ignore	WRSP

4.3.2.2.4.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 93 contient les fonctions ou macros utilisés par le DCPMCR ainsi que leurs arguments et leurs descriptions.

Tableau 93 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPMCR

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction locale lance ou relance un temporisateur
StopTimer	Fonction	Cette fonction locale arrête un temporisateur
TimerExpired	Fonction	Cette fonction locale est émise si un temporisateur expire
CalculateResponseDelay	Macro	Cette macro locale calcule le ResponseDelay conformément aux règles de la présente norme à l'aide du ResponseDelayFactor et de MACAddress de l'interface.

Nom	Type	Fonction/Signification
Successful comparison with ListOfFilters	Macro	Vérifie le PDU par rapport aux paramètres stockés dans l'ASE DCP
ResponseDelayTimer	Temporisateur	Ce temporisateur est utilisé pour générer le délai de réponse
ResponseDelay	Variable	Cette variable contient la valeur conforme aux règles de calcul mentionnées dans la présente norme.
SAM (Standard Antenna Method)	Variable	Cette variable locale contient la SourceAddress du demandeur utilisée pour la réponse.
SXID	Variable	Cette variable locale contient le Transaction ID XID utilisé pour identifier les services de manière unique.

4.3.2.2.5 Emetteur Multidiffusion Hello de DCP

4.3.2.2.5.1 Définitions de primitives

4.3.2.2.5.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par DCPHMCS sont décrites dans la définition de service et présentées dans le Tableau 94.

Tableau 94 – Primitives distantes émises ou reçues par DCPHMCS

Primitive	Source	Destination	Paramètres associés	Fonctions
DCP_Hello.req	CMINA	DCPHMCS	ListOfData	—
LMPM_A_Data.cnf (-)	LMPM	DCPHMCS	CREP, LMPM_status	—
LMPM_A_Data.cnf (+)	LMPM	DCPHMCS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPHMCS	CREP, DA, SA, Prio, A_SDU	—
DCP_Hello.cnf (-)	DCPHMCS	CMINA	ERRCLS, ERRCODE	—
DCP_Hello.cnf (+)	DCPHMCS	CMINA	—	—
LMPM_A_Data.req	DCPHMCS	LMPM	CREP, DA, SA, Prio, A_SDU	—

4.3.2.2.5.1.2 Primitives échangées entre les machines locales

Les primitives locales, y compris leurs paramètres associés, émises ou reçues par DCPHMCS sont décrites dans la définition de service et présentées dans le Tableau 95.

Tableau 95 – Primitives locales émises ou reçues par DCPHMCS

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.3.2.2.5.2 Schéma de transition d'états

Le schéma de transition d'états du DCPHMCS est présenté à la Figure 18.

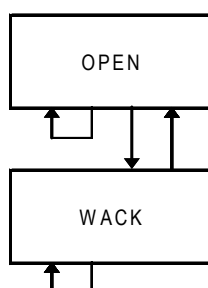


Figure 18 – Schéma de transition d'états de DCPHMCS

Etats du DCPHMCS

OPEN	Attendre une demande du service Hello de la couche supérieure
WACK	Attendre une confirmation de Hello de la couche inférieure

4.3.2.2.5.3 Description du diagramme d'états

La machine attend à l'état OPEN une demande du service Hello. A l'issue de la transmission des données, l'état WACK permet d'attendre la confirmation de transmission.

4.3.2.2.5.4 Table d'états DCPHMCS

Le Tableau 96 contient la description exhaustive du diagramme d'états de DCPHMCS.

Tableau 96 – Table d'états DCPHMCS

#	Etat en cours	Evénement /Condition => Action	Etat suivant
1	OPEN	DCP_Hello.req () => DA := DCP_MulticastMACadd for Hello A_SDU := DCP_Hello-ReqPDU LMPM_A_Data.req ()	WACK
2	OPEN	LMPM_A_Data.ind () => ignore	OPEN
3	OPEN	LMPM_A_Data.cnf () => ignore	OPEN
4	WACK	LMPM_A_Data.ind () => ignore	WACK
5	WACK	LMPM_A_Data.cnf (+) => DCP_Hello.cnf (+)	OPEN
6	WACK	LMPM_A_Data.cnf (-) => ERRCLS := PROTOCOL ERRCODE := LMPM DCP_Hello.cnf (-)	OPEN

4.3.2.2.5.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 97 contient les fonctions, macros, temporisateurs et variables utilisés par le DCPHMCS ainsi que leurs arguments et leurs descriptions.

Tableau 97 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPHMCS

Nom	Type	Fonction/Signification
SXID	Variable	Le XID doit être ignoré par le destinataire de Hello. Il peut être incrémenté par l'émetteur.

4.3.2.2.6 Destinataire Multidiffusion Hello de DCP

4.3.2.2.6.1 Définitions de primitives

4.3.2.2.6.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par DCPHMCR sont décrites dans la définition de service et présentées dans le Tableau 98.

Tableau 98 – Primitives distantes émises ou reçues par DCPHMCR

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_A_Data.cnf	LMPM	DCPHMCR	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	DCPHMCR	CREP, DA, SA, Prio, A_SDU	—
DCP_Hello.ind	DCPHMCR	CTLDNA	ListOfData	—

4.3.2.2.6.1.2 Primitives échangées entre les machines locales

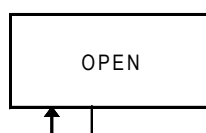
Les primitives locales, y compris leurs paramètres associés, émises ou reçues par DCPHMCR sont décrites dans la définition de service et présentées dans le Tableau 99.

Tableau 99 – Primitives locales émises ou reçues par DCPHMCR

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.3.2.2.6.2 Schéma de transition d'états

Le schéma de transition d'états du DCPHMCR est présenté à la Figure 19.

**Figure 19 – Schéma de transition d'états de DCPHMCR**

Etats du DCPHMCR

OPEN

Attendre des DCP-Hello-ReqPDU

4.3.2.2.6.3 Description du diagramme d'états

Le diagramme d'états attend des DCP-Hello-ReqPDU à l'état OPEN.

4.3.2.2.6.4 Table d'états DCPHMC

Le Tableau 100 contient la description exhaustive du diagramme d'états de DCPHMC. Une primitive LMPM_A_Data.ind sera acceptée si le Type est DCP-DCP-IdentifyReqPDU.

Tableau 100 – Table d'états DCPHMC

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	OPEN	LMPM_A_Data.ind () /DCP-Hello-ReqPDU && Successful comparison with ListOfData => DCP_Hello.ind ()	OPEN
2	OPEN	LMPM_A_Data.ind () /!DCP-Hello-ReqPDU => ignore	OPEN
3	OPEN	LMPM_A_Data.cnf () => ignore	OPEN

4.3.2.2.6.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 101 contient les fonctions, macros, temporisateurs et variables utilisés par le DCPHMC ainsi que leurs arguments et leurs descriptions.

Tableau 101 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DCPHMC

Nom	Type	Fonction/Signification
Successful comparison with ListOfData	Macro	Compare le contenu des PDU avec les informations stockées localement et retourne "TRUE" en cas de concordance.

4.3.3 Machines de Protocole de Mapping DLL

Il n'existe aucune Machine de Protocole de Mapping DLL (DMPM) définie pour ce Protocole.

4.4 Contrôle de temps de travail précis

4.4.1 Description de la syntaxe FAL

4.4.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.4.1.2 Syntaxe abstraite APDU

Le Tableau 102 définit la syntaxe abstraite des PDU PTCP appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU.

Tableau 102 – Syntaxe PTCP APDU

Nom de l'APDU	Structure de l'APDU
PTCP-PDU	PTCP-SYNC-PDU ^ PTCP-FUP-PDU ^ PTCP-ANNOUNCE-PDU ^ PTCP-DELAY-REQ-PDU ^ PTCP-DELAY-RSP-PDU ^ PTCP-DELAY-FU-RSP-PDU
PTCP-FUP-PDU	PTCP-Header-FUP, PTCP-FollowUpPDU
PTCP-SYNC-PDU	PTCP-Header-Sync, PTCP-RTSyncPDU
PTCP-ANNOUNCE-PDU	PTCP-Header-Announce, PTCP-AnnouncePDU
PTCP-DELAY-REQ-PDU	PTCP-Header-Delay, PTCP-DelayReqPDU
PTCP-DELAY-RSP-PDU	PTCP-Header-Delay, PTCP-DelayResPDU
PTCP-DELAY-FU-RSP-PDU	PTCP-Header-Delay, PTCP-DelayFuResPDU
PTCP-RTSyncPDU	PTCPSubdomain, PTCPTime, PTCPTimeExtension, PTCPMaster, [PTCPOption], PTCPEnd, [APDU_Status] ^a
PTCP-AnnouncePDU	PTCPSubdomain, PTCPMaster, [PTCPOption], PTCPEnd
PTCP-FollowUpPDU	PTCPSubdomain, PTCPTime, [PTCPOption], PTCPEnd
PTCP-DelayReqPDU	PTCPDelayParameter, [PTCPOption], PTCPEnd
PTCP-DelayResPDU	PTCPDelayParameter, PTCPPortParameter, PTCPPortTime, [PTCPOption], PTCPEnd
PTCP-DelayFuResPDU	PTCPDelayParameter, [PTCPOption], PTCPEnd

^a Prise en charge de la version originelle

Le Tableau 103 définit les structures pour la substitution des éléments des structures APDU.

Tableau 103 – Substitutions PTCP

Nom de substitution	Structure
PTCP-Header-Sync	PTCP_reserved_1, PTCP_reserved_2, PTCP_Delay10ns, PTCP_SequenceID, PTCP_Delay1ns_Byte ^d , Padding ^a , PTCP_Delay1ns
PTCP-Header-FUP	[HWPadding*] ^b , PTCP_SequenceID, [Padding*] ^c , PTCP_Delay1ns_FUP
PTCP-Header-Announce	[HWPadding*] ^b , PTCP_SequenceID, [Padding*] ^e
PTCP-Header-Delay	[HWPadding*] ^b , PTCP_SequenceID, [Padding*] ^c , PTCP_Delay1ns
PTCPSubdomain	PTCP_TLVHeader, PTCP_MasterSourceAddress, PTCP_SubdomainUUID
PTCPTime	PTCP_TLVHeader, PTCP_EPOCHNumber, PTCP_Seconds, PTCP_NanoSeconds
PTCPTimeExtension	PTCP_TLVHeader, PTCP_Flags, PTCP_CurrentUTCOffset, [Padding*] ^a
PTCPMaster	PTCP_TLVHeader, PTCP_MasterPriority1, PTCP_MasterPriority2, PTCP_ClockClass, PTCP_ClockAccuracy, PTCP_ClockVariance, [Padding*] ^a
PTCPDelayParameter	PTCP_TLVHeader, PTCP_PortMACAddress
PTCPPortParameter	PTCP_TLVHeader, [Padding*] ^a , PTCP_T2PortRxDelay, PTCP_T3PortTxDelay
PTCPPortTime	PTCP_TLVHeader, [Padding*] ^a , PTCP_T2TimeStamp
PTCPOption	PTCP_TLVHeader, PTCP_OUI, PTCP_SubType, Data*, [Padding*] ^a
PTCPend	PTCP_TLVHeader(0)

^a L'alignement sur 32 bits doit être assuré

^b Bourrage de 12 octets

^c Bourrage de 2 octets

^d Version originelle, pour répondre aux besoins matériels

^e Bourrage de 6 octets

4.4.1.3 Section de codage relative aux champs de base communs

4.4.1.3.1 Codage du champ PTCP_reserved_1

Ce champ doit être codé en type de données Unsigned32.

4.4.1.3.2 Codage du champ PTCP_reserved_2

Ce champ doit être codé en type de données Unsigned32.

4.4.1.3.3 Codage du champ HWPadding

Ce champ doit être codé en type de données Unsigned8.

4.4.1.3.4 Codage du champ PTCP_TLVHeader

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 8: PTCP_TLVHeader.Length

La valeur contient la somme des octets consécutifs du bloc concerné.

Bit 9 – 15: PTCP_TLVHeader.Type

Ce champ doit être codé avec les valeurs conformes au Tableau 104.

Tableau 104 – PTCP_TLVHeader.Type

Valeur (hexadécimale)	Signification	Utilisation
0x00	PTCPEnd	Obligatoire
0x01	PTCPSubdomain	Obligatoire
0x02	PTCPTime	Obligatoire
0x03	PTCPTimeExtension	Obligatoire
0x04	PTCPMaster	Obligatoire
0x05	PTCPPortParameter	Obligatoire
0x06	PTCPDelayParameter	Obligatoire
0x07	PTCPPortTime	Obligatoire
0x08 – 0x7E	Réservé	—
0x7F	PTCPOption Spécifique à l'organisation	Facultatif

4.4.1.3.5 Section de codage relative à PTCP-Header

Ce champ contient le délai de propagation, en nanosecondes. Il est divisé en un champ pour la partie 10 ns et un champ pour la partie 1 ns. Tous les champs relatifs au délai sont pertinents pour Sync-, FollowUp-, DelayRes- sans DelayFuRes- et DelayFuRes-Messages.

4.4.1.3.5.1 Codage du champ PTCP_Delay10ns

Ce champ doit être codé en type de données Unsigned32 et contenir la partie égale à 10 ns du délai de propagation. Ce champ doit être codé avec les valeurs conformes au Tableau 105.

Tableau 105 – PTCP_Delay10ns

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	Valide Partie 10 ns délai de propagation
0xFFFFFFFF	Non valide Cette valeur indique un dépassement de capacité et marque les champs de délai comme étant non valides

4.4.1.3.5.2 Codage du champ PTCP_Delay1ns_Byte

Ce champ contient la partie égale à 1 ns du délai de propagation.

NOTE Version originelle, pour répondre aux besoins matériels

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0 – 3: PTCP_Delay1ns_Byte.Value

La valeur contient la partie 1 ns du délai de propagation. Ce champ doit être codé avec les valeurs conformes au Tableau 106.

Tableau 106 – PTCP_Delay1ns_Byte.Value

Valeur (hexadécimale)	Signification
0x00 – 0x09	Nombre de ns
0x0A – 0x0F	Réservé

Bit 4 – 7: PTCP_Delay1ns_Byte.Reserved

Ce champ doit être mis conformément à 3.4.2.2.

Seul le PTCP_Delay1ns peut être utilisé à la place de ce champ

4.4.1.3.5.3 Codage du champ PTCP_Delay1ns

Ce champ doit être codé en type de données Unsigned32 et contenir la partie égale à 1 ns du délai de propagation. Ce champ doit être codé avec les valeurs conformes au Tableau 107.

Tableau 107 – PTCP_Delay1ns

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	Valide Partie 1 ns délai de propagation
0xFFFFFFFF	Non valide Cette valeur indique un dépassement de capacité et marque les champs de délai comme étant non valides

4.4.1.3.5.4 Codage du champ PTCP_Delay1ns_FUP

Ce champ doit être codé en type de données Integer32 et contenir la correction du délai de propagation à 1 ns. Ce champ doit être codé avec les valeurs conformes au Tableau 108.

Tableau 108 – PTCP_Delay1ns_FUP

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	Partie en nanoseconde du délai de propagation

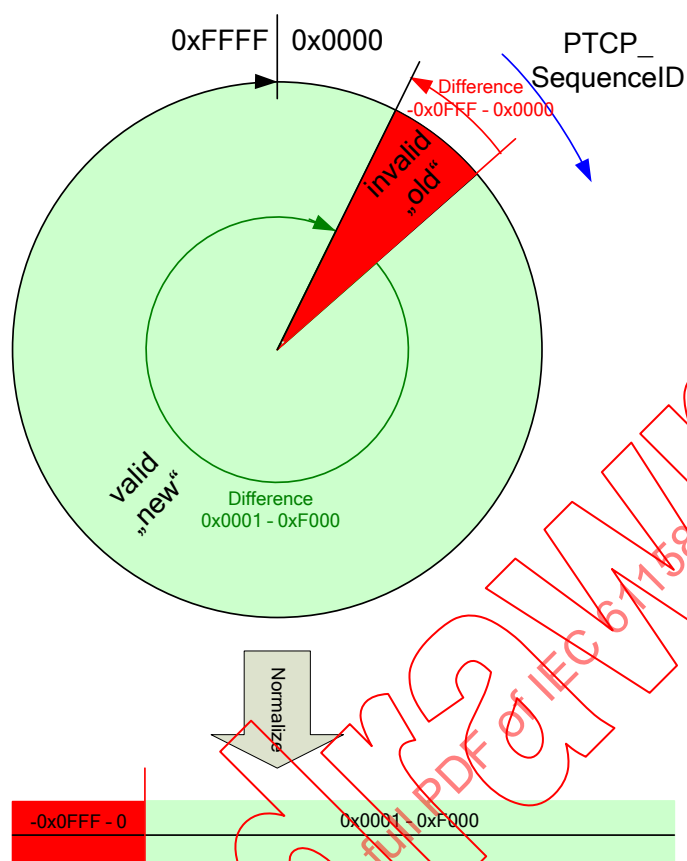
4.4.1.3.5.5 Codage du champ PTCP_SequenceID

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 109.

Tableau 109 – PTCP_SequenceID

Signification	Utilisation
Trame de synchronisation	Le maître PTCP incrémente ce champ pour chaque trame de synchronisation.
Trame de suivi	Le maître PTCP ou un suivi injectant le transitaire PTCP réplique la valeur de la trame de synchronisation.
Trame de demande de délai	Le demandeur de mesure de délai PTCP incrémente ce champ pour chaque mesure.
Trame de réponse de délai	Le répondeur réplique la valeur de la trame de demande de délai.
Trame de suivi de réponse de délai	Le répondeur utilise la valeur de la trame de réponse de délai.

Le destinataire des trames PTCP doit utiliser ce champ pour détecter les répétitions, la perte de trames et l'opportunité des données. C'est pourquoi la plage de valeurs PTCP_SequenceID doit être divisée selon le rapport 15/16 pour les valeurs valides et 1/16 pour les valeurs non valides comme présentées à la Figure 20.



Légende

Anglais	Français
Difference 0x0FFF – 0x0000	Différence 0x0FFF – 0x0000
Invalid "old"	"ancien" non valide
Valid "new"	"nouveau" valide
Normalize	Normaliser

Figure 20 – Plage de valeurs PTCP_SequenceID

4.4.1.3.6 Codage du champ PTCP_OUI

Ce champ doit être codé en OctetString[3] avec l'OUI (Organizationally Unique Identifier) défini dans l'IEEE 802.

NOTE La valeur de ce numéro administratif central est donnée par l'autorité d'enregistrement IEEE. Disponible à l'adresse suivante: <<http://standards.ieee.org/regauth>>

4.4.1.3.7 Codage du champ PTCP_SubType

Ce champ doit être codé en Unsigned8. La valeur spécifique à l'organisation. Pour l'OUI, le paramètre PTCP_SubType doit être codé conformément au Tableau 110.

Tableau 110 – PTCP_SubType for OUI (=00-0E-CF)

Valeur (hexadécimale)	Signification	Utilisation
0x00 – 0xFF	Réservé	—

4.4.1.3.8 Codage du champ PTCP_Seconds

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 111. La résolution de ce champ est de 1 s.

Tableau 111 – PTCP_Seconds

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	Temps en "s"

4.4.1.3.9 Codage du champ PTCP_NanoSeconds

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 112. La résolution de ce champ est de 1 ns.

Tableau 112 – PTCP_NanoSeconds

Valeur (hexadécimale)	Signification
0x00000000 – 0x3B9AC9FF	Temps en "ns"
0x3B9ACA00 – 0xFFFFFFFF	Réservé

4.4.1.4 Section de codage des champs spécifiques à PTCP-PDU

4.4.1.4.1 Codage du champ PTCP_MasterSourceAddress

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ Destination Address doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

4.4.1.4.2 Codage du champ PTCP_SubdomainUUID

Ce champ doit être codé en type de données UUID.

NOTE La valeur NIL indique qu'il n'y a aucune synchronisation au sein du service Read Real Sync Data.

4.4.1.4.3 Codage du champ PTCP_Flags

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 7: PTCP_Flags.reserved_1

Ce champ doit être mis conformément à 3.4.2.2.

Bit 8 – 9: PTCP_Flags.LeapSecond

Ce champ doit être mis conformément au Tableau 113.

Tableau 113 – PTCP_Flags.LeapSecond

Valeur (hexadécimale)	Signification
0x00	Pas de seconde intercalaire
0x01	La dernière minute comporte 61 s
0x02	La dernière minute comporte 59 s
0x03	Réservé

Bit 10 – 15: PTCP_Flags.reserved_2

Ce champ doit être mis à zéro.

4.4.1.4.4 Codage du champ PTCP_EPOCHNumber

Ce champ doit être codé en type de données Unsigned16 et contenir la partie égale à 0x100000000 s de l'heure. La valeur du champ PTCP_EPOCHNumber doit être celle d'epoch_number du jeu de données des propriétés de temps universel de l'horloge qui émet ce message conformément au Tableau 114 et au Tableau 115. L'heure de départ est similaire à 1970-01-01 T 00:00:00 Z de la CEI 61588.

Tableau 114 – Correspondance d'échelles de temps entre MJD, TUC et PTCP_EPOCHNumber

MJD (Modified Julian Day – Jour julien modifié)	TUC (Temps universel coordonné)	Décalage TUC (secondes intermédiaires)	Valeur PTCP_Seconds	Valeur PTCP_EPOCH- Number
0:00:00 15 020	1900-01-01 T 00:00:00 Z	—	—	—
0:00:00 40 587	1970-01-01 T 00:00:00 Z	—	0	0
0:00:00 41 317	1972-01-01 T 00:00:00 Z	10	63 072 000 + 10	0
16:57:44 51 357	1999-06-28 T 16:57:44 Z	32	930 589 064 + 32	0
—	2006-01-01 T 00:00:00 Z	33	—	0

NOTE Le codage AAAA-MM-JJ T HH:MM:SS Z est conforme à l'ISO 8601, où 'T' signale qu'une heure suit la date et 'Z' signale que l'heure est TUC.

Tableau 115 – Correspondance d'échelles de temps entre PTCP_EPOCHNumber, PTCP_Second, PTCP_Nanosecond, CycleCounter et SendClockFactor

Valeur CycleCounter (hexadécimale)	Valeur PTCP_EPOCHNumber (décimale)	Valeur PTCP_Second (décimale)	Valeur PTCP_Nanosecond (décimale)	Valeur SendClockFactor (décimale)
0x0000000000000000	0	0	0	32
0x0000000000001000	0	65	536 000 000	32
0x0000000100000000	0	4 294 967	296 000 000	32
0x0001000000000000	65	2 302 102 470	656 000 000	32

$$\text{PTCP\_Epoch} = \text{PTCP\_EpochNumber} \times 0x100000000 \quad (8)$$

où

PTCP_Epoch est la variable d'aide pour le calcul des secondes

PTCP_EPOCHNumber est le numéro époque

$$\text{PTCP\_Time} = \text{PTCP\_Epoch} + \text{PTCP\_Second} + \text{PTCP\_Nanosecond} \quad (9)$$

où

PTCP_Time est l'heure de PTCP

PTCP_Epoch est la variable d'aide pour le calcul des secondes

PTCP_Second est la valeur en secondes de PTCP

PTCP_Nanosecond

est la valeur en nanosecondes de PTCP

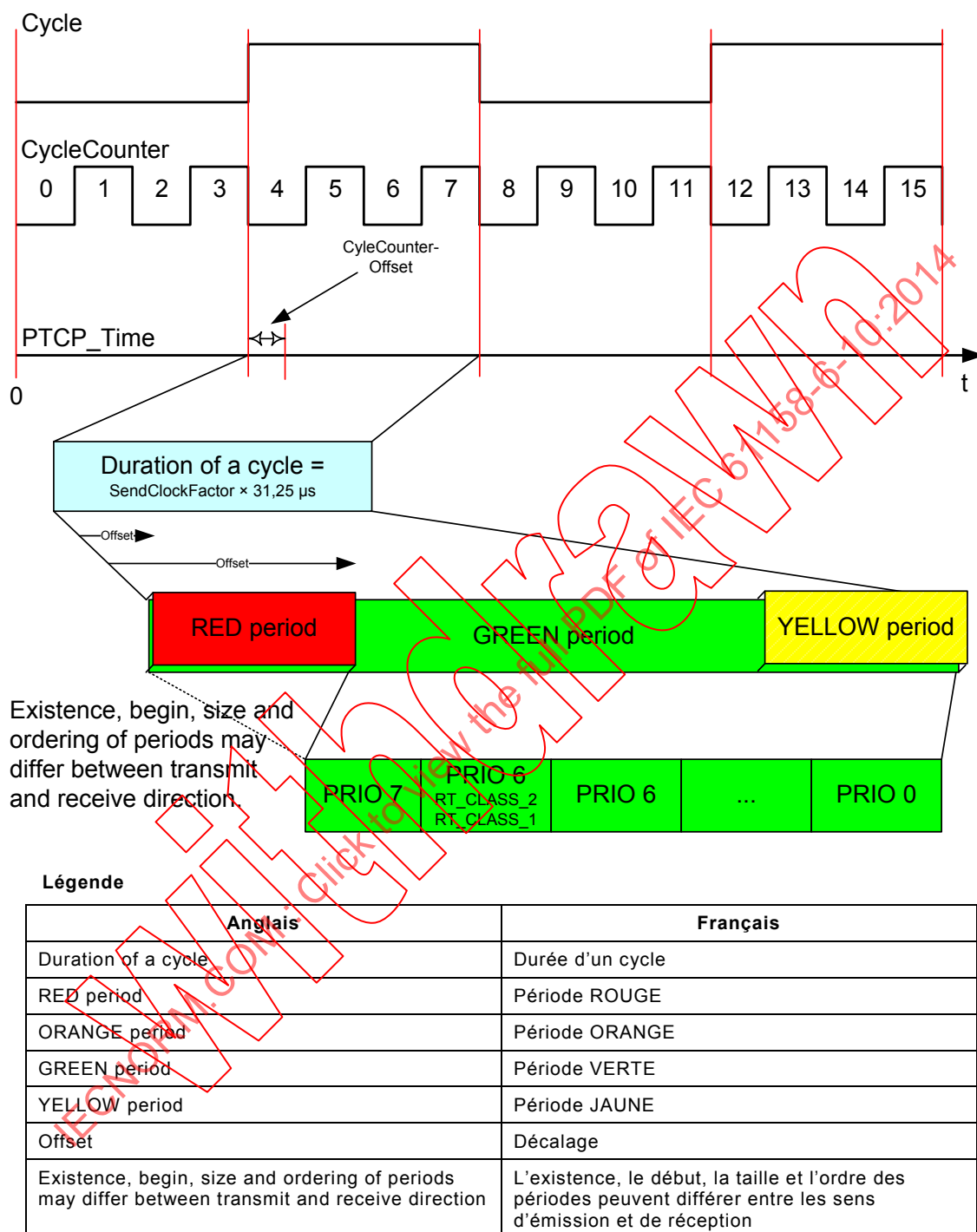


Figure 21 – Correspondance d'échelles de temps entre PTCP_Time et CycleCounter

Pour la synchronisation de CycleCounter présentée à la Figure 21, le calcul doit accepter les valeurs PTCP_Time avec un décalage pour que le cycle commence.

$$\text{Cycle} = \text{PTCP\_Time} \text{ DIV } (\text{SendClockFactor} \times 31,25 \mu\text{s}) \quad (10)$$

où

Cycle

est le cycle

PTCP_Time

est l'heure de PTCP

SendClockFactor est le facteur de l'horloge d'envoi, la longueur d'une phase

$$\text{CycleCounter} = \text{Cycle} \times \text{SendClockFactor} \quad (11)$$

où

CycleCounter est le compteur de cycles
 Cycle est le cycle
 SendClockFactor est le facteur de l'horloge d'envoi, la longueur d'une phase

$$\text{CycleCounterOffset} = \text{PTCP\_Time} \text{ MOD } (\text{SendClockFactor} \times 31,25 \text{ } \mu\text{s}) \quad (12)$$

où

CycleCounterOffset est le décalage du compteur de cycles
 PTCP_Time est l'heure de PTCP
 SendClockFactor est le facteur de l'horloge d'envoi, la longueur d'une phase

NOTE 1 DIV effectue une division entière et donne le quotient.

NOTE 2 MOD effectue une division entière et donne le reste.

4.4.1.4.5 Codage du champ PTCP_CurrentUTCOffset

Ce champ doit être codé en type de données Integer16. La valeur du champ PTCP_CurrentUTCOffset en cours doit être celle de current_utc_offset du jeu de données des propriétés de temps universel de l'horloge qui émet ce message conformément au Tableau 114.

4.4.1.4.6 Codage du champ PTCP_MasterPriority1

PTCP_MasterPriority1 est utilisé dans l'exécution du meilleur algorithme de l'horloge maître. Les valeurs inférieures l'emportent. Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0 – 2: PTCP_MasterPriority1.Priority

Ce champ doit être mis conformément au , au Tableau 116 et au Tableau 117.

Tableau 116 – PTCP_MasterPriority1.Priority for SyncID == 0 et SyncProperties.Role == 2

Valeur (hexadécimale)	Utilisation	Signification
0x00	Réservé	—
0x01	Maître de synchronisation principal	Synchronisation de l'horloge (maître primaire)
0x02	Maître de synchronisation secondaire	Synchronisation de l'horloge (maître secondaire)
0x03 – 0x07	Réservé	—

Tableau 117 – PTCP_MasterPriority1.Priority for SyncID == 0 et SyncProperties.Role == 1

Valeur (hexadécimale)	Utilisation	Signification
0x00	Esclave de synchronisation	Synchronisation de l'horloge (esclave)
0x01 – 0x07	Réservé	—

Bit 3 – 5: PTCP_MasterPriority1.Level

Ce champ doit être mis conformément au Tableau 118.

Tableau 118 – PTCP_MasterPriority1.Level

Valeur (hexadécimale)	Utilisation	Signification
0x00	Niveau 0 (le plus élevé)	Voir Annexe J
0x01	Niveau 1	
0x02 – 0x06	...	
0x07	Niveau 7 (le plus bas)	

Bit 6: PTCP_MasterPriority1.Reserved

Ce champ doit être mis à zéro.

Bit 7: PTCP_MasterPriority1.Active

Ce champ doit être mis à zéro pour PDSyncData.

Ce champ doit être mis à zéro pour PTCP-ANNOUNCE-PDU si le maître de synchronisation n'achemine pas de PTCP-SYNC-PDU et il doit être mis à un si le maître de synchronisation achemine des PTCP-SYNC-PDU.

Par conséquent, ce champ doit être mis à un pour PTCP-SYNC-PDU après que le choix du meilleur maître a été effectué.

4.4.1.4.7 Codage du champ PTCP_MasterPriority2

Ce champ doit être codé en Unsigned8. PTCP_MasterPriority2 est utilisé dans l'exécution du meilleur algorithme de l'horloge maître. Les valeurs inférieures l'emportent. La valeur de PTCP_MasterPriority2 doit pouvoir être configurée conformément au Tableau 119.

Tableau 119 – PTCP_MasterPriority2

Valeur (hexadécimale)	Utilisation	Signification
0x00 – 0xFE	Réservé	—
0xFF	Par défaut	Par défaut

4.4.1.4.8 Codage du champ PTCP_ClockClass

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 120. Pour synchroniser l'horloge de travail, l'échelle de temps arbitraire (ARB) doit être utilisée.

Tableau 120 – PTCP_ClockClass for SyncID == 0 (synchronisation de l'horloge de travail)

Valeur (hexadécimale)	Utilisation	Signification
0x00 – 0x0C	Réservé	—
0x0D	Doit désigner une horloge PTCP comme étant la référence principale synchronisée par rapport à une source temporelle spécifique à l'application. L'échelle de temps distribuée doit être ARBITRARY. Une horloge de classe 0x0D ne doit pas être l'esclave d'une autre horloge du domaine PTCP.	Par défaut
0x0E	Doit désigner une horloge PTCP préalablement conçue comme une horloge de classe 0x0D, mais qui n'a plus la possibilité d'assurer la synchronisation par rapport à une source temporelle spécifique à une application, et qui évolue en mode de rémanence dans le cadre de spécifications de rémanence. L'échelle de temps distribuée doit être ARBITRARY. Une horloge de classe 0x0E ne doit pas être l'esclave d'une autre horloge du domaine PTCP.	—
0x0F – 0xBA	Réservé	—
0xBB	Alternative de dégradation B d'une horloge PTCP de classe 0x07 hors du cadre d'une spécification de rémanence. Une horloge de classe 0xBB peut être l'esclave d'une autre horloge du domaine PTCP.	—
0xBC – 0xC0	Réservé	—
0xC1	Alternative de dégradation B d'une horloge PTCP de classe 0x0E hors du cadre d'une spécification de rémanence. Une horloge de classe 0xC1 peut être l'esclave d'une autre horloge du domaine PTCP.	—
0xC2 – 0xFE	Réservé	—
0xFF	Doit être la classe d'une horloge esclave uniquement.	—

4.4.1.4.9 Codage du champ PTCP_ClockAccuracy

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 121. PTCP_ClockAccuracy indique la précision prévue d'une horloge maîtresse. La valeur de PTCP_ClockAccuracy doit représenter son état en cours.

Tableau 121 – PTCP_ClockAccuracy

Valeur (hexadécimale)	Utilisation	Signification
0x00 – 0x1F	Réservé	—
0x20	L'heure est précise à 25 ns près.	—
0x21	L'heure est précise à 100 ns près.	Valeur par défaut pour synchroniser l'horloge de travail
0x22	L'heure est précise à 250 ns près.	—
0x23	L'heure est précise à 1 µs près.	—
0x24	L'heure est précise à 2,5 µs près.	—
0x25	L'heure est précise à 10 µs près.	—
0x26	L'heure est précise à 25 µs près.	—
0x27	L'heure est précise à 100 µs près.	—
0x28	L'heure est précise à 250 µs près.	—
0x29	L'heure est précise à 1 µs	—
0x30 – 0xFD	Réservé	—
0xFE	Inconnu	—
0xFF	Réservé	—

4.4.1.4.10 Codage du champ PTCP_ClockVariance

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 122. La valeur caractérise la qualité d'une horloge. Chaque horloge maîtresse doit conserver une estimation de sa précision inhérente. Il s'agit de la précision des horodatages inclus dans les trames de synchronisation par le maître de synchronisation PTCP, lorsqu'il n'est pas synchronisé à un autre maître de synchronisation PTCP à l'aide du protocole PTCP.

Tableau 122 – PTCP_ClockVariance

Valeur (hexadécimale)	Utilisation	Signification
0x0000	Réservé	Par défaut
0x0001 – 0xFFFF	Réservé	—

4.4.1.4.11 Codage du champ PTCP_T2PortRxDelay

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 123.

Tableau 123 – PTCP_T2PortRxDelay

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	Délai de réception de port (MDI vers SHIM) en nanosecondes.

4.4.1.4.12 Codage du champ PTCP_T3PortTxDelay

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 124.

Tableau 124 – PTCP_T3PortTxDelay

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	Délai de transmission de port (SHIM vers MDI) en nanosecondes.

4.4.1.4.13 Codage du champ PTCP_PortMACAddress

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ PTCP_PortMACAddress doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

4.4.1.4.14 Codage du champ PTCP_T2TimeStamp

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 125.

Tableau 125 – PTCP_T2TimeStamp

Valeur (hexadécimale)	Signification
0x00000000 – 0xFFFFFFFF	TimeStamp, en nanosecondes

4.4.2 Diagramme d'états de contexte AP

Il n'existe aucun diagramme d'états de contexte AP défini pour ce protocole.

4.4.3 Machines de Protocole de Service FAL

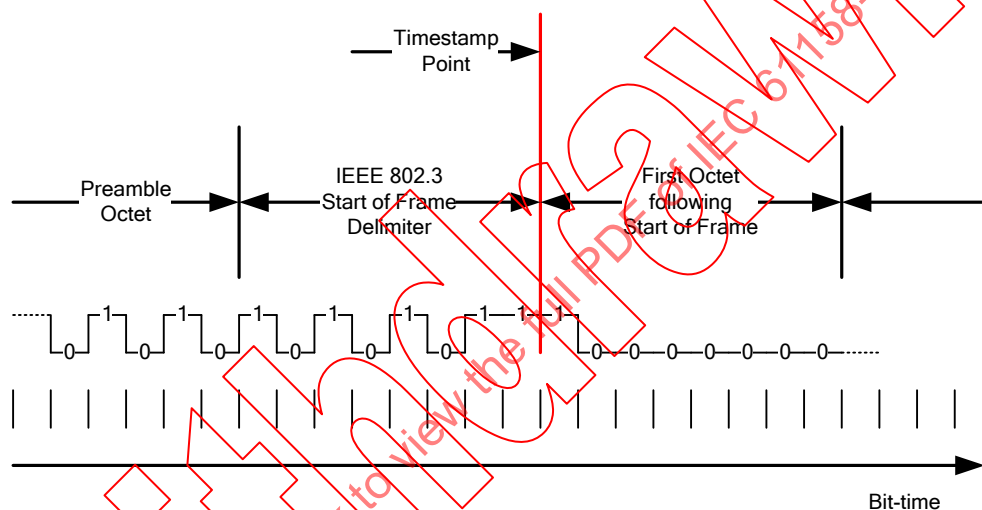
Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.4.4 Machines de protocole de relation entre applications

4.4.4.1 Synchronisation générique avec PTCP

4.4.4.1.1 Horodatage

Le point d'horodatage de message PTCP pour une trame transmise ou reçue doit correspondre au front d'impulsion du premier bit de l'octet qui suit immédiatement l'octet SFD (Délimiteur de début de trame) d'une trame IEEE 802.3 (voir Figure 22).



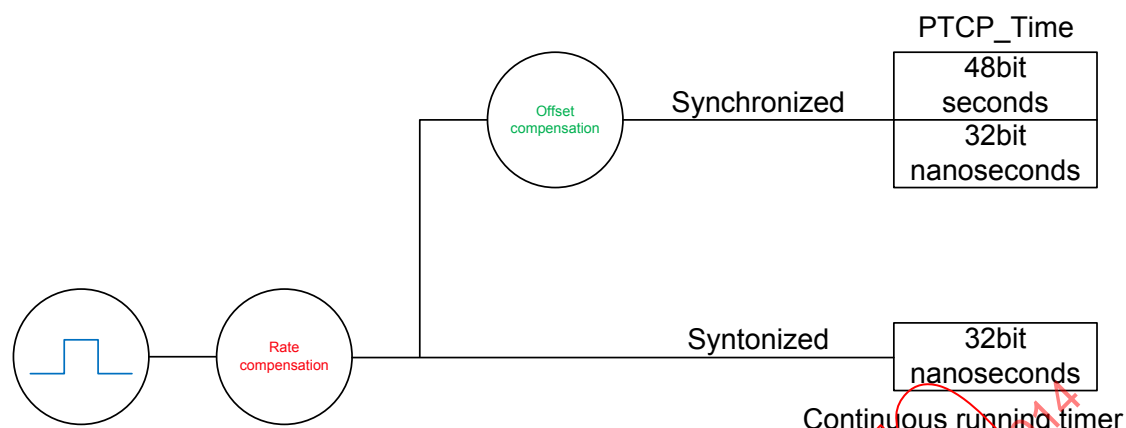
Légende

Anglais	Français
Timestamp Point	Point d'horodatage
Preamble Octet	Octet de préambule
Start of Frame Delimiter	Délimiteur de début de trame
First Octet following Start of Frame	Premier octet après le début de trame
Bit-time	Durée binaire

Figure 22 – Point d'horodatage de message

4.4.4.1.2 Modèle de temporisateur

PTCP utilise le modèle de temporisateur présenté à la Figure 23.



Légende

Anglais	Français
Synchronized	Synchronisé
Syntonized	Syntonisé
Rate compensation	Compensation de débit
Offset compensation	Compensation de décalage
seconds	Secondes
nanoseconds	Nanosecondes
Continuous running timer	Temporisateur en marche en continu

Figure 23 – Modèle de temporisateur

Il est composé de deux temporisateurs. Le premier temporisateur, doté d'une syntonisation d'au moins 32 bits de large et d'une résolution d'une nanoseconde est utilisé pour la mesure du retard de ligne et du retard de pont; le second temporisateur est synchronisé à 80 bits de large avec une résolution d'une nanoseconde et est utilisé pour le PTCP_Time du maître ou de l'esclave.

4.4.4.1.3 Modèle générique

Pour chaque message de synchronisation, de demande de délai ou de réponse de délai reçu et transmis, une unité d'horodatage doit générer un horodatage. La Figure 24 montre les quatre horodatages utilisés pour la synchronisation.

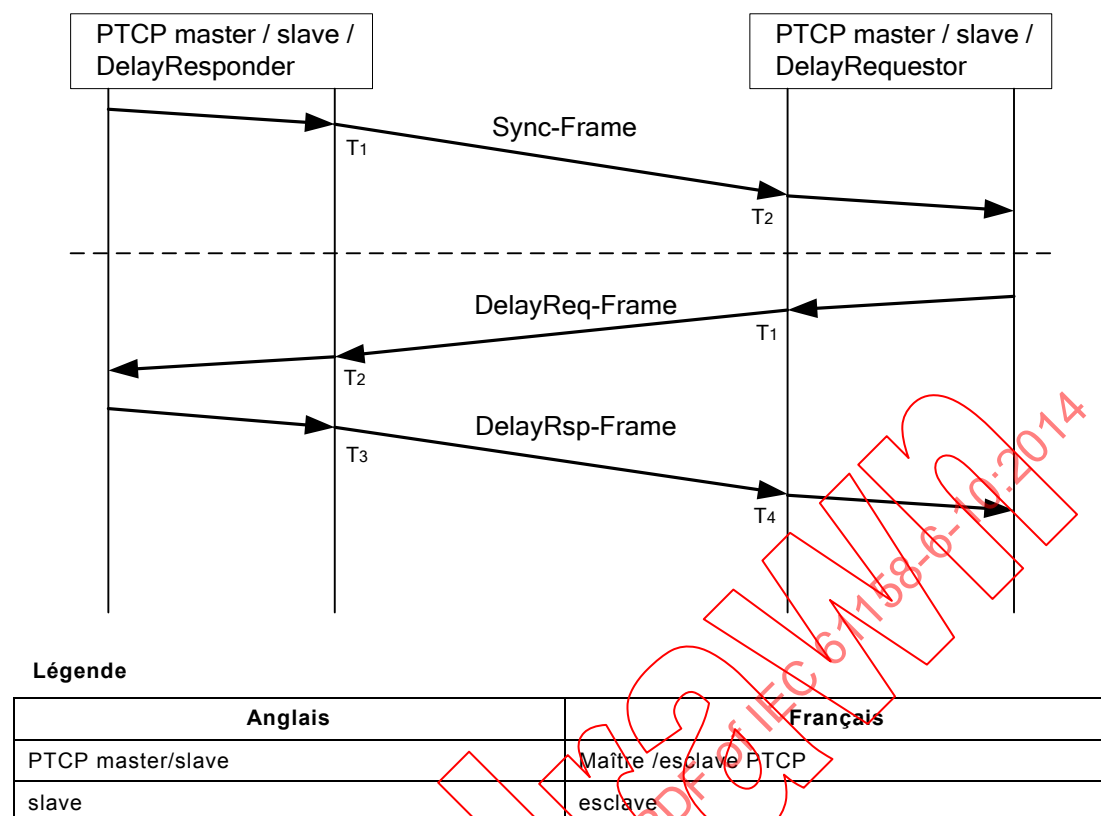


Figure 24 – Quatre horodatages de message

Le modèle générique utilise les formules suivantes.

$$\text{SendTimeStamp} = T_1 \quad (13)$$

$$\text{ReceiptTimeStamp} = T_2 \quad (14)$$

$$\text{SendTimeStamp} = T_3 \quad (15)$$

$$\text{ReceiptTimeStamp} = T_4 \quad (16)$$

où

- T_1 est mesuré par l'horloge locale du port qui envoie une Sync- ou une DelayReq-Frame
- T_2 est mesuré par l'horloge locale du port qui reçoit une Sync- ou une DelayReq-Frame
- T_3 est mesuré par l'horloge locale du port qui envoie une DelayRes-Frame
- T_4 est mesuré par l'horloge locale du port qui reçoit une DelayRes-Frame

4.4.4.1.4 Mesure du retard de ligne et compensation de débit homologue

La mesure du retard de ligne et le calcul de la compensation de débit homologue entre DelayRequestor et DelayResponder sont présentés à la Figure 25. DelayReq-, DelayRes et DelayFuRes-Messages doivent être utilisés pour mesurer le retard et ne doivent pas être propagés. DelayFuRes-Frame est utilisé pour transmettre la valeur de ResDelay au DelayRequestor.

Les DelayResponder qui ne peuvent pas mesurer le retard de ligne mais souhaitent recevoir la Sync-Frame doivent utiliser une DelayRes.req avec une valeur nulle de ResDelay.

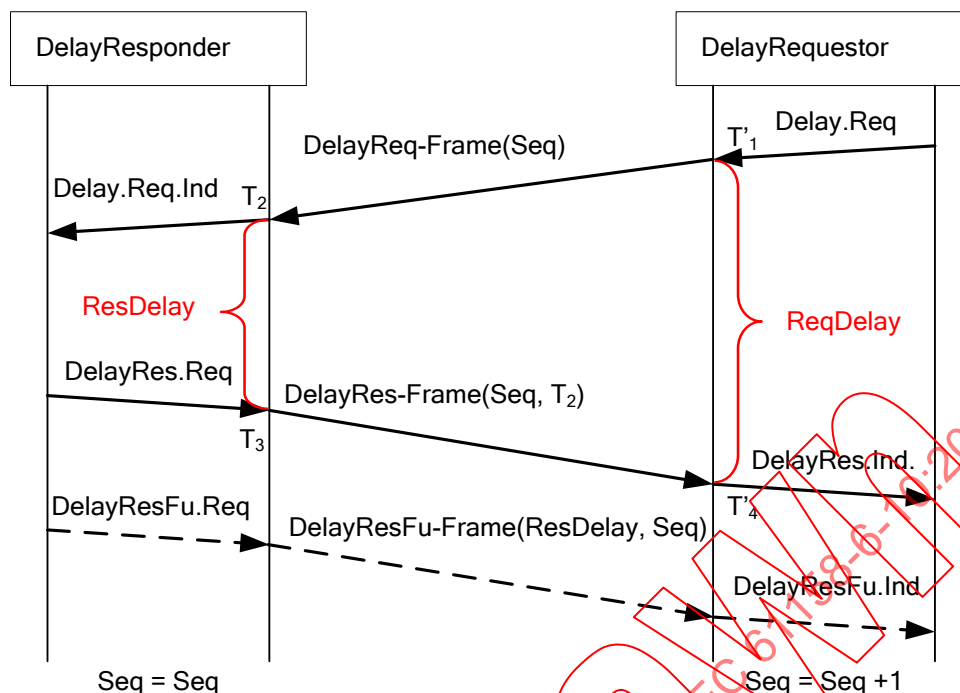


Figure 25 – Protocole de retard de ligne avec suivi

Si une unité d'horodatage est en mesure de calculer le délai de réponse à la volée et d'insérer le temps de réponse dans la trame de réponse de délai, aucun message de suivi n'est nécessaire.

La mesure du retard de ligne sans message DelayFuRes entre DelayRequestor et DelayResponder est décrite à la Figure 26.

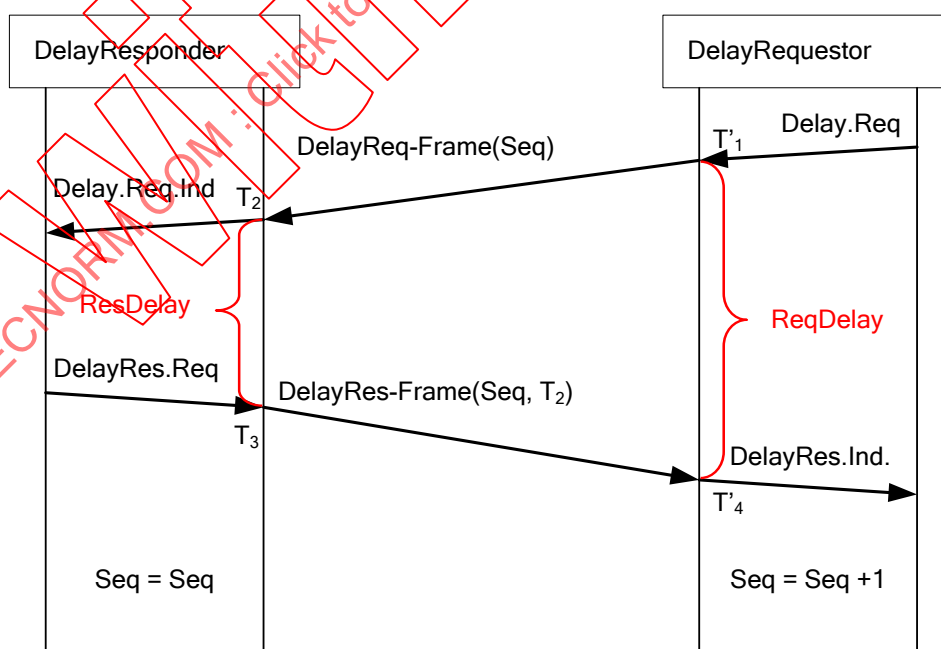


Figure 26 – Protocole de retard de ligne sans suivi

Le calcul du retard de ligne et du facteur de compensation de débit homologue est présenté dans les formules suivantes. Si le DelayRes-PDU contient le champ T_2 , il faut utiliser la formule (19), si ResDelay est zéro, RCF_{peer} doit être réglé à un.

$$\text{ResDelay} = T_3 - T_2 \quad (17)$$

où

ResDelay	est le retard côté répondeur
T_3	est l'horodatage de DelayRes-Frame émettrice comme DelayResponder
T_2	est l'horodatage de DelayReq-Frame réceptrice comme DelayResponder

$$\text{ReqDelay} = T'_4 - T'_1 \quad (18)$$

où

ReqDelay	demandeur de retard
T'_4	est l'horodatage de DelayRes-Frame réceptrice comme DelayRequestor
T'_1	est l'horodatage de DelayReq-Frame émettrice comme DelayRequestor

$$\text{RCF}_{\text{peer}} = \frac{T_2(\text{Seq}) - T_2(\text{Seq} - 1)}{T'_1(\text{Seq}) - T'_1(\text{Seq} - 1)} \quad (19)$$

où

RCF_{peer}	est le facteur de compensation de fréquence homologue
T_2	est l'horodatage de DelayReq-Frame réceptrice comme DelayResponder, il est transmis au DelayRequestor en utilisant le DelayRes-Frame
T'_1	est l'horodatage de DelayReq-Frame émettrice comme DelayRequestor
Seq	est la variable séquence

$$\text{ResDelay}_{\text{peer}} = \frac{\text{ResDelay}}{\text{RCF}_{\text{peer}}} \quad (20)$$

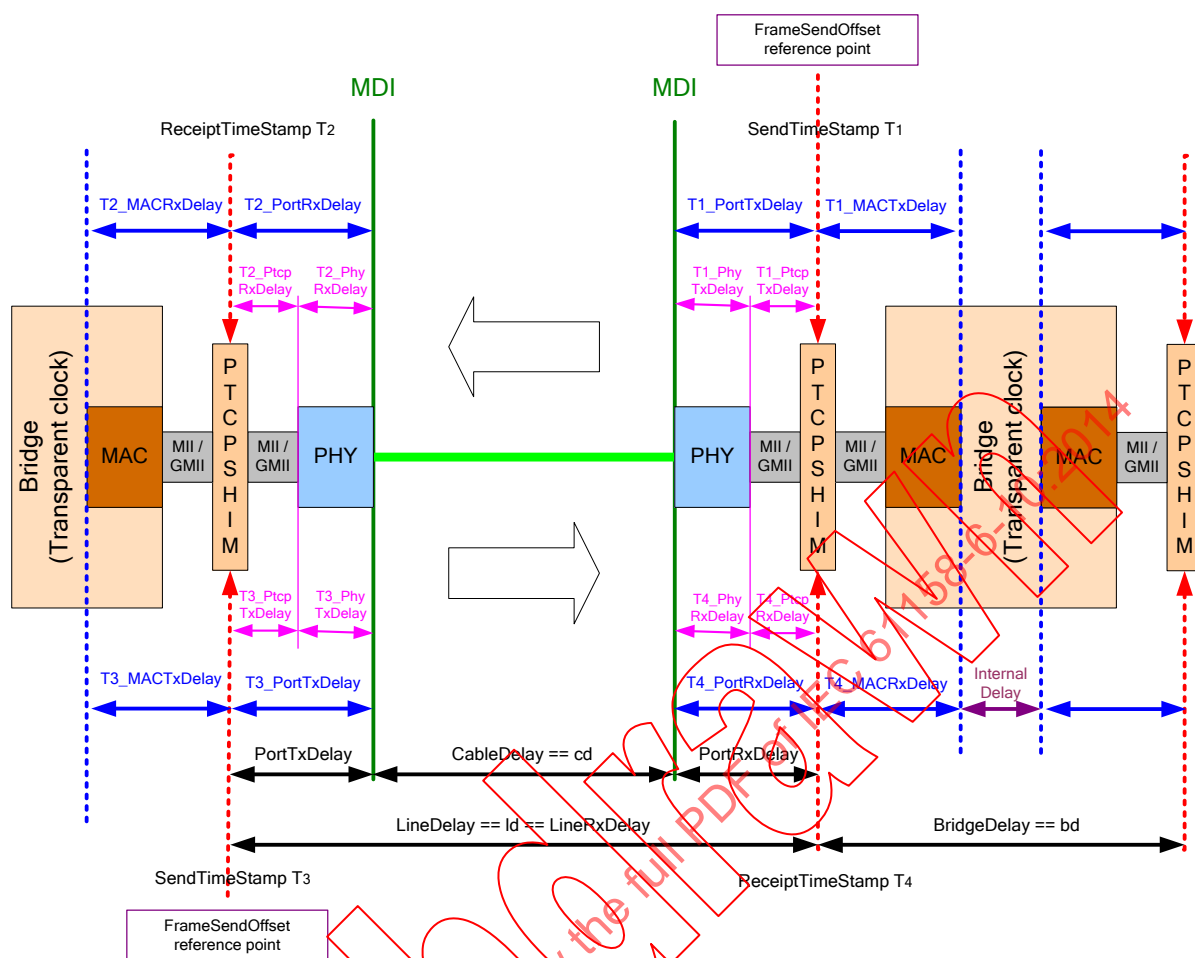
où

$\text{ResDelay}_{\text{peer}}$	est le répondeur de retard à cet homologue
ResDelay	est le répondeur de retard
RCF_{peer}	est le facteur de compensation de fréquence homologue

La mesure du délai doit être répétée en cycles.

4.4.4.1.5 Calcul du retard de ligne

Le retard de ligne entre deux appareils prenant en charge PTCP est déterminé conformément à la Figure 27.



Légende

Anglais	Français
FrameSendOffset reference point	Point de référence FrameSendOffset
Internal Delay	Délai interne
Bridge	Pont
Transparent clock	Horloge transparente

Figure 27 – Mesure du retard de ligne

NOTE 1 L'interface RMII (Reduced Media Independent Interface) synchronise son horloge interne 50 MHz avec l'horloge 25 MHz de PHY. Cela donne lieu à une erreur de quantification de 20 ns. Cette gigue ne peut pas être compensée. L'interface MII (Media Independent Interface) utilise l'horloge PHY sans cette erreur.

NOTE 2 Le type de câble Classe D présente une différence de délai maximal de 45 ns pour 100 m de câble. Il en résulte une erreur de décalage de 45 ns qui ne peut pas être compensée. Le choix d'un meilleur câble réduit cette erreur.

Un délai de réception/transmission de port (par exemple; T1_PortTxDelay) contient le retard de ligne du composant PHY utilisé et l'écart avec l'horodatage de référence.

$$T1\_PortTxDelay = T1\_PhyTxDelay + T1\_PtcpTxDelay \quad (21)$$

où

T1_PortTxDelay est le délai d'émission de port
T1_PhyTxDelay est le retard du composant PHY utilisé
T1_PtcpTxDelay est l'écart avec l'horodatage de référence

$$T3\_PortTxDelay = T3\_PhyTxDelay + T3\_PtcpTxDelay \quad (22)$$

où

$T3\_PortTxDelay$	est le délai d'émission de port
$T3\_PhyTxDelay$	est le retard du composant PHY utilisé
$T3\_PtcpTxDelay$	est l'écart avec l'horodatage de référence

$$T2\_PortRxDelay = T2\_PhyRxDelay + T2\_PtcpRxDelay \quad (23)$$

où

$T2\_PortRxDelay$	est le délai de réception de port
$T2\_PhyRxDelay$	est le retard du composant PHY utilisé
$T2\_PtcpRxDelay$	est l'écart avec l'horodatage de référence

$$T4\_PortRxDelay = T4\_PhyRxDelay + T4\_PtcpRxDelay \quad (24)$$

où

$T4\_PortRxDelay$	est le délai de réception de port
$T4\_PhyRxDelay$	est le retard du composant PHY utilisé
$T4\_PtcpRxDelay$	est l'écart avec l'horodatage de référence

$$CableDelay = (ReqDelay - ResDelay_{peer} - T1\_PortTxDelay - T2\_PortRxDelay - T3\_PortTxDelay - T4\_PortRxDelay) / 2 \quad (25)$$

où

$CableDelay$	est le retard du câble
$ReqDelay$	est le demandeur du retard
$ResDelay_{peer}$	est le répondeur de retard
$T1\_PortTxDelay$	est le délai d'émission de port
$T2\_PortRxDelay$	est le délai de réception de port
$T3\_PortTxDelay$	est le délai d'émission de port
$T4\_PortRxDelay$	est le délai de réception de port

$$LineDelay = CableDelay + T3\_PortTxDelay + T4\_PortRxDelay \quad (26)$$

où

$LineDelay$	est le retard de réception/transmission du port
$CableDelay$	est le retard du câble
$T3\_PortTxDelay$	est le retard de réception/transmission du port
$T4\_PortRxDelay$	est le retard de réception/transmission du port

$$LineDelay_{SyncMaster} = LineDelay \times RCF_{SyncMaster} \quad (27)$$

où

$LineDelay_{SyncMaster}$	est le temps de compensation requis pour transférer une trame
$LineDelay$	est la valeur mesurée entre deux homologues
$RCF_{SyncMaster}$	est le facteur entre la fréquence de SyncMaster et la fréquence de SyncSlave

NOTE 3 La formule (27) est seulement informative si la valeur LineDelay est mesurée avec le temporisateur synchronisé.

Une DelayRes.req avec une valeur nulle de ResDelay indique que CableDelay et LineDelay ne sont pas mesurables, mais que l'homologue de réponse souhaite recevoir le Sync-Frame.

LineSyncDelay est le retard de ligne de Sync-Frame. Compte tenu de l'asymétrie du chemin de communication concerné et de la résolution des horodatages, il est fondamentalement impossible de déterminer exactement le retard de ligne.

Une horloge PTCP peut détecter des voisins non PTCP en vérifiant le retard de ligne entre les nœuds. Un délai de commutation se trouvant dans la plage de plusieurs $\mu s > 2$), le délai entre deux nœuds connectés avec un câble à paire torsadée de 100 m (100 Base TX) est d'environ 500 ns.

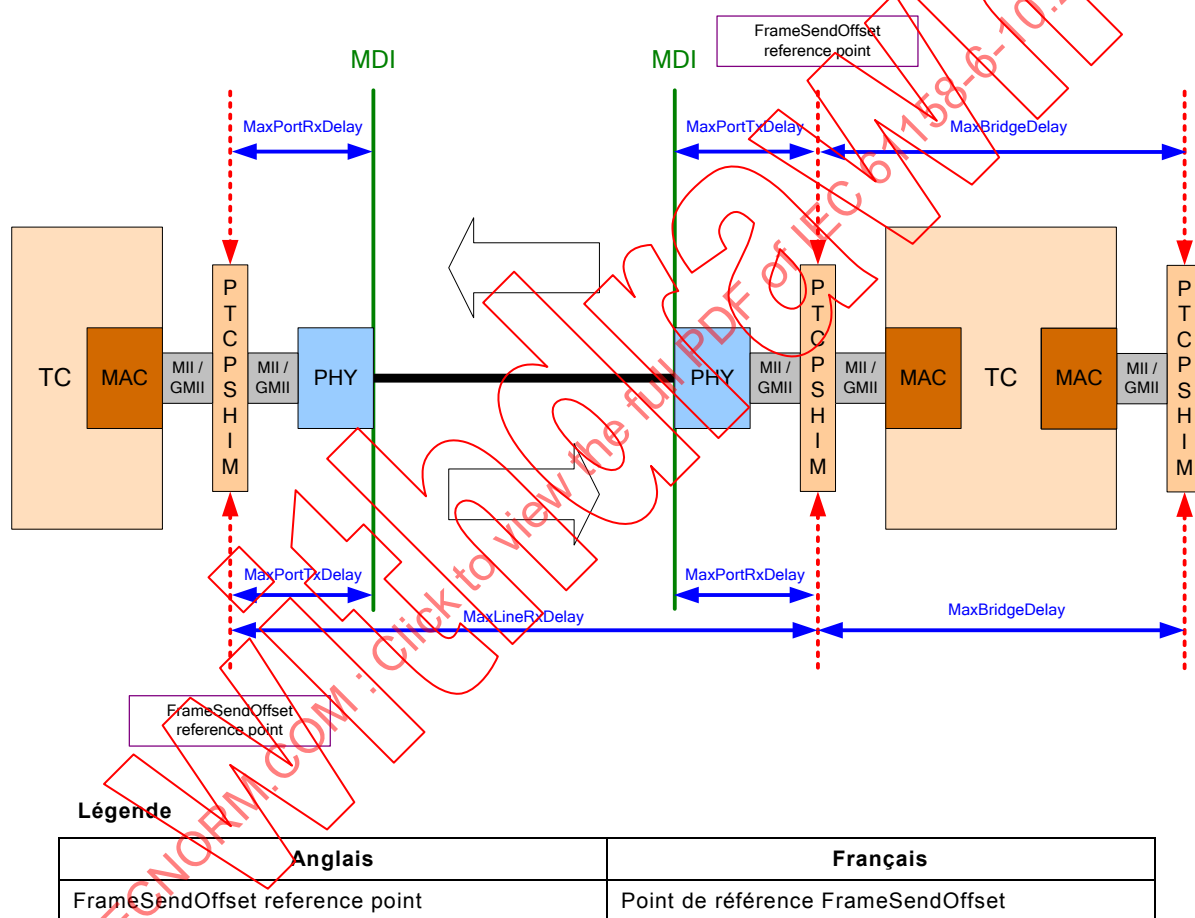


Figure 28 – Paramètre de modèle pour l'utilisation GSDML

La Figure 28 montre le paramètre de modèle pour l'utilisation GSDML. Les paramètres MaxBridgeDelay, MaxPortTxDelay et MaxPortRxDelay offrent des outils techniques comme base de calcul de RT_CLASS_3.

La valeur MaxBridgeDelay à utiliser dépend du mode de pontage comme transfert normal, transfert rapide et transfert concaténé DFP.

4.4.4.1.6 Transfert de frame de synchronisation

Un appareil qui transfère des messages de synchronisation PTCP mesure le délai de pontage (voir Figure 29) et ajoute cette durée au champ de délai du message de synchronisation ou de suivi.

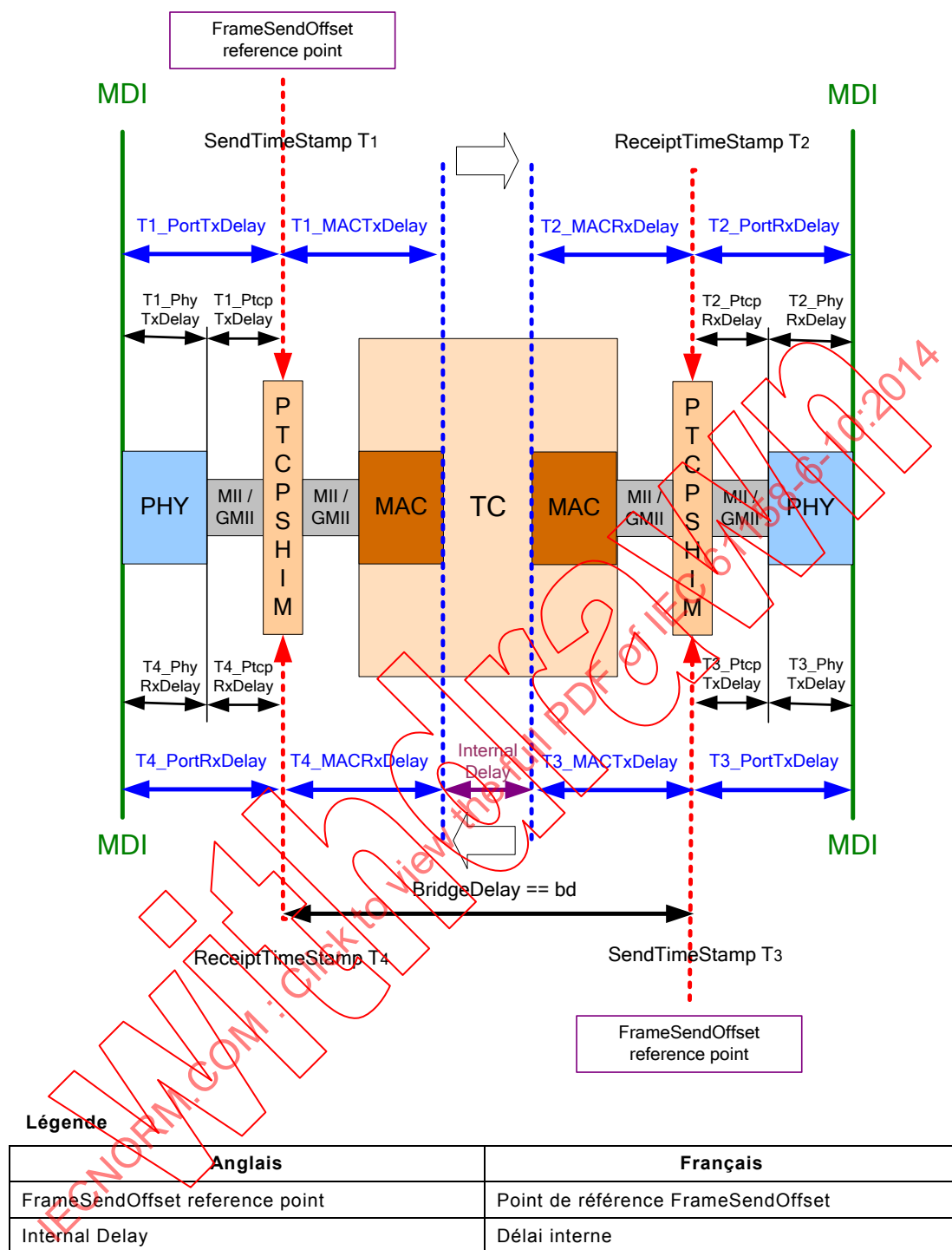


Figure 29 – Mesure du délai de pontage

Le calcul du délai de pontage est présenté dans les formules suivantes.

$$\text{BridgeDelay} = T_3 - T_4 \quad (28)$$

où

BridgeDelay

est le temps requis pour transférer une trame

T3

est le point dans le temps où le premier bit du SD d'une trame est transmis

T4

est le point dans le temps où le premier bit du SD d'une trame est reçu

$$\text{BridgeDelay}_{\text{SyncMaster}} = \text{BridgeDelay} \times \text{RCF}_{\text{SyncMaster}} \quad (29)$$

où

$\text{BridgeDelay}_{\text{SyncMaster}}$ est le temps de compensation requis pour transférer une trame

BridgeDelay est le temps requis pour transférer une trame

$\text{RCF}_{\text{SyncMaster}}$ est le facteur entre la fréquence de SyncMaster et la fréquence de SyncSlave

NOTE La formule (29) est seulement informative si la valeur LineDelay est mesurée avec le temporisateur synchronisé.

$$\text{RCF}_{\text{SyncMaster}} = \Delta t_{\text{SyncMaster}} / \Delta t_{\text{SyncSlave}} \quad (30)$$

où

$\text{RCF}_{\text{SyncMaster}}$ est le ratio entre l'horloge de SyncMaster et SyncSlave calculé par SyncSlave

$\Delta t_{\text{SyncMaster}}$ est un intervalle temporel d'au moins 200 ms à partir du point de vue de SyncMaster

$\Delta t_{\text{SyncSlave}}$ est un intervalle temporel mesuré au niveau de SyncSlave pour le $\Delta t_{\text{SyncMaster}}$ donné

Le principe de synchronisation grâce à une séquence d'appareils peut être consulté à la Figure 30. Le nœud qui émet les trames de temps pour la synchronisation dans le réseau est appelé "maître PTCP". Tous les autres nœuds qui reçoivent et/ou transmettent ces trames sont appelés "esclaves PTCP". Un maître PTCP utilise une référence à un système horaire local ou une source de temps globale.

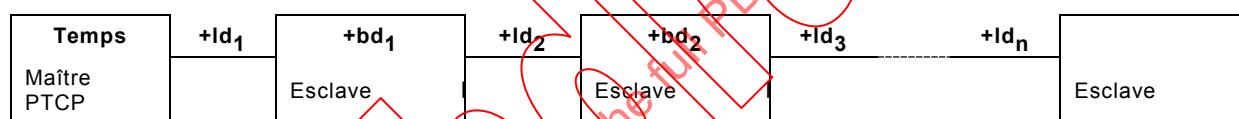


Figure 30 – Accumulation de retard

Les trames de synchronisation sont transmises sous la forme de trames P2P. Chaque appareil doit ajuster la trame de synchronisation

- en ajoutant son délai de pontage bd_x (la mesure du délai de pontage est présentée à la Figure 29),
- en ajoutant le retard de ligne ld_x (la mesure du retard de ligne est présentée à la Figure 27),

au champ de délai de la trame de synchronisation lors de sa transmission. Chaque esclave PTCP connaît le délai de propagation de la trame de synchronisation. Lorsque l'heure d'arrivée T_2 est connue, l'écart par rapport au maître PTCP peut également être déterminé.

4.4.4.1.7 Compensation de vitesse

4.4.4.1.7.1 Délai de pontage

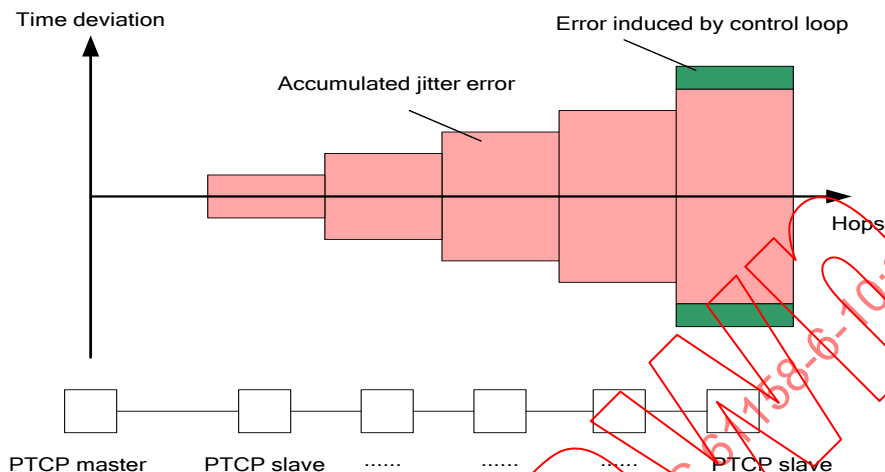
Il convient que le retard de pont soit mesuré par un temporisateur synchronisé / compensé en fréquence par rapport à SyncMaster pour obtenir une synchronisation de haute précision.

4.4.4.1.7.2 Mesure du retard de ligne

Il convient que le retard de ligne soit mesuré par un temporisateur synchronisé / compensé en fréquence par rapport à SyncMaster pour obtenir une synchronisation de haute précision.

4.4.4.1.8 Mesure de l'écart de temps

Pour mesurer l'écart (voir Figure 31), une signalisation matérielle est nécessaire. Un signal doit être généré pour chaque PTCP_SyncID pris en charge. Il peut être uniquement obtenu dans un laboratoire d'essai.

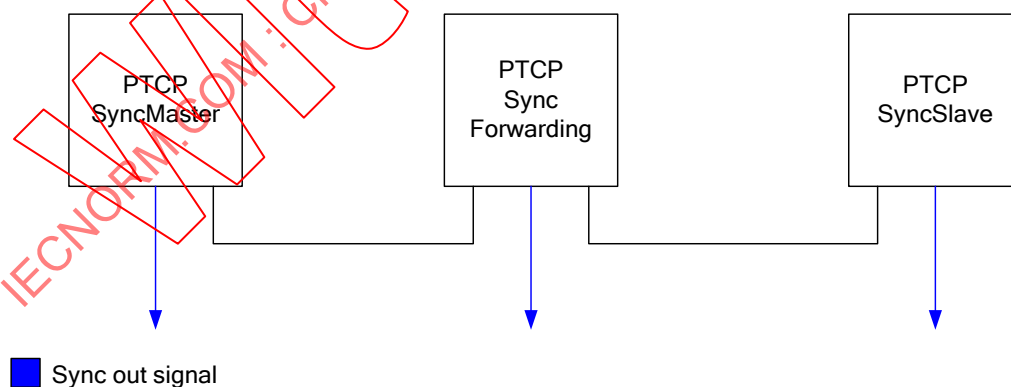


Légende

Anglais	Français
Time deviation	Écart de temps
Accumulated jitter error	Erreur de gigue accumulée
Error induced by control loop	Erreur induite par la boucle de commande
Hops	Sauts (d'un appareil à un autre)
PTCP master	Maître PTCP
PTCP slave	Esclave PTCP

Figure 31 – Cas le plus défavorable pour l'écart de temps de synchronisation cumulé

La Figure 32 et la Figure 33 présentent la mesure de l'écart dans son principe.



Légende

Anglais	Français
Sync out signal	Signal de synchronisation
PTCP Sync Forwarding	Transfert de synchronisation PTCP
PTCP SyncMaster	Maître de synchronisation PTCP
PTCP SyncSlave	Esclave de synchronisation PTCP

Figure 32 – Schéma de la mesure de l'écart

Chaque cycle présenté à la Figure 21 doit être signalé pour la synchronisation d'horloge. Chaque seconde doit être signalée pour la synchronisation temporelle.

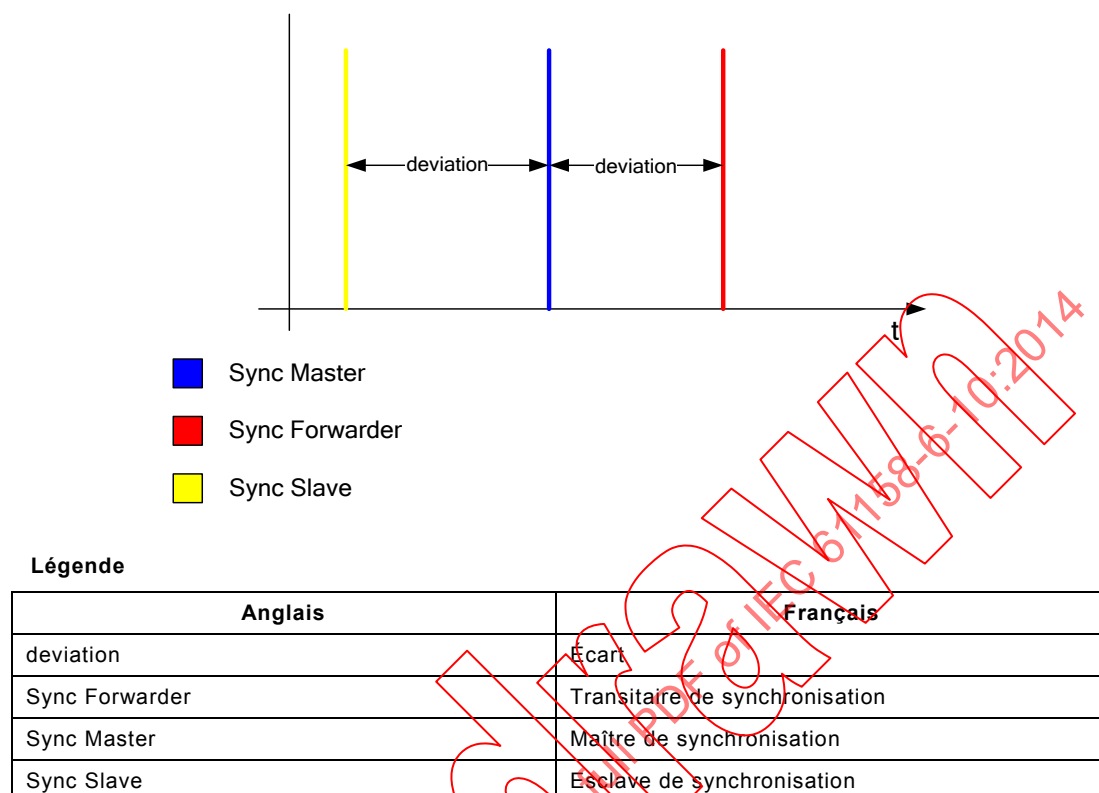


Figure 33 – Mesure de l'écart

4.4.4.1.9 Protocole de synchronisation

4.4.4.1.9.1 Généralités

Les maîtres PTCP envoient périodiquement des trames de synchronisation. La trame de synchronisation possède une heure d'envoi d'origine (T_{ORG}).

Un maître PTCP

- met en place le champ PTCPTime de la trame de synchronisation sur T_{ORG} , et
- met les champs PTCP_Delay de la trame de synchronisation sur son temps de prise en charge interne; c'est-à-dire le temps écoulé entre T_{ORG} et l'envoi de la trame de synchronisation.

Un esclave de PTCP ajoute les termes suivants au contenu du champ correspondant PTCP_Delay dans le PTCP-Header-Sync:

- le retard de ligne (T_{LD}) du port où la trame de synchronisation a été reçue, et
- le délai de pontage (T_{BD}); c'est-à-dire le temps pour transférer la trame de synchronisation.

Cas particulier: Si le maître PTCP ou certains esclaves PTCP ne peuvent pas ajouter directement les délais de retard dans les champs PTCP_Delay de la trame de synchronisation, ils doivent générer une trame dite de suivi (FÜ) et ajouter son délai de pontage au champ PTCP_Delay. Le retard de ligne est toujours ajouté aux champs PTCP_Delay de la trame de synchronisation.

Tous les délais sont toujours liés au point d'horodatage de mesure, voir Figure 22 pour plus de détails.

Chaque esclave PTCP compare son horloge locale avec la somme de PTCPTIME, PTCP_Delay en dehors de la trame de synchronisation et (le cas échéant) PTCP_Delay en dehors de la trame de suivi. L'écart entre T_{Slave} et T_{Master} peut alors être déterminé. L'esclave PTCP suit l'heure du maître PTCP.

4.4.4.1.9.2 Synchronisation des Maîtres PTCP avec Trame de Synchronisation

Les maîtres PTCP envoient périodiquement des trames de synchronisation. Le schéma de synchronisation décrit à la Figure 34 nécessite le transfert de l'heure d'envoi exacte de la trame de synchronisation comme valeur initiale du délai.

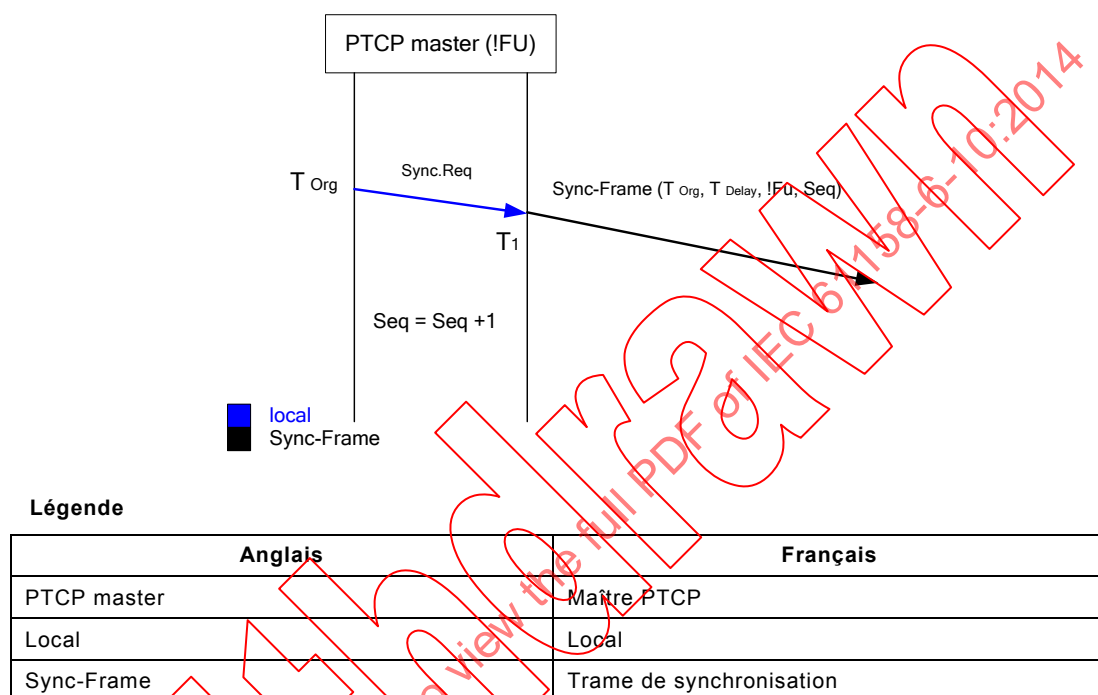


Figure 34 – Maître PTCP envoyant une Sync-Frame (trame de synchronisation) sans Follow Up-Frame (trame de suivi)

$$T_{Delay} = T_1 - T_{Org} \quad (31)$$

où

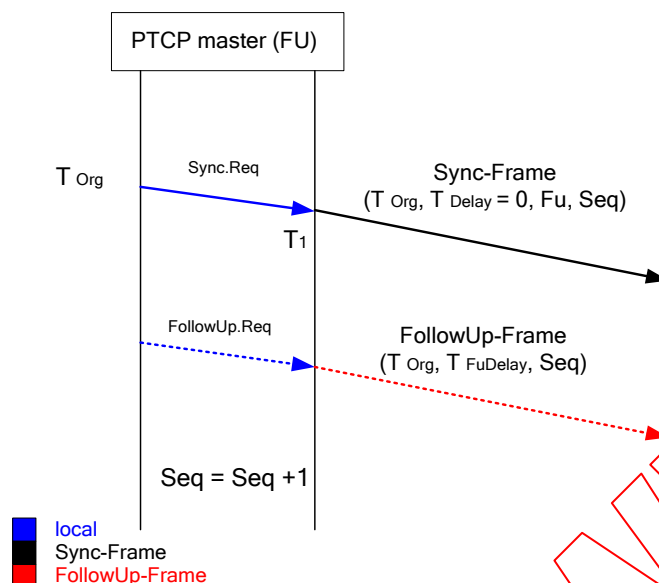
T_{Delay} est l'heure de prise en charge de la trame de synchronisation au sein du maître PTCP; il s'agit de la valeur du champ PTCP_Delay au sein de la trame de synchronisation envoyée par le maître PTCP Master

T_1 est l'horodatage d'envoi de la trame de synchronisation par le maître PTCP

T_{Org} est l'heure d'origine de l'horloge du maître PTCP

4.4.4.1.9.3 Synchronisation des Maîtres PTCP avec Trame de synchronisation et Trame de suivi

La Figure 35 présente un maître PTCP qui ne peut pas modifier de trame avant transmission. Un maître PTCP qui ne peut pas saisir l'heure de prise en charge interne directement dans la trame de synchronisation utilisera une trame de suivi.



Légende

Anglais	Français
PTCP master	Maître PTCP
PTCP slave	Esclave PTCP
Sync-Frame	Trame de synchronisation
FollowUp-Frame	Trame de suivi

Figure 35 – Maître PTCP envoyant une Sync-Frame (trame de synchronisation) avec FollowUp-Frame (trame de suivi)

Le champ "delay" dans la trame de synchronisation (Sync Frame) doit être mis à 0 par le maître PTCP. Le délai ($T_{FuDelay}$) dans la trame de suivi associée contient la valeur initiale du délai de la trame de synchronisation.

$$T_{FuDelay} = T_1 - T_{Org} \quad (32)$$

où

$T_{FuDelay}$ est l'heure de prise en charge de la trame de synchronisation au sein du maître PTCP; il s'agit de la valeur du champ PTCP_Delay au sein de la trame de suivi envoyée par le Maître PTCP.

T_{Org} est l'heure d'origine de l'horloge du maître PTCP

T_1 est l'horodatage d'envoi de la trame de synchronisation par le maître PTCP

4.4.4.1.9.4 Synchronisation de l'esclave PTCP (!FU)

Un esclave PTCP présenté à la Figure 36 peut modifier une trame avant la transmission (nœud !FU) et ajouter son délai de pontage interne et le retard de ligne à la valeur de délai de la trame de synchronisation au fur et à mesure de la transmission.

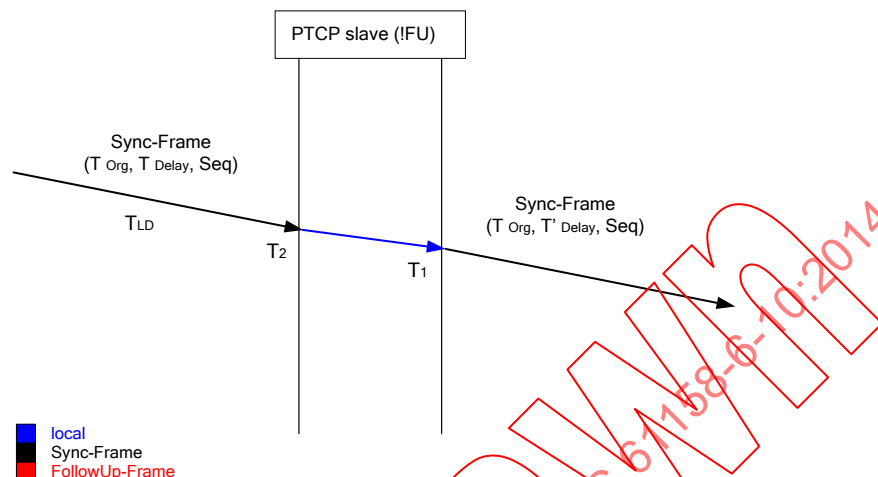


Figure 36 – Esclave de synchronisation !FU transférant une Sync-Frame (trame de synchronisation)

$$T'_{\text{Delay}} = T_{\text{Delay}} + T_{\text{LD}} + (T_1 - T_2) \quad (33)$$

où

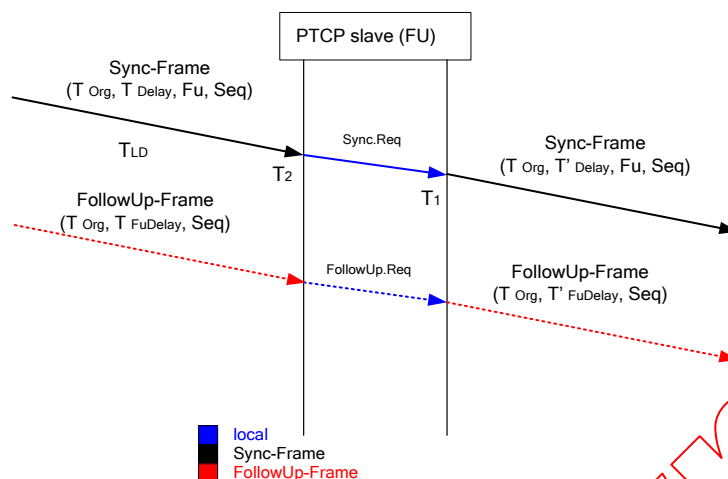
T'_{Delay}	est la valeur du champ PTCP_Delay au sein de la trame de synchronisation envoyée par l'esclave PTCP
T_{Delay}	est le délai reçu par l'esclave PTCP
T_{LD}	est le retard de ligne du port destinataire
T_1	est l'horodatage d'envoi de la trame de synchronisation par l'esclave PTCP
T_2	est l'horodatage de réception de la trame de synchronisation par l'esclave PTCP

Si la trame de synchronisation comporte une trame de suivi, elle est transférée sans modification par un nœud !FU.

4.4.4.1.9.5 Synchronisation esclave PTCP (FU)

Certaines mises en œuvre de l'IEEE 802.3 ne permettent pas la modification d'une trame lors de la transmission.

Par exemple, l'esclave PTCP (Nœud FU) ajoute toujours le retard de ligne à la valeur du retard dans la trame de synchronisation, mais ajoute son délai de pontage à la valeur du délai de la trame de suivi.



Légende

Anglais	Français
PTCP slave	Esclave PTCP
FollowUp-Frame	Trame de suivi
local	Local
Sync-Frame	Trame de synchronisation

Figure 37 – Esclave de synchronisation FU transférant une Sync-Frame (trame de synchronisation) et une FollowUp-Frame (trame de suivi)

$$T'_{\text{Delay}} = T_{\text{Delay}} + T_{\text{LD}} \quad (34)$$

où

T'_{Delay} est la valeur du champ PTCP_Delay au sein de la trame de synchronisation envoyée par l'esclave PTCP

T_{Delay} est le délai au sein de la trame de synchronisation reçue par l'esclave PTCP

T_{LD} est le délai du port destinataire

$$T'_{\text{FuDelay}} = T_{\text{FuDelay}} + (T_1 - T_2) \quad (35)$$

où

T'_{FuDelay} est la valeur du champ PTCP_Delay au sein de la trame de suivi envoyée par l'esclave PTCP

T_{FuDelay} est le délai au sein de la trame de suivi reçue par l'esclave PTCP

T_{LD} est le délai du port destinataire

T_1 est l'horodatage d'envoi de la trame de synchronisation par l'esclave PTCP

T_2 est l'horodatage de réception de la trame de synchronisation par l'esclave PTCP

Si la trame de suivi existe, elle est générée par un esclave PTCP de FU. Ce principe est illustré à la Figure 38.

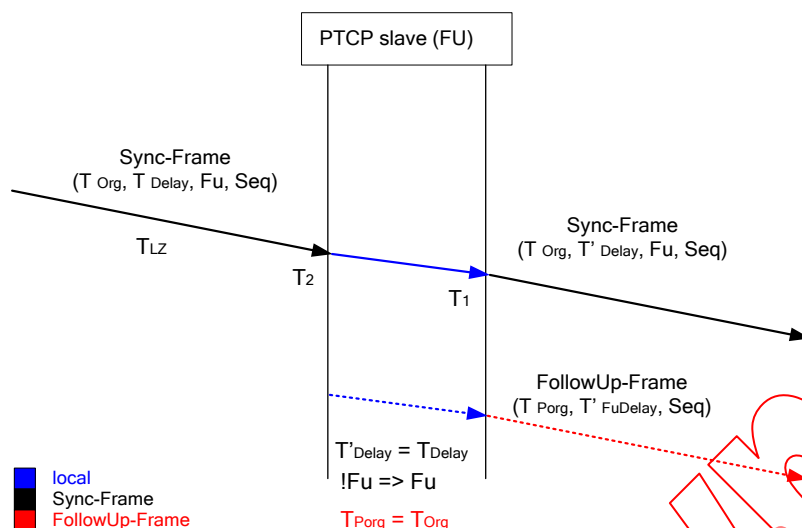


Figure 38 – Esclave de synchronisation FU transférant une Sync-Frame et générant une FollowUp-Frame (trame de suivi)

$$T'_{\text{Delay}} = T_{\text{Delay}} + T_{\text{LD}} \quad (36)$$

où

T'_{Delay} est la valeur du champ PTCP_Delay au sein de la trame de synchronisation envoyée par l'esclave PTCP

T_{Delay} est le délai au sein de la trame de synchronisation reçue par l'esclave PTCP

T_{LD} est le délai du port destinataire

$$T'_{\text{FuDelay}} = (T_1 - T_2) \quad (37)$$

où

T'_{FuDelay} est la valeur du champ PTCP_Delay au sein de la trame de suivi envoyée par l'esclave PTCP

T_{LD} est le délai du port destinataire

T_1 est l'horodatage d'envoi de la trame de synchronisation par l'esclave PTCP

T_2 est l'horodatage de réception de la trame de synchronisation par l'esclave PTCP

4.4.4.1.10 Reprise après erreur

Une interruption de liaison supprime le retard de ligne. Une erreur de mesure du retard de ligne interrompt le transfert de tous les messages de synchronisation.

Pour traiter les permutations redondantes en cas de chemins redondants, une trame de synchronisation doit être transférée uniquement si la différence de numéro de séquence est positive par rapport au précédent message transféré. Cela évite les doublons qui peuvent corrompre la séquence de suivi.

L'utilisation d'un autre chemin est possible, tant qu'une mesure de retard de ligne valide est disponible.

Il existe 2 types de temporisation:

- Mesure du délai:
Durée admise entre la demande et la réponse de délai

- Surveillance de synchronisation:
Durée admise entre deux trames de synchronisation consécutives

La temporisation de la mesure du délai d'attente doit être aussi élevée que celle de la base de temps et calculée indépendamment à chaque mise à jour du retard de ligne.

En cas de limitations de ressource, la taille de la base de données principale peut être limitée. Deux maîtres doivent être possibles à tout moment. Cela est possible, car l'algorithme BMA annule les activités d'un maître de synchronisation.

Pour certaines technologies, les valeurs de retard de ligne peuvent être limitées (la valeur de retard de 100 Base TX est inférieure à 1 μ s, par exemple). L'erreur doit être signalée, mais la réaction n'entre pas dans le domaine d'application de la présente spécification.

4.4.4.2 Mesure du retard de ligne

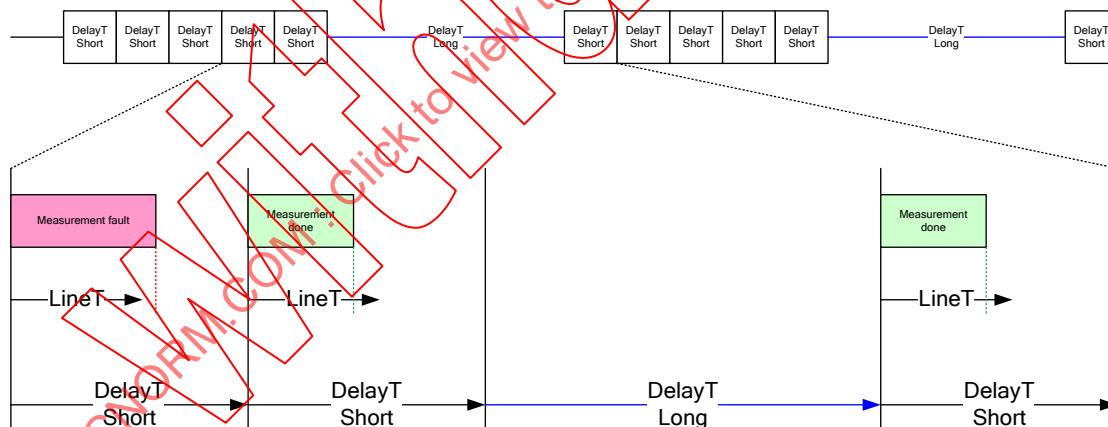
4.4.4.2.1 Machine de Protocole de Demande de Retard de Ligne

4.4.4.2.1.1 Généralités

4.4.4.2.1.1.1 Généralités

La Figure 39 montre que le retard de ligne est surveillé par les temporisateurs LineT et DelayT. A chaque fois que le DelayT expire, une mesure du retard de ligne est lancée. Si la LineT expire avant que la mesure du retard a été effectuée, le nombre d'erreurs augmente.

Si le cycle de mesures actuel (5 fois) est terminé, une coupure de DelayT (long) est effectuée.



Key:
 DelayT (Short) is the delay between two measurements
 DelayT (Long) is the delay between measurement bursts
 LineT is the monitoring time for one measurement

Légende

Anglais	Français
Key	Légende
DelayT (Short) is the delay between two measurements	DelayT (Court) est le délai entre deux mesures
DelayT (Long) is the delay between measurement bursts	DelayT (Long) est le délai entre deux déclenchements de mesure
LineT is the monitoring time for one measurement	Line T est le temps de surveillance d'une mesure

Anglais	Français
Measurement fault	Erreur de mesure
Measurement done	Mesure effectuée

Figure 39 – Principe de surveillance de la mesure de retard de ligne

LineT peut être remplacée par la vérification de la valeur ResDelay value en combinaison avec DelayT(Court).

4.4.4.2.1.2 Définitions de primitives

4.4.4.2.1.2.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Demande de Retard de Ligne (DELAY_REQ), sont décrites dans la définition de service et présentées dans le Tableau 126.

Tableau 126 – Primitives distantes émises ou reçues par DELAY_REQ

Primitive	Source	Destination	Paramètres associés	Fonctions
MAUType_Change_ind	IEEE802.3	DELAY_REQ	PortID, MAU_Type, LINK_Status	—
LMPM_P_Data.ind	LMPM	DELAY_REQ	CREP, S_Port, Tstamp, DA, SA, A_SDU	La LMPM indique une réponse de retard ou une réponse de suivi de retard reçue.
LMPM_P_Data.cnf ()	LMPM	DELAY_REQ	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.req	DELAY_REQ	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	La DELAY_REQ émet une demande de retard.

4.4.4.2.1.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DELAY_REQ sont décrites dans la définition de service et présentées dans le Tableau 127.

Tableau 127 – Primitives locales émises ou reçues par DELAY_REQ

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.4.4.2.1.3 Schéma de transition d'états

Le schéma de transition d'états du DELAY_REQ est présenté à la Figure 40.

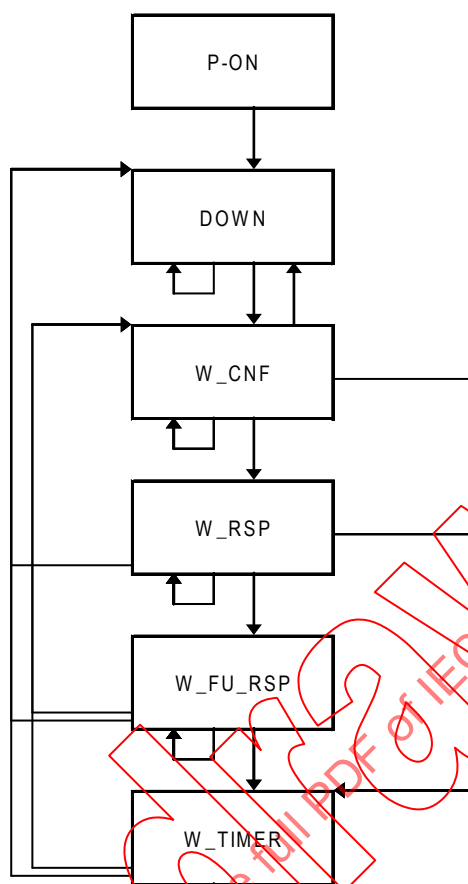


Figure 40 – Schéma de transition d'états de DELAY_REQ

Etats du DELAY_REQ

P-ON	Initialisation des données locales.
DOWN	La liaison du port est interrompue. La mesure du retard de ligne doit être initiée après l'établissement de liaison en envoyant une trame de demande de retard.
W_CNF	Attente de confirmation de la demande de retard et enregistrement de l'horodatage de transmission du message de demande de retard.
W_RSP	Attente de la trame de réponse de retard provenant de DelayResponder.
W_FU_RSP	Attente de la trame de suivi de réponse de délai et calcul du retard de ligne.
W_TIMER	Attente de la temporisation pour répéter la mesure du retard de ligne.

4.4.4.2.1.4 Description du diagramme d'états

La mesure du retard de ligne doit être initiée par chaque port. Il convient de la répéter toutes les 8 s. Le diagramme d'états qui envoie une trame de demande de délai est appelé DelayRequestor.

4.4.4.2.1.5 Table d'états DELAY_REQ

Le Tableau 128 contient la table d'états utilisée par DELAY_REQ.

Tableau 128 – Table d'états DELAY_REQ

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P-ON	=> Nrepeat := 0 SequenceID := 0 ErrorCounter := 0 RESET_RCF_PEER RESET_LINE_DELAY	DOWN
2	DOWN	MAUType_Change_ind () /!LINK_OK => ignore	DOWN
3	DOWN	MAUType_Change_ind () /LINK_OK => A_PDU := PTCP_DELAY_REQ_PDU StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
4	W_CNF	DelayReq_Cnf () /Status == !OK => SequenceID++	W_TIMER
5	W_CNF	DelayReq_Cnf () /Status == OK => Tx_Tstamp_T1 := Tstamp	W_RSP
6	W_CNF	MAUType_Change_ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
7	W_CNF	MAUType_Change_ind () /LINK_OK => ignore	W_CNF
8	W_CNF	TimerExpired (DelayT) => A_PDU := PTCP_DELAY_REQ_PDU SequenceID++ StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
9	W_CNF	TimerExpired (LineT) /MAX_ERROR_COUNT > ErrorCounter => ErrorCounter++	W_TIMER
10	W_CNF	TimerExpired (LineT) /MAX_ERROR_COUNT <= ErrorCounter => ErrorCounter := 0 RESET_LINE_DELAY	W_TIMER

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
11	W_RSP	DelayResp_Ind () /!CHECK_DELAY_RSP_VALID => ignore	W_RSP
12	W_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID && CHECK_DELAY_RSP_WITH_FU_RSP => Rx_Tstamp2_T2 := Rx_Tstamp1_T2 Rx_Tstamp1_T2 := PTCP_DELAY_RES_PDU.T2TimeStamp Delay := PTCP_DELAY_RES_PDU.Delay Rx_Tstamp_T4 := Tstamp if (Nrepeat > 0) CALC_RCF_PEER	W_FU_RSP
13	W_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID && !CHECK_DELAY_RSP_WITH_FU_RSP && CHECK_DELAY_VALID => Rx_Tstamp2_T2 := Rx_Tstamp1_T2 Rx_Tstamp1_T2 := PTCP_DELAY_RES_PDU.T2TimeStamp Rx_Tstamp_T4 := Tstamp Delay := PTCP_DELAY_RES_PDU.Delay if (Nrepeat > 0) CALC_RCF_PEER CALC_LINE_DELAY Nrepeat++ ErrorCounter := 0	W_TIMER
14	W_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID && !CHECK_DELAY_RSP_WITH_FU_RSP && !CHECK_DELAY_VALID => SET_DUMMY_LINE_DELAY StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL)	W_TIMER
15	W_RSP	DelayFuResp_Ind () => ignore	W_RSP
16	W_RSP	MAUType_Change_ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
17	W_RSP	MAUType_Change_ind () /LINK_OK => ignore	W_RSP
18	W_RSP	TimerExpired (DelayT) => SequenceID++ Nrepeat := 0 RESET_RCF_PEER StartTimer (DelayT, DELAY_REQ_LONG_INTERVAL)	W_TIMER
19	W_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT > ErrorCounter => ErrorCounter++	W_TIMER

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
20	W_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT <= ErrorCounter => ErrorCounter := 0 RESET_LINE_DELAY	W_TIMER
21	W_FU_RSP	DelayResp_Ind () /!CHECK_DELAY_RSP_VALID => ignore	W_FU_RSP
22	W_FU_RSP	DelayResp_Ind () /CHECK_DELAY_RSP_VALID => Nrepeat := 0 StopTimer (DelayT) StartTimer (DelayT, DELAY_REQ_LONG_INTERVAL) RESET_LINE_DELAY	W_TIMER
23	W_FU_RSP	DelayFuResp_Ind () /!CHECK_DELAY_FU_RSP_VALID => ignore	W_FU_RSP
24	W_FU_RSP	DelayFuResp_Ind () /CHECK_DELAY_FU_RSP_VALID => Delay := Delay + PTCP_DELAY_FURES_PDU.Delay if (Nrepeat > 0) CALC_LINE_DELAY Nrepeat++ ErrorCounter := 0	W_TIMER
25	W_FU_RSP	MAUType_Change_Ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
26	W_FU_RSP	MAUType_Change_ind () /!LINK_OK => ignore	W_FU_RSP
27	W_FU_RSP	TimerExpired (DelayT) => A_PDU := PTCP_DELAY_REQ_PDU SequenceID++ StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
28	W_FU_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT > ErrorCounter => ErrorCounter++	W_TIMER
29	W_FU_RSP	TimerExpired (LineT) /MAX_ERROR_COUNT <= ErrorCounter => ErrorCounter := 0 RESET_LINE_DELAY	W_TIMER

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
30	W_TIMER	DelayResp_Ind () => ignore	W_TIMER
31	W_TIMER	DelayFuResp_Ind () => ignore	W_TIMER
32	W_TIMER	MAUType_Change_ind () /!LINK_OK => Nrepeat := 0 ErrorCounter := 0 SequenceID++ StopTimer (DelayT) StopTimer (LineT) RESET_RCF_PEER RESET_LINE_DELAY	DOWN
33	W_TIMER	MAUType_Change_ind () /LINK_OK => ignore	W_TIMER
34	W_TIMER	TimerExpired (DelayT) /Nrepeat < MAX_DELAY_REQ_REPEAT => A_PDU := PTCP_DELAY_REQ_PDU SequenceID++ StartTimer (DelayT, DELAY_REQ_SHORT_INTERVAL) StartTimer (LineT, TIMEOUT_LINE_DELAY) DelayReq_Req ()	W_CNF
35	W_TIMER	TimerExpired (DelayT) /Nrepeat >= MAX_DELAY_REQ_REPEAT => Nrepeat := 0 RESET_RCF_PEER StartTimer (DelayT, DELAY_REQ_LONG_INTERVAL)	W_TIMER
36	W_TIMER	TimerExpired (LineT) => ignore	W_TIMER

4.4.4.2.1.6 Paramètres

4.4.4.2.1.7 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 129 contient les fonctions, macros, temporisateurs et variables utilisés par le DELAY_REQ ainsi que leurs arguments et leurs descriptions.

Tableau 129 – Fonctions, macros, temporisateurs et variables utilisés par le DELAY_REQ

Nom	Type	Fonction/Signification
CALC_LINE_DELAY	Fonction	Retard de ligne pour la moyenne arithmétique des 7 dernières mesures de retard au moins.
CALC_RCF_PEER	Fonction	Calcul de l'écart de fréquence par rapport au voisin
CHECK_DELAY_RSP_VALID	Fonction	Vérifier si la réponse de retard est valide conformément aux règles de codage

Nom	Type	Fonction/Signification
CHECK_DELAY_RSP_WITH_FU_RSP	Fonction	Vérifier si le type de réponse de retard est avec ou sans trame de suivi de la réponse de retard. TRUE:= Attend le suivi de la réponse de délai FALSE:=
CHECK_DELAY_VALID	Fonction	Vérifier le paramètre de PTCP_DELAY_RSP_PDU TRUE:= La mesure de retard de ligne est prise en charge. FALSE:= "Cas particulier"; le nœud connecté ne prend pas en charge la mesure de retard de ligne mais souhaite recevoir les messages de synchronisation. Ce cas est signalé par PTCP_DELAY_RSP_PDU.Delay1ns:= 0x00
LINK_OK	Fonction	Vérifie la liaison MAU_Type == FULL_DUPLEX LINK_Status == Up LINK_Status == UP_CLOSED
RESET_LINE_DELAY	Fonction	Met le retard de ligne à zéro
RESET_RCF_PEER	Fonction	Met RCF_PEER à un
SET_DUMMY_LINE_DELAY	Fonction	Met le retard de ligne à un
StartTimer	Fonction	Cette fonction locale est utilisée pour lancer ou relancer un temporisateur
StopTimer	Fonction	Cette fonction locale est utilisée pour arrêter un temporisateur
TimerExpired	Fonction	Cette fonction locale est émise si un temporisateur expire
DelayFuResp_Ind (S_Port, Tstamp, PTCP_DELAY_FU_RSP_PDU)	Macro	Recevoir PTCP-PDU conformément à PTCP_DELAY_FU_RSP_PDU LMPM_P_Data.ind (CREP, D_Port, Tstamp, DA, SA, A_SDU) Affectations: PTCP_DELAY_FU_RSP_PDU := A_SDU without LT and FrameID
DelayReq_Cnf (D_Port, Tstamp, Status)	Macro	LMPM_P_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Affectations: Status := LMPM_status
DelayReq_Req (Port, PTCP_DELAY_REQ_PDU)	Macro	Créer PTCP-PDU conformément à PTCP_DELAY_REQ_PDU Affectations: D_Port := Port DA :=PTCP_MulticastMACadd for PTCP_DELAY_REQ_PDU SA := local port address PTCP_DELAY_REQ_PDU.SequenceID := SequenceID PTCP_DELAY_REQ_PDU.PortMACAddress := local port address A_SDU := LT, FrameID, PTCP_DELAY_REQ_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
DelayResp_Ind (S_Port, Tstamp, PTCP_DELAY_RSP_PDU)	Macro	Recevoir PTCP-PDU conformément à PTCP_DELAY_RSP_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Affectations: PTCP_DELAY_RSP_PDU := A_SDU without LT and FrameID
DelayT	Temporisateur	Ce temporisateur local est utilisé pour contrôler le début d'une mesure de retard de ligne
LineT	Temporisateur	Ce temporisateur local est utilisé pour surveiller qu'une indication de DelayResponse a été effectuée correctement en temps opportun.
DELAY_REQ_LONG_INTERVAL	Variable	Cette variable locale contient le temps de la pause d'une mesure de retard de ligne. Valeur par défaut: 8 s

Nom	Type	Fonction/Signification
DELAY_REQ_SHORT_INTERVAL	Variable	Cette variable locale contient le temps entre deux mesures de retard de ligne. Valeur par défaut: 200 ms
ErrorCounter	Variable	Cette variable locale contient le nombre réel d'erreurs.
MAX_DELAY_REQ_REPEAT	Variable	Cette variable locale contient le nombre de mesures de retard de ligne lors d'une pause. Valeur par défaut: 5
MAX_ERROR_COUNT	Variable	Cette variable locale contient le nombre de mesures de ligne erronées autorisées avant la réinitialisation du retard de ligne. Valeur par défaut: 3
Rx_Tstamp_T4	Variable	Cette variable locale contient l'horodatage de réception de la trame de réponse du retard.
Rx_Tstamp1_T2	Variable	Cette variable locale contient un horodatage de réception de la trame de demande de délai par le répondeur de délai.
Rx_Tstamp2_T2	Variable	Cette variable locale contient un horodatage de réception de la trame de demande de délai par le répondeur de délai.
SequenceID	Variable	Cette variable locale contient le numéro de séquence du message de demande de retard. Elle doit être incrémentée avec chaque mesure de retard.
TIMEOUT_LINE_DELAY	Variable	Cette variable locale contient la temporisation de la réponse de mesure de retard de ligne. Valeur par défaut: 100 ms
Tx_Tstamp_T1	Variable	Cette variable locale contient l'horodatage de transmission de la trame de demande de retard.

4.4.4.2.2 Machine de Protocole de Réponse de Retard de Ligne

4.4.4.2.2.1 Définitions de primitives

4.4.4.2.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Réponse de Retard de Ligne (DELAY_RSP) sont décrites dans la définition de service et présentées dans le Tableau 130.

Tableau 130 – Primitives distantes émises ou reçues par DELAY_RSP

Primitive	Source	Destination	Paramètres associés	Fonctions
MAUType_Change_ind	IEEE802.3	DELAY_RSP	PortID, MAU_Type, LINK_Status	—
LMPM_P_Data.ind	LMPM	DELAY_RSP	CREP, S_Port, Tstamp, DA, SA, A_SDU	—
LMPM_P_Data.cnf ()	LMPM	DELAY_RSP	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.req	DELAY_RSP	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.2.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DELAY_RSP sont décrites dans la définition de service et présentées dans le Tableau 131.

Tableau 131 – Primitives locales émises ou reçues par DELAY_RSP

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.4.4.2.2 Schéma de transition d'états

Le schéma de transition d'états du DELAY_RSP est présenté à la Figure 41.

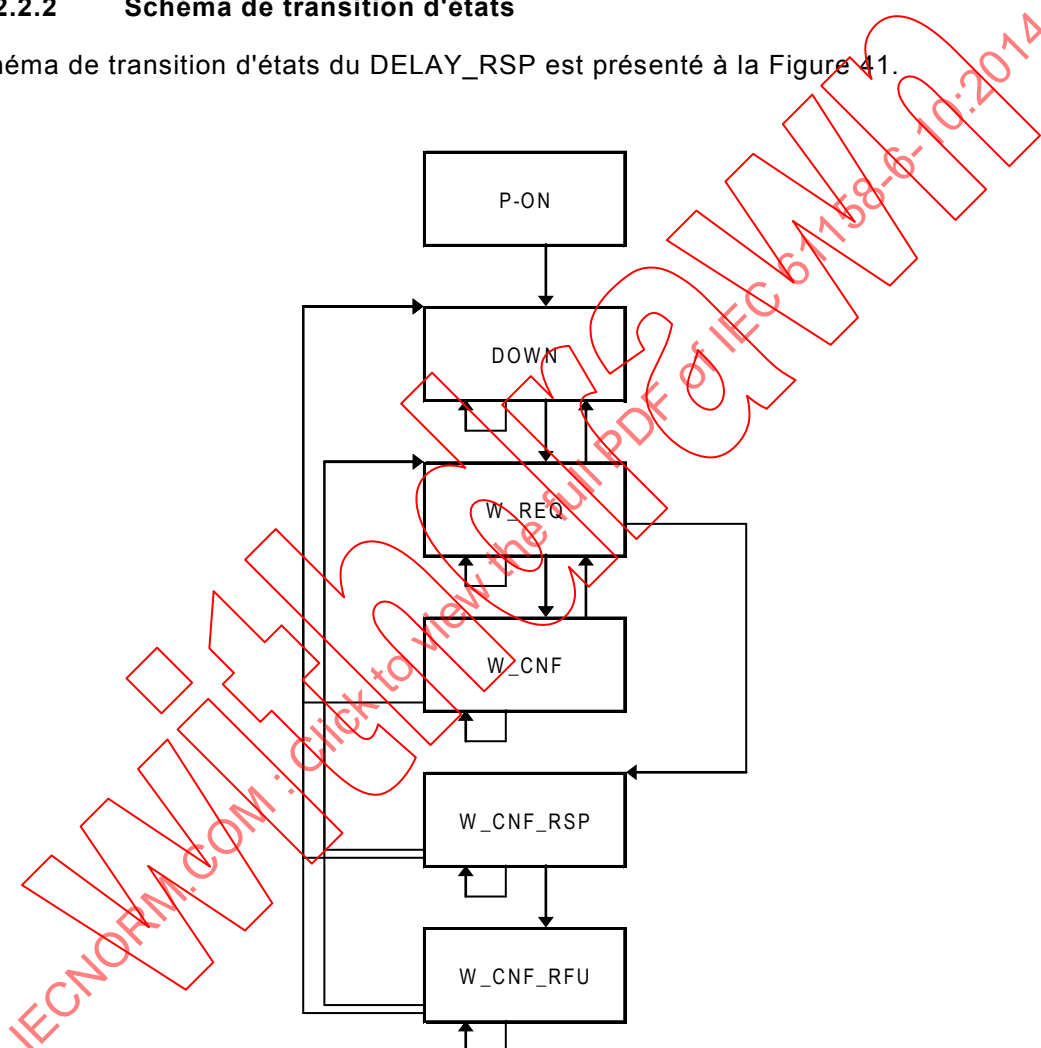


Figure 41 – Schéma de transition d'états de DELAY_RSP

Etats du DELAY_RSP

P-ON	Initialisation des données.
DOWN	La liaison du port est interrompue. Attend l'établissement de liaison.
W_REQ	Si un message de demande de retard est reçu, l'horodatage de réception est enregistré et DelayResonder envoie un message de réponse de délai.
W_CNF	Attend la confirmation de la demande de réponse de retard.
W_CNF_RSP	Attend la confirmation du message de réponse de retard, calcule le délai de réponse du retard et envoie un message de

suivi de réponse de retard.

W_CNF_RFU

Attend la confirmation du message de suivi de réponse de retard.

4.4.4.2.2.3 Description du diagramme d'états

Le diagramme d'états de DelayResponder prend en charge la fonction de répondeur de la mesure de retard de ligne. Il reçoit la demande de mesure de retard de ligne et répond par une réponse de mesure de retard de ligne.

Le DelayResponder doit répondre avec la DelayResponse sans suivi si possible. Dans le cas contraire, il doit utiliser la DelayResponse avec suivi.

Chaque message de demande de retard doit être répondu par un message de réponse de retard.

4.4.4.2.2.4 Table d'états DELAY_RSP

Le Tableau 132 contient la table d'états utilisée par DELAY_RSP.

Tableau 132 – Table d'états DELAY_RSP

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P-ON	=>	DOWN
2	DOWN	MAUType_Change_ind () // LINK_OK => ignore	DOWN
3	DOWN	MAUType_Change_ind () //LINK_OK => ignore	W_REQ
4	W_REQ	DelayReq_Ind () //FU_RES => store PTCP_DELAY_REQ_PDU A_PDU := PTCP_DELAY_RES_PDU Rx_Tstamp_T2 := Tstamp PTCP_DELAY_RES_PDU.T2TimeStamp := Rx_Tstamp_T2 DelayResp_Req ()	W_CNF_RSP
5	W_REQ	DelayReq_Ind () //FU_RES => store PTCP_DELAY_REQ_PDU A_PDU := PTCP_DELAY_RES_PDU Rx_Tstamp_T2 := Tstamp PTCP_DELAY_RES_PDU.T2TimeStamp := Rx_Tstamp_T2 DelayResp_Req ()	W_CNF
6	W_REQ	DelayReq_Ind () //FU_TIMESTAMP => store PTCP_DELAY_REQ_PDU A_PDU := PTCP_DELAY_RES_PDU PTCP_DELAY_RES_PDU.Delay := 0 DelayResp_Req ()	W_CNF

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
7	W_REQ	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
8	W_REQ	MAUType_Change_ind () /LINK_OK => ignore	W_REQ
9	W_CNF	DelayResp_Cnf () => ignore	W_REQ
10	W_CNF	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
11	W_CNF	MAUType_Change_ind () /LINK_OK => ignore	W_CNF
12	W_CNF_RSP	DelayResp_Cnf () /Status != OK => ignore	W_REQ
13	W_CNF_RSP	DelayResp_Cnf () /Status == OK => Tx_Tstamp_T3 := Tstamp A_PDU := PTCP_DELAY_FU_RES_PDU PTCP_DELAY_FU_RES_PDU.Delay := CALC_RESIDENTIAL_TIME DelayFuResp_Req ()	W_CNF_RFU
14	W_CNF_RSP	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
15	W_CNF_RSP	MAUType_Change_ind () /LINK_OK => ignore	W_CNF_RSP
16	W_CNF_RFU	DelayFuResp_Cnf () => ignore	W_REQ
17	W_CNF_RFU	MAUType_Change_ind () /! LINK_OK => ignore	DOWN
18	W_CNF_RFU	MAUType_Change_ind () /LINK_OK => ignore	W_CNF_RFU

4.4.4.2.2.5 Fonctions, Macros, Temporisateurs et Variables

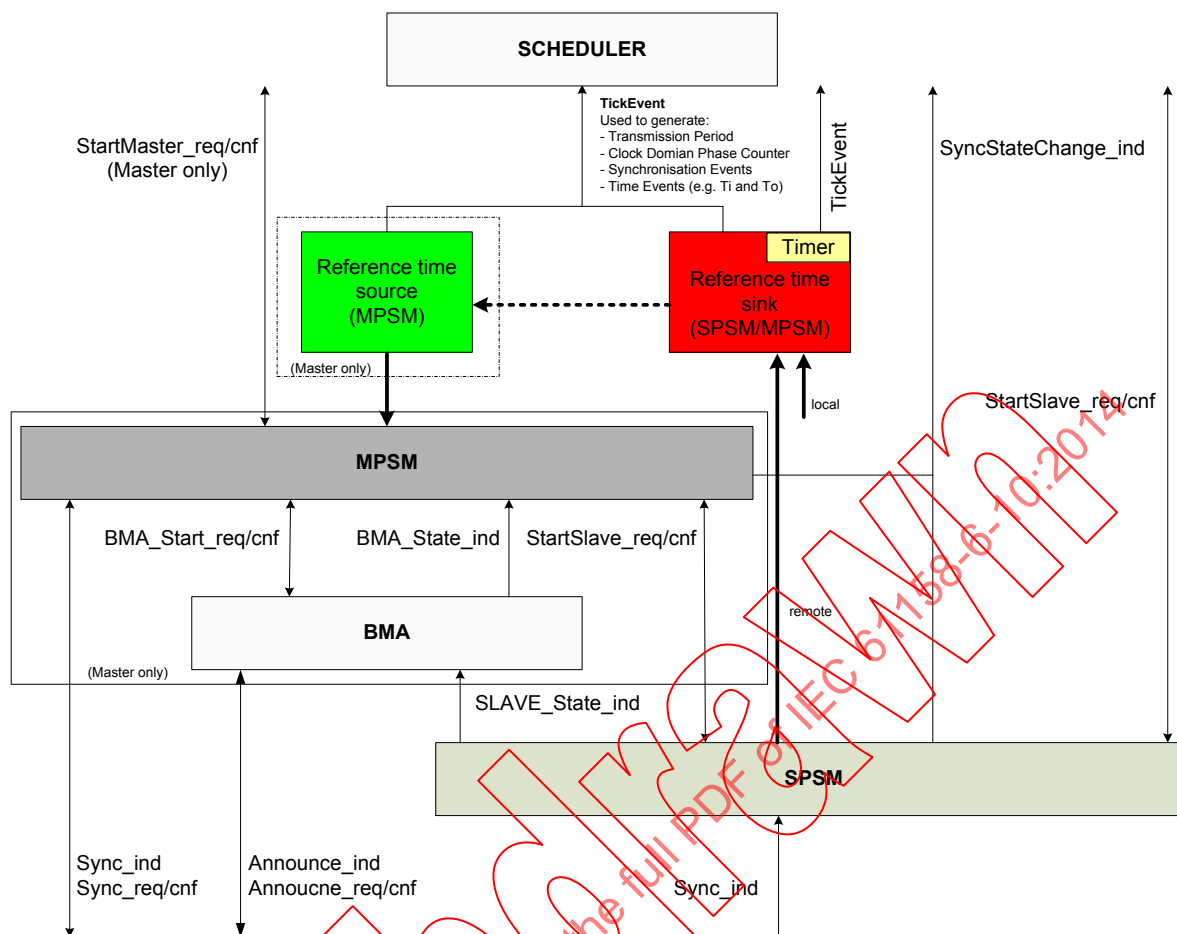
Le Tableau 133 contient les fonctions, macros, temporisateurs et variables utilisés par le DELAY_RSP ainsi que leurs arguments et leurs descriptions.

Tableau 133 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DELAY_RSP

Nom	Type	Fonction/Signification
DelayResp_Req (Port, PTCP_DELAY_RSP_PDU)	Macro	Créer PTCP-PDU conformément à PTCP_DELAY_RSP_PDU Attributions: D_Port := Port DA := Multicast-MAC-Address for PTCP_DELAY_RSP_PDU SA := local port address PTCP_DELAY_RSP_PDU.SequenceID := PTCP_DELAY_REQ_PDU.SequenceID PTCP_DELAY_RSP_PDU.PortMACAddress := PTCP_DELAY_REQ_PDU.PortMACAddress PTCP_DELAY_REQ_PDU.T2PortRxDelay := T2_PortRxDelay PTCP_DELAY_REQ_PDU.T3PortTxDelay := T3_PortTxDelay PTCP_DELAY_REQ_PDU.T2TimeStamp := T2_TimeStamp A_SDU := LT, FrameID, PTCP_DELAY_RSP_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
DelayFuResp_Req (Port, ResTime, PTCP_DELAY_FU_RSP_PDU)	Macro	Créer PTCP-PDU conformément à PTCP_DELAY_FU_RSP_PDU Affectations: D_Port := Port DA := Multicast-MAC-Address for PTCP_DELAY_FU_RSP_PDU SA := local source address PTCP_Delay := ResTime A_SDU := LT, FrameID, PTCP_DELAY_FU_RSP_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
DelayResp_Cnf (D_Port, Tstamp, Status)	Macro	LMPM_P_Data.cnf (CREP, D_Port, TStamp, LMPM_status) Affectations: Status := LMPM_status
DelayFuResp_Cnf (D_Port, Tstamp, Status)	Macro	LMPM_P_Data.cnf (CREP, D_Port, TStamp, LMPM_status) Affectations: Status := LMPM_status
DelayReq_Ind (S_Port, Tstamp, PTCP_DELAY_REQ_PDU)	Macro	Recevoir PTCP-PDU conformément à PTCP_DELAY_RSP_PDU LMPM_P_Data.ind (CREP, S_Port, TStamp, DA, SA, A_SDU) Affectations: PTCP_DELAY_REQ_PDU := A_SDU without LT and FrameID
Rx_Tstamp_T2	Variable	Cette variable locale contient l'horodatage de réception du message de demande de retard.
Tx_Tstamp_T3	Variable	Cette variable locale contient l'horodatage de transmission du message de réponse de retard.
LINK_OK	Macro	Vérifier si MAU_Type est FULL_DUPLEX et si LINK_Status est UP ou BLOCKED.
CALC_RESIDENTIAL_TIME	Macro	Calculer le temps résidentiel (Tx_Tstamp_T3 - Rx_Tstamp_T2)

4.4.4.3 Vue d'ensemble des tables d'états de maître et d'esclave

La Figure 42 montre une vue d'ensemble de l'interaction entre les tables d'états PTCP. Pour les appareils dépourvus de support PTCP ou pour les esclaves dépourvus de maître PTCP actif, le temps de référence est généré par des moyens locaux.



Légende

Anglais	Français
Used to generate	Permet de générer:
Transmission period	une période de transmission
Clock Domain Phase Counter	un compteur de phases de domaine d'horloge
Synchronization Events	des événements de synchronisation
Time Events (e.g. Ti and To)	des événements chronologiques (Ti et To, par exemple)
Reference time source (PTCP Master)	Source de temps de référence (maître PTCP)
Reference time sink (PTCP Slave/PTCP Master)	Temps de référence (esclave PTCP/maître PTCP)
PTCP MASTER	MAÎTRE PTCP
(Master only)	(maître uniquement)
PTCP SLAVE	ESCLAVE PTCP

Figure 42 – Vue d'ensemble de PTCP

Pour l'utilisation d'un PTCP sans connexion, il convient que les principes de l'IEEE 802.1AS soient appliqués.

4.4.4.4 Algorithme de meilleur maître (Best Master Algorithm)

4.4.4.4.1 Machine de Protocole BMA (Best-Master-Algorithm)

4.4.4.4.1.1 Définitions de primitives

4.4.4.4.1.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole BMA (Best-Master-Algorithm) (SYN_BMA) sont décrites dans la définition de service et présentées dans le Tableau 134.

Tableau 134 – Primitives distantes émises ou reçues par SYN_BMA

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_A_Data.req	BMA	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—
LMPM_A_Data.cnf	LMPM	BMA	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_A_Data.ind	LMPM	BMA	CREP, S_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.4.1.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par SYN_BMA sont décrites dans la définition de service et présentées dans le Tableau 135.

Tableau 135 – Primitives locales émises ou reçues par SYN_BMA

Primitive	Source	Destination	Paramètres associés	Fonctions
BMA_Start_req	MPSM	BMA	—	Active la fonction de meilleur maître
BMA_Start_cnf (+/-)	BMA	MPSM	—	Confirmation du démarrage de la fonction de meilleur maître
BMA_Stop_req	MPSM	BMA	—	Désactive la fonction de meilleur maître
BMA_Stop_cnf (+/-)	BMA	MPSM	—	Confirmation de l'arrêt de la fonction de meilleur maître
BMA_State_ind	BMA	MPSM	Rôle	Au cours de la période de démarrage, le BMA choisit le meilleur maître de synchronisation. Valeur admise pour Role: - MASTER - SLAVE
SLAVE_State_ind	SPSM	BMA	Etat	Indication de perte du maître. L'événement est généré par l'esclave. - MASTER_LOST

Primitive	Source	Destination	Paramètres associés	Fonctions
Announce_req	BMA	LMPM	PTCP_ANNOUNCE_PDU	Créer PTCP-PDU en fonction de PTCP_ANNOUNCE_PDU Affectations: D_Port := AUTO DA := PTCP_MulticastMACadd for PTCP_ANNOUNCE_PDU SA := local source address PTCP_ANNOUNCE_PDU.SequenceID := SequenceID PTCP_ANNOUNCE_PDU.DomainUUID := DomainUUID PTCP_ANNOUNCE_PDU.MasterSourceAddress := local master address PTCP_ANNOUNCE_PDU.MasterPriority1 := Priority PTCP_ANNOUNCE_PDU.ClockClass := Class PTCP_ANNOUNCE_PDU.ClockAccuracy := Accuracy PTCP_ANNOUNCE_PDU.ClockVariance := Variance PTCP_ANNOUNCE_PDU.MasterPriority2 := 0xFF A_SDU := LT, FrameID, PTCP_ANNOUNCE_PDU LMPM_A_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
Announce_cnf (+/-)	LMPM	BMA	Etat	LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) Affectations: Status := LMPM_status
Announce_ind	LMPM	BMA	PTCP_ANNOUNCE_PDU	Recevoir PTCP-PDU conformément à PTCP_SYNC_PDU LMPM_A_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Affectations: PTCP_SYNC_PDU := A_SDU without LT and FrameID

4.4.4.4.1.2 Schéma de transition d'états

Le schéma de transition d'états du SYN_BMA est présenté à la Figure 43.

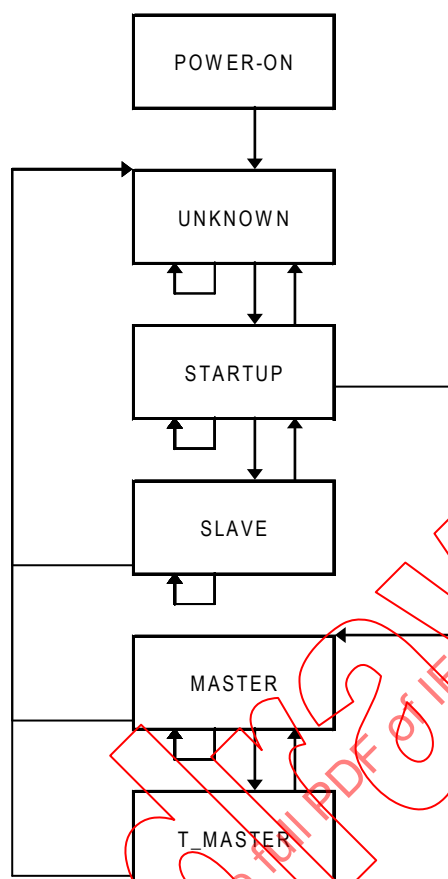


Figure 43 – Schéma de transition d'états de SYN_BMA

Etats du SYN_BMA

POWER-ON	Initialisation des données
UNKNOWN	Le rôle de synchronisation est inconnu. Un rôle doit être défini grâce aux services de PTCP.
STARTUP	Rôle défini grâce aux services de PTCP. Le maître potentiel commence à envoyer des messages d'annonce.
SLAVE	Le maître potentiel à l'état SLAVE doit être synchronisé à l'aide des messages de synchronisation provenant du maître actif dans le domaine de synchronisation.
MASTER	Dans cet état, et compte tenu des attributs de synchronisation, l'appareil est choisi comme étant le meilleur maître.
T_MASTER	Il s'agit d'un état de transition qui indique la réception de messages d'annonce provenant d'un autre maître potentiel du domaine de synchronisation.

4.4.4.4.1.3 Description du diagramme d'états

Le diagramme d'états BMA doit choisir le meilleur maître parmi tous les appareils annoncés pouvant tenir ce rôle pendant la période de démarrage. Ces décisions sont prises en local par chaque maître. Il existe une instance du diagramme d'états pour chaque SyncID de PTCP.

Etant donné que tous les appareils ne peuvent pas être maîtres ou il convient qu'ils ne le soient pas, le comportement du diagramme d'états dépend de son rôle. Ce rôle est défini par les services PTCP. Les rôles sont les suivants: maître, maître secondaire et esclave. Un maître a également besoin d'une fonction d'esclave.

Le rôle de maître ou maître secondaire joué par un appareil est décidé au démarrage par l'algorithme BMA (Best Master Algorithm). Chaque maître potentiel doit envoyer des messages d'annonce pour participer au choix du maître et du maître secondaire.

Le choix du maître a lieu en deux étapes. Avec la première annonce reçue, le maître potentiel est enregistré dans une liste. Avec l'annonce suivante, le SYN_BMA choisit ce maître, s'il s'agit du meilleur maître connu.

Pendant la compétition, tous les maîtres potentiels sont rassemblés dans une liste. Une fois le choix effectué, ils sont retirés de la liste.

Une fois la décision prise, le meilleur maître choisi envoie des messages de synchronisation. Les autres maîtres potentiels arrêtent d'envoyer des messages d'annonce et doivent devenir des esclaves. En cas de défaillance du maître actif, les maîtres de secours recommencent à envoyer des messages de synchronisation et d'annonce.

4.4.4.4.1.4 Table d'états SYN_BMA

Le Tableau 136 contient la table d'états utilisée par SYN_BMA.

Tableau 136 – Table d'états SYN_BMA

#	Etat en cours	Evénement /Condition => Action	Etat suivant
1	POWER-ON	=> init MasterList SequenceID := 0 BestMaster = FALSE ACTIVE_MASTER_FLAG := FALSE AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	UNKNOWN
2	UNKNOWN	BMA_Start_req () => Status := OK A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) StartTimer (StartupTimer, MasterStartupTime) BMA_Start_cnf (+) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	STARTUP
3	UNKNOWN	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
4	UNKNOWN	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	UNKNOWN
5	STARTUP	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	STARTUP
6	STARTUP	BMA_Stop_req () => StopTimer (AnnTimer) StopTimer (StartupTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
7	STARTUP	Announce_cnf () => ignore	STARTUP
8	STARTUP	SLAVE_State_ind (State) => ignore	STARTUP
9	STARTUP	Announce_ind () /!VALID_SYNC_MASTER => ignore	STARTUP
10	STARTUP	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST	STARTUP
11	STARTUP	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY	STARTUP
12	STARTUP	TimerExpired (AnnTimer) /!LOCAL_IS_BEST_MASTER => BestMaster := FALSE StopTimer (StartupTimer) StartTimer (AnnTimer, MasterAnnounceTime) BMA_State_ind (SLAVE)	SLAVE
13	STARTUP	TimerExpired (AnnTimer) /LOCAL_IS_BEST_MASTER => BestMaster := TRUE SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) BMA_State_ind (MASTER) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	STARTUP
14	STARTUP	TimerExpired (StartupTimer) => ACTIVE_MASTER_FLAG := TRUE	MASTER
15	STARTUP	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	STARTUP
16	SLAVE	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	SLAVE
17	SLAVE	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
18	SLAVE	Announce_cnf () => ignore	SLAVE
19	SLAVE	SLAVE_State_ind (State) /State != MASTER_LOST => ignore	SLAVE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
20	SLAVE	SLAVE_State_ind (State) /State == MASTER_LOST && !BestMaster => ignore	SLAVE
21	SLAVE	SLAVE_State_ind (State) /State == MASTER_LOST && BestMaster => SequenceID++ A_PDU:= PTCP_ANNOUNCE_PDU StartTimer (StartupTimer, MasterStartupTime) BMA_State_ind (MASTER) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	STARTUP
22	SLAVE	Announce_ind () /!VALID_SYNC_MASTER => ignore	SLAVE
23	SLAVE	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST	SLAVE
24	SLAVE	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY	SLAVE
25	SLAVE	TimerExpired (AnnTimer) /!LOCAL_IS_BEST_MASTER => BestMaster := FALSE StartTimer (AnnTimer, MasterAnnounceTime)	SLAVE
26	SLAVE	TimerExpired (AnnTimer) /LOCAL_IS_BEST_MASTER => BestMaster := TRUE StartTimer (AnnTimer, MasterAnnounceTime)	SLAVE
27	SLAVE	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	SLAVE
28	MASTER	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	MASTER
29	MASTER	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
30	MASTER	Announce_cnf () => ignore	MASTER
31	MASTER	SLAVE_State_ind (State) => ignore	MASTER

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
32	MASTER	Announce_ind () /!VALID_SYNC_MASTER => ignore	MASTER
33	MASTER	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST	MASTER
34	MASTER	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY	MASTER
35	MASTER	TimerExpired (AnnTimer) /!MASTER_LIST_EMPTY => SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) StartTimer (StartupTimer, MasterStartupTime) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	T_MASTER
36	MASTER	TimerExpired (AnnTimer) /MASTER_LIST_EMPTY => StartTimer (AnnTimer, MasterAnnounceTime)	MASTER
37	MASTER	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	MASTER
38	T_MASTER	BMA_Start_req () => Status := WRONG_SEQUENCE BMA_Start_cnf (-)	T_MASTER
39	T_MASTER	BMA_Stop_req () => StopTimer (AnnTimer) Status := OK BMA_Stop_cnf (+)	UNKNOWN
40	T_MASTER	Announce_cnf () => ignore	T_MASTER
41	T_MASTER	SLAVE_State_ind (State) => ignore	T_MASTER
42	T_MASTER	Announce_ind () /!VALID_SYNC_MASTER => ignore	T_MASTER
43	T_MASTER	Announce_ind () /VALID_SYNC_MASTER && !CHECK_IN_MASTER_LIST => ADD_TO_MASTER_LIST StartTimer (StartupTimer, MasterStartupTime)	T_MASTER
44	T_MASTER	Announce_ind () /VALID_SYNC_MASTER && CHECK_IN_MASTER_LIST => UPDATE_MASTER_ENTRY StartTimer (StartupTimer, MasterStartupTime)	T_MASTER

#	Etat en cours	Événement /Condition =>Action	Etat suivant
45	T_MASTER	TimerExpired (AnnTimer) //MASTER_LIST_EMPTY => SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) StartTimer (StartupTimer, MasterStartupTime) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	T_MASTER
46	T_MASTER	TimerExpired (AnnTimer) //MASTER_LIST_EMPTY => SequenceID++ A_PDU := PTCP_ANNOUNCE_PDU StartTimer (AnnTimer, MasterAnnounceTime) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Announce_req (i, PTCP_ANNOUNCE_PDU)	T_MASTER
47	T_MASTER	TimerExpired (StartupTimer) => ignore	MASTER
48	T_MASTER	TimerExpired (MasterListAgingTimer) => AGING_MASTER_LIST StartTimer (MasterListAgingTimer, MasterAgingTime)	T_MASTER

4.4.4.4.1.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 137 contient les fonctions, macros, temporisateurs et variables utilisés par le SYN_BMA ainsi que leurs arguments et leurs descriptions.

Tableau 137 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYN_BMA

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction locale lance ou relance un temporisateur
StopTimer	Fonction	Cette fonction locale arrête un temporisateur
TimerExpired	Fonction	Cette fonction locale est émise si un temporisateur expire
ADD_TO_MASTER_LIST	Macro	Insérer le maître de synchronisation dans la liste des maîtres de synchronisation distants sm_sa:= PTCP_ANNOUNCE_PDU.MasterSourceAddress if (MasterList.Num_entry < MAX_CNT_REMOTE_MASTER) MasterList.Insert(sm_sa) MasterList.Entry{sm_sa}.Priority1 := PTCP_ANNOUNCE_PDU.Priority1 MasterList.Entry{sm_sa}.Accuracy := PTCP_ANNOUNCE_PDU.Accuracy MasterList.Entry{sm_sa}.Variance := PTCP_ANNOUNCE_PDU.Variance MasterList.Entry{sm_sa}.Priority2 := PTCP_ANNOUNCE_PDU.Priority2 MasterList.Entry{sm_sa}.SA := sm_sa MasterList.Entry{sm_sa}.Receipt := 1 MasterList.Num_entry++
AGING_MASTER_LIST	Macro	Vieillessement des entrées dans la liste des maîtres de synchronisation distants pour m := 0 à MasterList.Num_entry if (MasterList.Entry[m].Receipt == 0) MasterList.Num_entry-- MasterList.Remove(sm_sa)

Nom	Type	Fonction/Signification
Announce_cnf (+/-) (State)	Macro	LMPM_A_Data.cnf (CREP, D_Port, TStamp, LMPM_status) Affectations: Status := LMPM_status
Announce_ind (PTCP_ANNOUNCE_PDU)	Macro	Recevoir PTCP-PDU conformément à PTCP_ANNOUNCE_PDU LMPM_A_Data.ind (CREP, S_Port, TStamp, DA, SA, A_SDU) Affectations: PTCP_ANNOUNCE_PDU := A_SDU without LT and FrameID
Announce_req (Port, PTCP_ANNOUNCE_PDU)	Macro	Créer PTCP-PDU en fonction de PTCP_ANNOUNCE_PDU Affectations: D_Port := Port DA := PTCP_MulticastMACadd for PTCP_ANNOUNCE_PDU SA := local source address PTCP_ANNOUNCE_PDU.SequenceID := SequenceID PTCP_ANNOUNCE_PDU.DomainUUID := DomainUUID PTCP_ANNOUNCE_PDU.MasterSourceAddress := local master address PTCP_ANNOUNCE_PDU.MasterPriority1 := Priority PTCP_ANNOUNCE_PDU.ClockClass := Class PTCP_ANNOUNCE_PDU.ClockAccuracy := Accuracy PTCP_ANNOUNCE_PDU.ClockVariance := Variance PTCP_ANNOUNCE_PDU.MasterPriority2 := 0xFF A_SDU := LT, FrameID, PTCP_ANNOUNCE_PDU LMPM_A_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
CHECK_IN_MASTER_LIST	Macro	Master in PTCP_SubDomainUUID master list sm_sa := PTCP_ANNOUNCE_PDU.MasterSourceAddress sm_sa in MasterList
init MasterList	Macro	Initialise l'ensemble des données pour les maîtres distants MasterListNum_entry := 0
LOCAL_IS_BEST_MASTER	Macro	Effectue une comparaison conformément à 4.4.4.4.1.6 et retourne TRUE si le maître local est trouvé, dans le cas contraire, retourne FALSE.
TRANSMIT_PORT_VALID	Macro	Le Tableau 152 doit être appliqué.
UPDATE_MASTER_ENTRY	Macro	Mettre à jour les entrées dans la liste des maîtres de synchronisation distants MasterList.Entry{sm_sa}.Priority1 := PTCP_ANNOUNCE_PDU.Priority1 MasterList.Entry{sm_sa}.Accuracy := PTCP_ANNOUNCE_PDU.Accuracy MasterList.Entry{sm_sa}.Variance := PTCP_ANNOUNCE_PDU.Variance MasterList.Entry{sm_sa}.Priority2 := PTCP_ANNOUNCE_PDU.Priority2 MasterList.Entry{sm_sa}.Receipt += 1
VALID_SYNC_MASTER	Macro	Le domaine est pris en charge PTCP_ANNOUNCE_PDU.DomainUUID == DomainUUID
AnnTimer	Temporisateur	Ce temporisateur local est utilisé pour créer l'intervalle pour l'envoi des trames d'annonce
MasterListAgingTimer	Temporisateur	Ce temporisateur local est utilisé pour créer l'intervalle pour le vieillissement des entrées de la liste des maîtres.
StartupTimer	Temporisateur	Ce temporisateur local est utilisé pour générer l'intervalle pour la vérification des trames d'annonce
MasterAgingTime	Variable	Temps du vieillissement de la liste des maîtres Valeur par défaut: "2 x Temporisation de prise de contrôle de PTCP"
MasterAnnounceTime	Variable	Une trame d'annonce doit être envoyée à chaque MasterAnnounceTime jusqu'au MasterStartupTime. Valeur par défaut: 100 ms
MasterStartupTime	Variable	Temps pour la vérification des trames d'annonce à la réception. Valeur par défaut: 10 s

4.4.4.4.1.6 Recherche du meilleur Maître de Synchronisation

PTCP contient la prise en charge de plusieurs maîtres de synchronisation. Le filtrage des maîtres de synchronisation stockés pour trouver celui "à utiliser" est effectué par BMA à l'aide des attributs-clés SyncID et PTCP_SubdomainUUID ainsi que des règles de vérification suivantes:

NOTE 1 La PTCP_ClockClass n'est pas utilisée.

NOTE 2 Les valeurs inférieures l'emportent.

- 1) Vérifier si un maître de synchronisation PTCP_MasterPriority1.Active == TRUE existe

Le BMA vérifie si un maître de synchronisation ACTIF existe. Si un autre maître de synchronisation est actif, le maître chargé de la vérification passe en mode esclave de synchronisation.

Si aucun maître de synchronisation ACTIF n'existe, il continue avec la prochaine vérification.

NOTE 3 Si le maître de synchronisation local est actif, il reste actif jusqu'à ce que l'application en décide autrement.

- 2) Trouver le maître de synchronisation avec le meilleur PTCP_MasterPriority1.Level

Le BMA recherche les maîtres de synchronisation avec la meilleure valeur, en prenant en compte sa propre valeur.

Si plus d'un maître de synchronisation existe dans la liste résiduelle, continuer avec la prochaine vérification puis utiliser celui trouvé.

- 3) Trouver le maître de synchronisation avec le meilleur PTCP_MasterPriority1.Priority

Le BMA recherche les maîtres de synchronisation avec la meilleure valeur de la liste résiduelle.

Si plus d'un maître de synchronisation existe dans la liste résiduelle, continuer avec la prochaine vérification puis utiliser celui trouvé.

- 4) Trouver le maître de synchronisation avec le meilleur PTCP_MasterPriority2

Le BMA recherche les maîtres de synchronisation avec la meilleure valeur de la liste résiduelle.

Si plus d'un maître de synchronisation existe dans la liste résiduelle, continuer avec la prochaine vérification puis utiliser celui trouvé.

- 5) Trouver le maître de synchronisation avec le meilleur PTCP_ClockVariance

Le BMA recherche les maîtres de synchronisation avec la meilleure valeur de la liste résiduelle.

Si plus d'un maître de synchronisation existe dans la liste résiduelle, continuer avec la prochaine vérification puis utiliser celui trouvé.

- 6) Trouver le maître de synchronisation avec le meilleur PTCP_ClockAccuracy

Le BMA recherche les maîtres de synchronisation avec la meilleure valeur de la liste résiduelle.

Si plus d'un maître de synchronisation existe dans la liste résiduelle, continuer avec la prochaine vérification puis utiliser celui trouvé.

7) Trouver le maître de synchronisation avec la meilleure PTCP_MasterSourceAddress

Le BMA recherche les maîtres de synchronisation avec la valeur la plus basse (en comparant avec les octets de zéro à cinq) de PTCP_MasterSourceAddress de la liste résiduelle.

Utiliser celui trouvé.

NOTE 4 PTCP_MasterSourceAddress est utilisé comme "disjoncteur".

4.4.4.5 Machine de Protocole de Maître PTCP

4.4.4.5.1 Définitions de primitives

4.4.4.5.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Maître PTCP (SYN_MPSM) sont décrites dans la définition de service et présentées dans le Tableau 138.

Tableau 138 – Primitives distantes émises ou reçues par SYN_MPSM

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_P_Data.req	MPSM	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—
LMPM_P_Data.cnf	LMPM	MPSM	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.ind	LMPM	MPSM	CREP, S_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.5.1.1.1 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par SYN_MPSM sont décrites dans la définition de service et présentées dans le Tableau 139.

Tableau 139 – Primitives locales émises ou reçues par SYN_MPSM

Primitive	Source	Destination	Paramètres associés	Fonctions
StartMaster_req	PTCP ASE	MPSM	—	Active la fonction de maître
StartMaster_cnf	MPSM	PTCP ASE	Status	Confirmation du démarrage d'un maître
StopMaster_req	PTCP ASE	MPSM	—	Désactive la fonction de maître
StopMaster_cnf	MPSM	PTCP ASE	Status	Confirmation de l'arrêt d'un maître
StartSlave_req	MPSM	SPSM	—	Active la fonction d'esclave
StartSlave_cnf	SPSM	MPSM	—	Confirmation du démarrage d'un esclave
StopSlave_req	MPSM	SPSM	—	Désactive la fonction d'esclave
StopSlave_cnf	SPSM	MPSM	—	Confirmation de l'arrêt d'un esclave
BMA_Start_req	MPSM	BMA	—	Active la fonction de meilleur maître

Primitive	Source	Destination	Paramètres associés	Fonctions
BMA_Start_cnf	BMA	MPSM	—	Confirmation du démarrage de la fonction de meilleur maître
BMA_Stop_req	MPSM	BMA	—	Désactive la fonction de meilleur maître
BMA_Stop_cnf	BMA	MPSM	—	Confirmation de l'arrêt de la fonction de meilleur maître
BMA_State_ind	BMA	MPSM	Rôle	Au cours de la période de démarrage, le BMA choisit le meilleur maître de synchronisation. Valeur admise pour Role: - MASTER - SLAVE
SyncStateChange_ind	MSPM	FSPM	Status	Indication pour les événements SINGLE_MASTER - MULTIPLE_MASTER

4.4.4.5.2 Schéma de transition d'états

Le schéma de transition d'états du SYN_MPSM est présenté à la Figure 44.

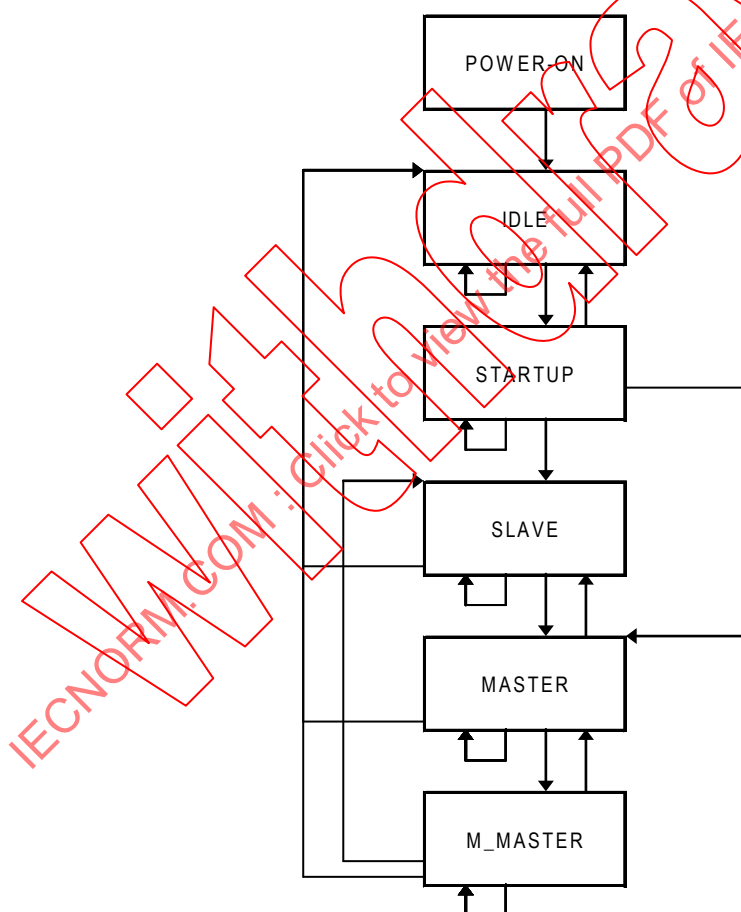


Figure 44 – Schéma de transition d'états de SYN_MPSM

Etats du SYN_MPSM

POWER-ON

Après le démarrage, la gestion de données doit être initialisée (POWER-ON) et le rôle de synchronisation défini (IDLE).

IDLE

Le rôle du maître est défini par les services de PTCP. Ensuite, le MPSM passe de l'état IDLE à l'état STARTUP et démarre le diagramme d'états de BMA.

STARTUP	Pendant le démarrage, le MPSM est à l'état STARTUP.
SLAVE	Le maître est synchronisé par des messages de synchronisation provenant d'un autre maître du domaine de synchronisation.
MASTER	Il s'agit du maître principal actif. Il envoie donc les messages de synchronisation.
M_MASTER	Cet état traite le problème lié à la présence de plusieurs maîtres.

4.4.4.5.3 Description du diagramme d'états

Le diagramme d'états de protocole maître contrôle la transition entre les rôles maître et esclave. Les transitions sont exécutées en local par chaque maître pour tous les SyncID de PTCP pris en charge et les SubDomainUUID de PTCP affectés par l'ingénierie. Il existe donc une instance du diagramme d'états pour chaque SyncID avec l'attribut-clé SubDomainUUID. Pour procéder à ces transitions, le MPSM doit lancer des messages de synchronisation et d'annonce. Les messages d'annonce doivent être envoyés par SYN_BMA pour faire connaître les capacités du maître et signaler qu'un seul maître est actif.

L'état STARTUP et l'algorithme de meilleur maître doivent empêcher la présence de plusieurs maîtres primaires actifs et algorithmes de meilleur maître. Par conséquent, la décision de passer de l'état STARTUP à l'état MASTER (maître de synchronisation primaire) est prise par SYN_BMA en toute connaissance des maîtres concourants.

Si un maître actif (doté de peu de capacités) existe déjà et que, suite au démarrage, un nouveau maître se présente, ce dernier nécessite une synchronisation avec le maître actif. Pour la synchronisation, le MPSM passe à l'état SLAVE.

4.4.4.5.4 Table d'états SYN_MPSM

Le Tableau 140 contient la table d'états utilisée par SYN_MPSM.

Tableau 140 – Table d'états SYN_MPSM

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	POWER-ON	=> SequenceID := 0 MultipleMaster := FALSE	IDLE
2	IDLE	StartMaster_req () => Status := OK BMA_Start_req () StartMaster_cnf (Status)	STARTUP
3	IDLE	StopMaster_req () => Status := OK StopMaster_cnf (Status)	IDLE
4	IDLE	Sync_cnf () => ignore	IDLE
5	IDLE	BMA_Stop_cnf () => ignore	IDLE
6	STARTUP	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status)	STARTUP

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
7	STARTUP	StopMaster_req () => Status := OK StopMaster_cnf (Status) BMA_Stop_req ()	IDLE
8	STARTUP	Sync_cnf () => ignore	STARTUP
9	STARTUP	BMA_State_ind (State) /State == SLAVE => Status := OK StartSlave_req ()	SLAVE
10	STARTUP	BMA_State_ind (State) /State == MASTER => State := SINGLE_MASTER SequenceID++ StartTimer (SyncTimer, SyncInterval) StartTimer (TimeoutT, SyncTimeOut) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	MASTER
11	SLAVE	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status) BMA_Start_req ()	SLAVE
12	SLAVE	StopMaster_req () => Status := OK BMA_Stop_req () StopMaster_cnf (Status)	IDLE
13	SLAVE	Sync_cnf () => ignore	SLAVE
14	SLAVE	BMA_State_ind (State) /State == SLAVE => ignore	SLAVE
15	SLAVE	BMA_State_ind (State) /State == MASTER => State := SINGLE_MASTER SequenceID++ StartTimer (SyncTimer, SyncInterval) StartTimer (TimeoutT, SyncTimeOut) StopSlave_req () for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	MASTER
16	MASTER	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status)	MASTER

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
17	MASTER	StopMaster_req () => Status := OK StopTimer (SyncTimer) StopTimer (TimeoutT) BMA_Stop_req () StopMaster_cnf (Status)	IDLE
18	MASTER	BMA_State_ind (State) /State == SLAVE => StopTimer (SyncTimer) StopTimer (TimeoutT) StartSlave_req ()	SLAVE
19	MASTER	BMA_State_ind (State) /State == MASTER => ignore	MASTER
20	MASTER	Sync_ind () /RECEIVE_PORT_VALID => ignore	MASTER
21	MASTER	Sync_ind () /RECEIVE_PORT_VALID && !NEW_SYNC_FRAME => ignore	MASTER
22	MASTER	Sync_ind () /RECEIVE_PORT_VALID && NEW_SYNC_FRAME => MultipleMaster := TRUE	MASTER
23	MASTER	Sync_cnf () => ignore	MASTER
24	MASTER	TimerExpired (SyncTimer) => SequenceID++ StartTimer (SyncTimer, SyncInterval) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	MASTER
25	MASTER	TimerExpired (TimeoutT) /!MultipleMaster => StartTimer (TimeoutT, SyncTimeOut)	MASTER
26	MASTER	TimerExpired (TimeoutT) /MultipleMaster => MultipleMaster := FALSE StartTimer (TimeoutT, SyncTimeOut) SyncStateChange_ind (MULTIPLE_MASTER)	M_MASTER
27	M_MASTER	StartMaster_req () => Status := WRONG_SEQUENCE StartMaster_cnf (Status)	M_MASTER

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
28	M_MASTER	StopMaster_req () => Status := OK StopTimer (SyncTimer) StopTimer (TimeoutT) BMA_Stop_req () StopMaster_cnf (Status)	IDLE
29	M_MASTER	BMA_State_ind (State) /State == SLAVE => StopTimer (SyncTimer) StopTimer (TimeoutT) StartSlave_req ()	SLAVE
30	M_MASTER	BMA_State_ind (State) /State == MASTER => ignore	M_MASTER
31	M_MASTER	Sync_ind () /RECEIVE_PORT_VALID => ignore	M_MASTER
32	M_MASTER	Sync_ind () /RECEIVE_PORT_VALID && !NEW_SYNC_FRAME => ignore	M_MASTER
33	M_MASTER	Sync_ind () /RECEIVE_PORT_VALID && NEW_SYNC_FRAME => MultipleMaster := TRUE	M_MASTER
34	M_MASTER	Sync_cnf () => ignore	M_MASTER
35	M_MASTER	TimerExpired (SyncTimer) => SequenceID++ StartTimer (SyncTimer, SyncInterval) for i := all op. D_Ports do if (TRANSMIT_PORT_VALID) Sync_req (i, PTCP_SYNC_PDU)	M_MASTER
36	M_MASTER	TimerExpired (TimeoutT) /!MultipleMaster => StartTimer (TimeoutT, SyncTimeOut) SyncStateChange_ind (SINGLE_MASTER)	MASTER
37	M_MASTER	TimerExpired (TimeoutT) /MultipleMaster => MultipleMaster := FALSE StartTimer (TimeoutT, SyncTimeOut)	M_MASTER

4.4.4.5.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 141 contient les fonctions, macros, temporisateurs et variables utilisés par le SYN_MPSM ainsi que leurs arguments et leurs descriptions.

Tableau 141 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYN_MPSM

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction locale lance ou relance un temporisateur
StopTimer	Fonction	Cette fonction locale arrête un temporisateur
TimerExpired	Fonction	Cette fonction locale est émise si un temporisateur expire
NEW_SYNC_FRAME	Macro	Vérifie si la trame PTCP_SYNC est valide, que le numéro de séquence est le plus récent et issu de la MasterMACAddress stockée.
RECEIVE_PORT_VALID	Macro	Le Tableau 151 doit être appliqué.
Sync_cnf (+/-) (D_Port)	Macro	LMPM_P_Data.cnf (CREP, D_Port, TStamp, LMPM_status) Affectations: Status := LMPM_status
Sync_ind (S_Port, PTCP_SYNC_PDU)	Macro	Recevoir PTCP-PDU conformément à PTCP_SYNC_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Affectations: PTCP_SYNC_PDU := A_SDU without LT and FrameID
Sync_req (D_Port, PTCP_SYNC_PDU)	Macro	Créer PTCP-PDU conformément à PTCP_SYNC_PDU Affectations: D_Port := D_Port DA := Multicast Address for PTCP_SYNC_PDU and corresponding SyncID SA := port source address PTCP_SYNC_PDU.SequenceID := SequenceID PTCP_SYNC_PDU.DomainUUID := DomainUUID PTCP_SYNC_PDU.MasterSourceAddress := local master address PTCP_SYNC_PDU.Time := local time from source PTCP_SYNC_PDU.TimeExtension := CurrentUTCOffset := Current UTC Offset PTCP_SYNC_PDU.MasterPriority1 := Priority PTCP_SYNC_PDU.ClockClass := Class PTCP_SYNC_PDU.ClockAccuracy := Accuracy PTCP_SYNC_PDU.ClockVariance := Variance PTCP_SYNC_PDU.MasterPriority2 := 0xFF A_SDU := LT, FrameID, PTCP_SYNC_PDU LMPM_P_Data.req (CREP, D_Port, Tstamp, DA, SA, A_SDU)
TRANSMIT_PORT_VALID	Macro	Le Tableau 152 doit être appliqué.

4.4.4.6 Machine de Protocole d'Esclave PTCP

4.4.4.6.1 Définitions de primitives

4.4.4.6.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole d'Esclave PTCP (SYN_SPSM) sont décrites dans la définition de service et présentées dans le Tableau 142.

Tableau 142 – Primitives distantes émises ou reçues par SYN_SPSM

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_P_Data.ind	LMPM	SPSM	CREP, S_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.6.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par SYN_SPSM sont décrites dans la définition de service et présentées dans le Tableau 143.

Tableau 143 – Primitives locales émises ou reçues par SYN_SPSM

Primitive	Source	Destination	Paramètres associés	Fonctions
StartSlave_req	MPSM	SPSM	—	Active la fonction d'esclave
StartSlave_cnf	SPSM	MPSM	Status	Confirmation du démarrage d'un esclave
StopSlave_req	MPSM	SPSM	—	Désactive la fonction d'esclave
StopSlave_cnf	SPSM	MPSM	Status	Confirmation de l'arrêt d'un esclave
SyncStateChange_ind	SPSM	FSPM	Status	Indication pour les événements - JITTER_OUT_OF_BOUNDARY NO_SYNC_MESSAGE_RECEIVED
SLAVE_State_ind	SPSM	BMA	Status	Indication pour les événements utilisée par l'algorithme de meilleur maître - MASTER_LOST

4.4.4.6.2 Schéma de transition d'états

Le schéma de transition d'états du SYN_SPSM est présenté à la Figure 45.

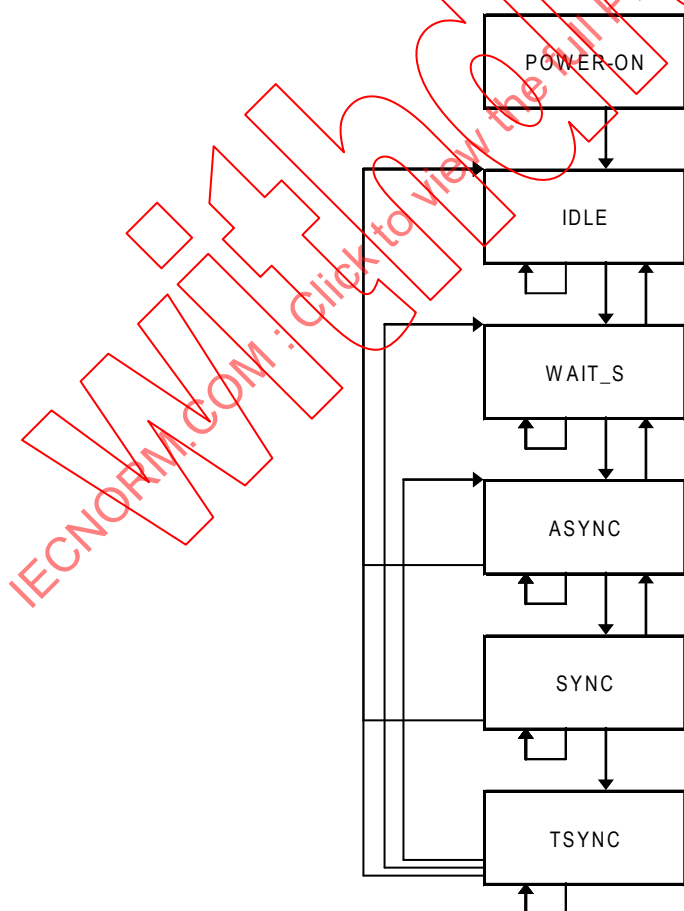


Figure 45 – Schéma de transition d'états de SYN_SPSM

POWER-ON	Initialisation des données
IDLE	La StartSlave_req du service ASE doit lancer la fonction d'esclave de l'appareil.
WAIT_S	L'esclave attend la réception de messages de synchronisation à l'état WAIT_S.
ASYN	L'esclave n'est pas synchronisé avec le maître actif du domaine de synchronisation.
SYNC	L'esclave est synchronisé avec le maître actif du domaine de synchronisation.
TSYN	L'esclave attend toujours les nouveaux messages de synchronisation.

4.4.4.6.3 Description du diagramme d'états

La machine de protocole esclave contrôle la transition d'un appareil esclave de synchronisation. Il existe donc une instance du diagramme d'états pour chaque SyncID avec l'attribut-clé PTCP_SubDomainUUID.

Après le démarrage, la gestion de données doit être initialisée (POWER-ON) et le rôle de synchronisation défini (IDLE). Ce rôle est défini par les services PTCP. Le SPSM passe ensuite de l'état IDLE à l'état WAIT_S. A l'état WAIT_S, l'esclave attend les messages de synchronisation pour synchroniser par rapport à un maître actif. Lors de la réception des messages de synchronisation, le SPSM passe de l'état WAIT_S à l'état ASYN. Si l'esclave est synchronisé, le SPSM doit passer à l'état SYNC. Si aucun message de synchronisation n'est reçu à l'état SYNC, le SPSM doit passer à l'état TSYN.

4.4.4.6.4 Table d'états SYN_SPSM

Le Tableau 144 contient la table d'états utilisée par SYN_SPSM.

Tableau 144 – Table d'états de SYN_SPSM

#	Etat en cours	Événement /Condition =>Action	Etat suivant
1	POWER-ON	=> Stored PTCP_MasterSourceAddress := NIL	IDLE
2	IDLE	StartSlave_req () => store Parameter Status := OK StartTimer (TakeoverTimeoutT, MasterTimeOut) StartSlave_cnf (Status)	WAIT_S
3	IDLE	StopSlave_req () => Status := OK StopSlave_cnf (Status)	IDLE
4	WAIT_S	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	WAIT_S
5	WAIT_S	StopSlave_req () => Status := OK StopTimer (TakeoverTimeoutT) StopSlave_cnf (Status)	IDLE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
6	WAIT_S	Sync_ind () /!VALID_SYNC_FRAME => ignore	WAIT_S
7	WAIT_S	Sync_ind () /VALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time OFFSET_CTRL_START (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval) StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (JITTER_OUT_OF_BOUNDARY)	ASYNC
8	WAIT_S	TimerExpired (TakeoverTimeoutT) => Stored PTCP_MasterSourceAddress := NIL SLAVE_State_ind (MASTER_LOST)	WAIT_S
9	ASYNC	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	ASYNC
10	ASYNC	StopSlave_req () => OFFSET_CTRL_STOP Status := OK StopTimer (OffsetContLoopT) StopTimer (TakeoverTimeoutT) StopTimer (SyncTimeoutT) StopSlave_cnf (Status)	IDLE
11	ASYNC	Sync_ind () /!VALID_SYNC_FRAME => ignore	ASYNC
12	ASYNC	Sync_ind () /VALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (INFO, M_Time, L_Time)	ASYNC
13	ASYNC	TimerExpired (OffsetContLoopT) /!OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval)	ASYNC
14	ASYNC	TimerExpired (OffsetContLoopT) /OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval) SyncStateChange_ind (SYNC)	SYNC
15	ASYNC	TimerExpired (TakeoverTimeoutT) => Stored PTCP_MasterSourceAddress := NIL StopTimer (SyncTimeoutT) SLAVE_State_ind (MASTER_LOST)	WAIT_S

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
16	SYNC	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	SYNC
17	SYNC	StopSlave_req () => OFFSET_CTRL_STOP Status := OK StopTimer (OffsetContLoopT) StopTimer (TakeoverTimeoutT) StopTimer (SyncTimeoutT) StopSlave_cnf (Status)	IDLE
18	SYNC	Sync_ind () //INVALID_SYNC_FRAME => ignore	SYNC
19	SYNC	Sync_ind () //INVALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCIP_SYNC_DATA.Time StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (INFO, M_Time, L_Time)	SYNC
20	SYNC	TimerExpired (OffsetContLoopT) //OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval)	SYNC
21	SYNC	TimerExpired (OffsetContLoopT) //OFFSET_CTRL_IS_SYNC => OFFSET_CTRL_CALC (L_Time, M_Time) StartTimer (OffsetContLoopT, SyncInterval) SyncStateChange_ind (JITTER_OUT_OF_BOUNDARY)	ASYN
22	SYNC	TimerExpired (TakeoverTimeoutT) => Stored PTCIP_MasterSourceAddress := NIL SLAVE_State_ind (MASTER_LOST)	TSYN
23	TSYN	StartSlave_req () => Status := WRONG_SEQUENCE StartSlave_cnf (Status)	TSYN
24	TSYN	StopSlave_req () => OFFSET_CTRL_STOP Status := OK StopTimer (OffsetContLoopT) StopTimer (TakeoverTimeoutT) StopTimer (SyncTimeoutT) StopSlave_cnf (Status)	IDLE
25	TSYN	Sync_ind () //INVALID_SYNC_FRAME => ignore	TSYN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
26	TSYNC	Sync_ind () /VALID_SYNC_FRAME => L_Time := LocalTime M_Time := PTCP_SYNC_DATA.Time StartTimer (TakeoverTimeoutT, MasterTimeOut) StartTimer (SyncTimeoutT, SyncTimeOut) SyncStateChange_ind (INFO, M_Time, L_Time)	ASync
27	TSYNC	TimerExpired (OffsetContLoopT) => StartTimer (OffsetContLoopT, SyncInterval) //NOTE Il convient que la boucle de commande essaie de maintenir l'horloge stable	TSYNC
28	TSYNC	TimerExpired (SyncTimeoutT) => OFFSET_CTRL_STOP StopTimer (OffsetContLoopT) SyncStateChange_ind (NO_SYNC_MESSAGE_RECEIVED)	WAIT_S

4.4.4.6.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 145 contient les fonctions, macros, temporisateurs et variables utilisés par le SYN_SPSM ainsi que leurs arguments et leurs descriptions.

Tableau 145 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYN_SPSM

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction locale lance ou relance un temporisateur
StopTimer	Fonction	Cette fonction locale arrête un temporisateur
TimerExpired	Fonction	Cette fonction locale est émise si un temporisateur expire
OFFSET_CTRL_CALC	Macro	Cette macro locale effectue une compensation de l'écart Paramètres: - LocalTime - MasterTime
OFFSET_CTRL_IS_SYNC	Macro	Cette macro locale mesurait l'erreur de synchronisation dans PLL Window //NOTE La détection d'un défaut dépend de l'algorithme de filtre de la boucle de commande sélectionnée étant donné que la décision est prise uniquement à partir des informations locales.
OFFSET_CTRL_START	Macro	Cette macro locale initialise la boucle de commande pour la compensation de l'écart Paramètres: - LocalTime - MasterTime
OFFSET_CTRL_STOP	Macro	Cette macro locale arrête la compensation de l'écart

Nom	Type	Fonction/Signification
Sync_ind	Macro	Recevoir PTCP-PDU conformément à PTCP_SYNC_PDU LMPM_P_Data.ind (CREP, S_Port, TStamp, DA, SA, A_SDU) if A_SDU.FrameID == WITHOUT_FUP_PDU PTCP_SYNC_DATA:= PTCP_SYNC_PDU else wait for valid PTCP_FUP_PDU from active Master LMPM_P_Data.ind (CREP, S_Port, DA, SA, A_SDU) PTCP_SYNC_DATA.Delay += PTCP_FUP_PDU.Delay Affectations: PTCP_SYNC_PDU:= A_SDU without LT and FrameID PTCP_FUP_PDU:= A_SDU without LT and FrameID Paramètres: - SourcePort - LocalTime - PTCP_SYNC_DATA
VALID_SYNC_FRAME	Macro	Voir Tableau 146
OffsetContLoopT	Temporisateur	Ce temporisateur local est utilisé pour la vérification de l'écart entre LocalTime et MasterTime. Il doit être aligné avec SyncInterval pour de meilleurs résultats de la boucle de commande.
SyncTimeoutT	Temporisateur	Ce temporisateur local est utilisé pour surveiller l'existence d'un maître de synchronisation. Si cette heure expire, le maître de synchronisation est perdu.
TakeoverTimeoutT	Temporisateur	Ce temporisateur local est utilisé pour surveiller l'existence d'un maître de synchronisation. Si cette heure expire, la préparation d'un basculement de maître est effectuée. Cette heure doit toujours être inférieure à SyncTimeoutT.
MasterTimeout	Variable	Cette variable locale contient la valeur de la temporisation du maître au niveau avertissement / préparation. Valeur par défaut: 3 x SyncInterval //NOTE Cette valeur est issue de PTCPTakeoverTimeoutFactor.
PTCP_MasterSourceAddress	Variable	Cette variable partagée contient la MasterSourceAddress du SyncMaster réel. Elle est en outre modifiée par SYNC_RELAY.
Status	Variable	Cette variable locale est utilisée pour stocker le statut pour une utilisation ultérieure.
SyncInterval	Variable	Cette variable locale contient la valeur de l'intervalle recevant l'indication de synchronisation. Valeur par défaut: SyncInterval //NOTE Cette valeur est issue de SyncSendFactor.
SyncTimeout	Variable	Cette variable locale contient la temporisation du maître à sa valeur par défaut. Valeur par défaut: 6 x SyncInterval //NOTE Cette valeur est issue de PTCPTimeoutFactor.

4.4.4.6.6 Table de vérité des règles de réception de la synchronisation pour un SyncID

Le Tableau 146 contient la table de vérité utilisée par le SPSM.

Tableau 146 – Table de vérité pour un SyncID pour la réception de trames de synchronisation et de suivi

Entrée	Sortie
Condition du port de réception ^b	Local

Port-State ^d	MAU-Type ^f	Line-Delay	Ingress Filter	Sub-domain-UUID	Master-MAC-Address	PTCP_Sequence ID	PTCP_Delay1ns / PTCP_Delay10ns	Trame du port valide
Désactivé	—	—	—	—	—	—	—	Non
!Disabled	Incorrect	—	—	—	—	—	—	Non
!Disabled	Ok	Non	—	—	—	—	—	Non
!Disabled	Ok	Oui	Oui	—	—	—	—	Non
!Disabled	Ok	Oui	Non	Incorrect	—	—	—	Non
!Disabled	Ok	Oui	Non	Egal ^a	Incorrect	—	—	Non
!Disabled	Ok	Oui	Non	Egal ^a	Egal ^a	Ancien	—	Non
!Disabled	Ok	Oui	Non	Egal ^a	Egal ^a	Nouveau ^c	Non valide ^e	Non
!Disabled	Ok	Oui	Non	Egal ^a	Egal ^a	Nouveau ^c	Valide ^e	Oui

^a Si le propre PTCP_SubdomainUUID et/ou la MasterMACAddress est inconnu(e) (codé par zéro), la comparaison avec les valeurs de la trame de synchronisation de PTCP peut retourner "égal" pour optimiser le démarrage de la synchronisation.

^b Au niveau du port de réception, le PTCP_SubdomainUUID local et/ou la MasterMACAddress sont/est vérifié(e)(s) par rapport à la trame PTCP.

^c Si la propre MasterMACAddress est inconnu (codée par zéro), une trame de synchronisation de PTCP avec le PTCP_SubdomainUUID local défini comme "nouveau". Le PTCP_SequenceID est rattaché à la MasterMACAddress.

^d Conformément à l'IEEE 802.3 les PortStates sont Disabled, Discarding, Learning et Forwarding. Seul le PortState Disabled ne permet aucun transfert ni aucune réception de trames.

^e Voir la définition des champs PTCP_Delay10ns et PTCP_Delay1ns pour plus de détails.

^f MAUType est en mode simultané et une liaison existe.

4.4.4.7 Machine de Protocole de Relais de Synchronisation

4.4.4.7.1 Définitions de primitives

4.4.4.7.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Relais de Synchronisation (SYNC_RELAY) sont décrites dans la définition de service et présentées dans le Tableau 147.

Tableau 147 – Primitives distantes émises ou reçues par SYNC_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_P_Data.req ()	SYNC_RELAY	LMPM	CREP, D_Port, Tstamp, DA, SA, A_SDU	—
LMPM_P_Data.cnf ()	LMPM	SYNC_RELAY	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_P_Data.ind ()	LMPM	SYNC_RELAY	CREP, D_Port, Tstamp, DA, SA, A_SDU	—

4.4.4.7.1.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par SYNC_RELAY sont décrites dans la définition de service et présentées dans le Tableau 148.

Tableau 148 – Primitives locales émises ou reçues par SYNC_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.4.4.7.2 Schéma de transition d'états

Le schéma de transition d'états du SYNC_RELAY est présenté à la Figure 46.

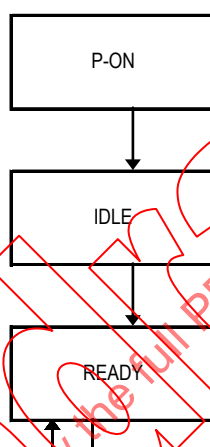


Figure 46 – Schéma de transition d'états de SYNC_RELAY

Etats du SYNC_RELAY

P-ON

Initialisation de la base de données de filtrage des trames de synchronisation.

IDLE

Démarrage du temporisateur de réception pour le vieillissement des entrées de la base de données de filtrage.

PRET

L'état READY décrit les règles de transfert des messages de synchronisation et de suivi.

4.4.4.7.3 Description du diagramme d'états

La machine de protocole de SYNC_RELAY décrit les règles de transfert pour la synchronisation et le suivi des ponts.

Chaque unité d'horodatage du pont doit corriger le champ delay des trames de synchronisation à la volée. La correction temporelle (délai de pontage et retard de ligne) est assurée par la transmission d'une trame de synchronisation.

La machine de protocole de SYNC_RELAY permet d'éviter la circulation des messages de synchronisation dans le réseau.

Pour une grande précision, il convient que le délai de pontage des trames de synchronisation et de suivi soit petit et stable.

Les trames d'annonce doivent être prises en charge par MAC_RELAY. En plus des règles MAC_RELAY, il convient qu'une frontière implicite d'entrée et de sortie soit active pour chaque port qui ne prend pas en charge la mesure de retard de ligne.

4.4.4.7.4 Table d'états SYNC_RELAY

Le Tableau 149 contient la table d'états utilisée par SYNC_RELAY. Seules les trames PTCP provenant d'un maître et un PTCP_SubdomainUUID correspondant à un SyncID doivent être transférées. Une fois le maître choisi, les autres trames PTCP doivent être supprimées. Le Tableau 151 et le Tableau 152 montrent les conditions en un coup d'œil.

Tableau 149 – Table d'états de SYNC_RELAY

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P-ON	=> Init RelayTable	IDLE
2	IDLE	=> StartTimer (PTCP_Timeout) StartTimer (PTCP_TakeoverTimeout)	READY
3	READY	TimerExpired (PTCP_Timeout) => AGING_OF_FWSM_LIST(PT) StartTimer (PTCP_Timeout)	READY
4	READY	TimerExpired (PTCP_TakeoverTimeout) => AGING_OF_FWSM_LIST(PTT) StartTimer (PTCP_TakeoverTimeout)	READY
5	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID => ignore	READY
6	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID && NEW_SYNC_FRAME => ignore	READY
7	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID && NEW_SYNC_FRAME && RATE_VALID => UPDATE_MASTER_ENTRY CalculateMasterRCF	READY
8	READY	Sync_Ind (S_Port, PTCP_SYNC_PDU) /RECEIVE_PORT_VALID && NEW_SYNC_FRAME && RATE_VALID => UPDATE_MASTER_ENTRY CalculateMasterRCF for i := all op. D_Ports \ S_Port do if (TRANSMIT_PORT_VALID) Sync_Req (i, PTCP_SYNC_PDU) //NOTE Si le matériel ne prend pas en charge la demande de synchronisation mais seulement celle de transfert, alors une trame de synchronisation et une trame de suivi sont acheminées.	READY
9	READY	Sync_Cnf () => ignore	READY

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
10	READY	Followup_Ind (S_Port, PTCP_FUP_PDU) /RECEIVE_PORT_VALID => ignore	READY
11	READY	Followup_Ind (S_Port, PTCP_FUP_PDU) /RECEIVE_PORT_VALID && !INVALID_FU_FRAME => ignore	READY
12	READY	Followup_Ind (S_Port, PTCP_FUP_PDU) /RECEIVE_PORT_VALID && VALID_FU_FRAME && RATE_VALID => SET_FU_RECEIVED for i := all op. D_Ports \ S_Port do if (TRANSMIT_PORT_VALID) Followup_Req (i, PTCP_FUP_PDU)	READY
13	READY	Followup_Cnf () => ignore	READY

4.4.4.7.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 150 contient les fonctions, macros, temporisateurs et variables utilisés par le SYNC_RELAY ainsi que leurs arguments et leurs descriptions.

Tableau 150 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SYNC_RELAY

Nom	Type	Fonction/Signification
CalculateMasterRCF	Fonction	Cette fonction locale calcule et met à jour la vitesse (RCF_SyncMaster)
AGING_OF_FWSM_LIST	Macro	Cette macro locale doit prendre en charge l'entrée dans RelayTable. Dans des conditions normales, aucune entrée ou une seule existe. PTCP_Timeout expiré (PT): Supprimer la MasterMACAddress stockée pour activer la réception d'une trame de synchronisation de la part d'un maître primaire/secondaire PTCP_TakeoverTimeout expiré (PTT): Supprimer la MasterMACAddress stockée, le PTCP_SubdomainUUID stocké (si pas défini localement) et mettre la vitesse comme non valide RelayTableEntry[SyncID, DomainUUID]: - MasterAddress - SequenceID - RxPort - FuFrame
Followup_Cnf	Macro	LMPM_P_Data.cnf (CREP, D_Port, TStamp, LMPM_status) Affectations: Status:= LMPM_status Paramètres: - DestinationPort - Status

Nom	Type	Fonction/Signification
Followup_Ind	Macro	Recevoir PTCP-PDU conformément à PTCP_FUP_PDU LMPM_P_Data.ind (CREP, S_Port, TStamp, DA, SA, A_SDU) Affectations: PTCP_FUP_PDU:= A_SDU without LT and FrameID Paramètres: - SourcePort - PTCP_FUP_PDU
Followup_Req	Macro	Créer PTCP-PDU conformément à PTCP_FUP_PDU Affectations: D_Port:= Port DA:= multicast address fo PTCP_FUP_PDU SA:= local port address A_SDU:= LT, FrameID, PTCP_FUP_PDU LMPM_P_Data.req (CREP, D_Port, TStamp, DA, SA, A_SDU) Paramètres: - DestinationPort - PTCP_FUP_PDU
NEW_SYNC_FRAME	Macro	Cette macro locale vérifie en tant que seconde vérification après RECEIVE_PORT_VALID si la trame PTCP_SYNC est valide. Elle vérifie la sémantique et les valeurs des champs delay.
RATE_VALID	Macro	Cette macro locale vérifie si la vitesse (RCFSyncMaster) est calculée et valide dans la plage d'une horloge de $\pm 100 \frac{\mu\text{Hz}}{\text{Hz}}$ ($\pm 250 \frac{\mu\text{Hz}}{\text{Hz}}$)
RECEIVE_PORT_VALID	Macro	Le Tableau 151 doit être appliqué.
SET_FU_RECEIVED	Macro	Met à jour l'entrée dans RelayTable[SyncID, DomainUUID] FuFrame:= TRUE
Sync_Cnf	Macro	LMPM_P_Data.cnf (CREP, D_Port, TStamp, LMPM_status) Affectations: Status:= LMPM_status Paramètres: - DestinationPort - Status
Sync_Ind	Macro	Recevoir PTCP-PDU conformément à PTCP_SYNC_PDU LMPM_P_Data.ind (CREP, S_Port, Tstamp, DA, SA, A_SDU) Affectations: PTCP_SYNC_PDU:= A_SDU without LT and FrameID Paramètres: - SourcePort - PTCP_SYNC_PDU
Sync_Req	Macro	Créer PTCP-PDU conformément à PTCP_SYNC_PDU Affectations: D_Port:= Port DA:= multicast address fo PTCP_SYNC_PDU SA:= local port address A_SDU:= LT, FrameID, PTCP_SYNC_PDU LMPM_P_Data.req (CREP, D_Port, TStamp, DA, SA, A_SDU) Paramètres: - DestinationPort - PTCP_SYNC_PDU
TRANSMIT_PORT_VALID	Macro	Le Tableau 152 doit être appliqué.
UPDATE_MASTER_ENTRY	Macro	Met à jour l'entrée dans RelayTable[SyncID, DomainUUID] pour le maître de synchronisation MasterAddress:= PTCP_SYNC_PDU.MasterSourceAddress SequenceID:= PTCP_SYNC_PDU.SequenceID RxPort:= S_Port FuFrame:= FALSE
VALID_FU_FRAME	Macro	Vérifie RelayTable[SyncID, DomainUUID] et le port de réception et si le numéro de séquence est valide MasterAddress == PTCP_FUP_PDU.MasterSourceAddress SequenceID == PTCP_FUP_PDU.SequenceID RxPort == S_Port FuFrame == FALSE

Nom	Type	Fonction/Signification
PTCP_TakeoverTimeout	Temporisateur	Cette variable locale contient la valeur de la temporisation du maître au niveau avertissement / préparation. Valeur par défaut: 3 x SyncInterval //NOTE Cette valeur est issue de PTCPTakeoverTimeoutFactor.
PTCP_Timeout	Temporisateur	Cette variable locale contient la temporisation du maître à sa valeur par défaut. Valeur par défaut: 6 x SyncInterval //NOTE Cette valeur est issue de PTCPTimeoutFactor.

4.4.4.7.6 Table de vérité des règles de transfert de la synchronisation pour les ponts pour un SyncID

Le Tableau 151 et le Tableau 152 contiennent la table de vérité utilisée par le SYNC_RELAY.

Tableau 151 – Table de vérité pour un SyncID pour la réception

Entrée						Sortie
Condition du port de réception ^b						Local
PortState ^c	MAUType	LineDelay	Ingress-Filter	Subdomain-UUID	MasterMACAddress	Trame du port valide
Désactivé	—	—	—	—	—	Non
!Disabled	Incorrect	—	—	—	—	Non
!Disabled	Ok	Non	—	—	—	Non
!Disabled	Ok	Oui	Oui	—	—	Non
!Disabled	Ok	Oui	Non	Incorrect	—	Non
!Disabled	Ok	Oui	Non	Egal ^a	Incorrect	Non
!Disabled	Ok	Oui	Non	Egal ^a	Egal ^a	Oui ^d

^a Si le propre PTCP_SubdomainUUID et/ou la MasterMACAddress est inconnu(e) (codé par zéro), la comparaison avec les valeurs de la trame de synchronisation de PTCP peut retourner "égal" pour optimiser le démarrage de la synchronisation.

^b Au niveau du port de réception, le PTCP_SubdomainUUID local et/ou la MasterMACAddress sont/est vérifié(e)(s) par rapport à la trame PTCP.

^c Selon l'IEEE 802.3, les états PortState sont Disabled, Discarding, Learning et Forwarding. Seul le PortState Disabled ne permet aucun transfert ni aucune réception de trames.

^d Il convient qu'un filtre facultatif soit utilisé pour écarter les trames dont le contenu des champs PTCP_Delay1ns / PTCP_Delay10ns n'est pas valide.

Tableau 152 – Table de vérité pour un SyncID pour la transmission

Entrée							Sortie
Local	Conditions d'un port de transmission ^b						Local
Vitesse connue	Subdomain-UUID	MasterMAC-Address	PortState	MAU-Type	Line-Delay	Egress-Filter	Port pour trame valide
Non	—	—	—	—	—	—	Non
Oui	Incorrect	—	—	—	—	—	Non
Oui	Egal ^a	Incorrect	—	—	—	—	Non
Oui	Egal ^a	Egal ^a	Discarding	—	—	—	Non
Oui	Egal ^a	Egal ^a	!Discarding	Incorrect	—	—	Non
Oui	Egal ^a	Egal ^a	!Discarding	Ok	Non	—	Non
Oui	Egal ^a	Egal ^a	!Discarding	Ok	Oui	Oui	Non
Oui	Egal ^a	Egal ^a	!Discarding	Ok	Oui	Non	Oui

^a Si le propre PTCP_SubdomainUUID et/ou la MasterMACAddress est inconnu(e) (codé par zéro), la comparaison avec les valeurs de la trame de PTCP peut retourner "égal" pour optimiser le démarrage de la synchronisation.

^b Au port de transmission, le PTCP_SubdomainUUID distant et/ou la MasterMACAddress sont/est vérifié(e) par rapport à la trame PTCP.

4.4.4.8 Scheduler Protocol Machine (Machine de protocole de programmeur)

4.4.4.8.1 Définitions de primitives

4.4.4.8.1.1 Primitives échangées entre les machines distantes

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par SCHEDULER (PROGRAMMATEUR) sont décrites dans la définition de service et présentées dans le Tableau 153.

Tableau 153 – Primitives distantes émises ou reçues par SCHEDULER

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_C_Data.req	SCHEDULER (PROGRAMMATEUR)	LMPM	CREP, PortList, DA, SA, Prio, C_SDU, APDU_Status	—
UDP_C_Data.req	SCHEDULER (PROGRAMMATEUR)	IP	IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status	—
LMPM_C_Data.cnf	LMPM	SCHEDULER (PROGRAMMATEUR)	CREP, PortList, DA, SA, Prio, C_SDU, APDU_Status	—

Primitive	Source	Destination	Paramètres associés	Fonctions
UDP_C_Data.cnf	IP	SCHEDULER (PROGRAMMATEUR)	IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status	—

4.4.4.8.1.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par SCHEDULER (PROGRAMMATEUR) sont décrites dans la définition de service et présentées dans le Tableau 154.

Tableau 154 – Primitives locales émises ou reçues par SCHEDULER

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_Schedule.cnf	DEMUX	SCHEDULER (PROGRAMMATEUR)	—	—
MUX_Schedule.cnf	MUX	SCHEDULER (PROGRAMMATEUR)	—	—
DMUX_Schedule_req	SCHEDULER (PROGRAMMATEUR)	DEMUX	CycleCounter, Rx_Period	—
DMUX_Schedule_req	SCHEDULER (PROGRAMMATEUR)	DEMUX	Rx_Period	—
MUX_Schedule_req	SCHEDULER (PROGRAMMATEUR)	MUX	CycleCounter, Tx_Period	—
MUX_Schedule_req	SCHEDULER (PROGRAMMATEUR)	MUX	Tx_Period	—
SCHEDULER_add_req	PPM	SCHEDULER (PROGRAMMATEUR)	CREP, ReductionRatio, Phase, Sequence, TxOptions	Ajoute une PPM au programmeur
SCHEDULER_remove_req	PPM	SCHEDULER (PROGRAMMATEUR)	CREP	Enlève une PPM du programmeur
SCHEDULER_error_ind	SCHEDULER (PROGRAMMATEUR)	CMSU CTLSU	CREP, ErrorCode	Signale une surcharge. Dans ce cas, la phase SUIVANTE doit être sautée
SCHEDULER_add.cnf	SCHEDULER (PROGRAMMATEUR)	PPM	CREP	Ajoute une PPM au programmeur
SCHEDULER_remove.cnf	SCHEDULER (PROGRAMMATEUR)	PPM	CREP	Enlève une PPM du programmeur

4.4.4.8.2 Schéma de transition d'états

Le schéma de transition d'états de l'ALPMI est présenté à la Figure 47.

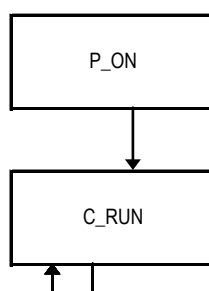


Figure 47 – Schéma de transition d'états de SCHEDULER

Etats du SCHEDULER

P_ON	Initialisation des données
C_RUN	Le programmeur est actif et fonctionne

4.4.4.8.3 Description du diagramme d'états

Le SCHEDULER génère les signaux pour la gestion locale des événements temporels. Le TickEvent est en outre utilisé pour déclencher le transfert en temps voulu des trames RT_CLASS_3.

4.4.4.8.4 Table d'états de SCHEDULER

Le Tableau 155 contient la table d'états utilisée par SCHEDULER.

Tableau 155 – Table d'états SCHEDULER

#	Etat en cours	Événement /Condition =>Action	Etat suivant
1	P_ON	Scheduler_Init_req () => TxPeriod := GREEN RxPeriod := GREEN MUX_Schedule_req(TxPeriod) DMUX_Schedule_req(RxPeriod)	C_RUN
2	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == GREEN && TxPeriodChecker () == GREEN => ignore	C_RUN
3	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == RED && TxPeriodChecker () == GREEN => TxPeriod := GREEN MUX_Schedule_req(CycleCounter, TxPeriod)	C_RUN
4	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == RED && TxPeriodChecker () == RED => ignore	C_RUN
5	C_RUN	TickEvent (CycleCounter, CycleOffset) /TxPeriod == GREEN && TxPeriodChecker () == RED	C_RUN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
		=> TxPeriod := RED MUX_Schedule_req(CycleCounter, TxPeriod)	
6	C_RUN	MUX_Schedule_cnf() => ignore	C_RUN
7	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == GREEN && RxPeriodChecker () == GREEN => ignore	C_RUN
8	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == RED && RxPeriodChecker () == GREEN => RxPeriod := GREEN DMUX_Schedule_req(CycleCounter, RxPeriod)	C_RUN
9	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == RED && RxPeriodChecker () == RED => ignore	C_RUN
10	C_RUN	TickEvent (CycleCounter, CycleOffset) /RxPeriod == GREEN && RxPeriodChecker () == RED => RxPeriod := RED DMUX_Schedule_req(CycleCounter, RxPeriod)	C_RUN
11	C_RUN	DMUX_Schedule_cnf() => ignore	C_RUN
12	C_RUN	TickEvent (CycleCounter, CycleOffset) /for each active PPM check whether CycleCounter, CycleOffset fits to FrameSendOffset and ReductionRatio check if PPM ready to schedule and frame not already at the LMPM => C_Data_req()	C_RUN
13	C_RUN	TickEvent (CycleCounter, CycleOffset) /for each active PPM check whether CycleCounter, CycleOffset fits to FrameSendOffset and ReductionRatio check if PPM ready to schedule and frame already at the LMPM => SCHEDULER_error.ind (ErrorCode = Overload)	C_RUN
14	C_RUN	C_Data_cnf() => ignore	C_RUN
15	C_RUN	SCHEDULER_add.req (CREP, ReductionRatio, Phase, Sequence) => Add PPM to list SCHEDULER_add.cnf (CREP)	C_RUN
16	C_RUN	SCHEDULER_remove.req (CREP) => Remove PPM from list SCHEDULER_remove.cnf (CREP)	C_RUN

4.4.4.8.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 156 contient les fonctions, macros, temporisateurs et variables utilisés par le SCHEDULER ainsi que leurs arguments et leurs descriptions.

Tableau 156 – Fonctions, Macros, Temporisateurs et Variables utilisés par le SCHEDULER

Nom	Type	Fonction/Signification
RxPeriodChecker	Fonction	Cette fonction locale vérifie la RxPeriod active et la retourne.
TickEvent	Fonction	Événement généré en local à partir d'un temporisateur. Le temporisateur doit être contrôlé par le maître ou l'esclave PTCP en cas de synchronisation ou d'exécution libre sans synchronisation. Paramètre: - CycleCounter - CycleOffset
TxPeriodChecker	Fonction	Cette fonction locale vérifie la TxPeriod active et la retourne.
C_Data_cnf	Macro	si (txOption == RTC) LMPM_C_Data.cnf (CREP, LMPM_status) si (txOption == UDP) UDP_C_Data.cnf (IPPort, CREP, LMPM_status)
C_Data_req	Macro	si (txOption == RTC) LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) si (txOption == UDP) UDP_C_Data.req (IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status)
RxPeriod	Variable	Cette variable contient les valeurs: ROUGE et VERTE
TxPeriod	Variable	Cette variable contient les valeurs: ROUGE et VERTE

4.4.4.8.6 Tables de vérité

4.4.4.8.6.1 Table de vérité des règles de PeriodChecker

Le Tableau 157 et le Tableau 158 donnent la table de vérité utilisée par le SCHEDULER (programmeur) pour détecter la période nécessaire à chaque port.

Tableau 157 – Table de vérité pour RxPeriodChecker d'un port

Entrée		Sortie
Etat du port RTC3PSM	CycleOffset à l'intérieur d'une période RED	RxPeriod
OFF	—	VERTE
—	Extérieur	VERTE
UP	Intérieur	VERTE
RUN	Intérieur	ROUGE

Tableau 158 – Table de vérité pour TxPeriodChecker d'un port

Entrée		Sortie
Etat du port RTC3PSM	CycleOffset à l'intérieur d'une période RED	TxPeriod
OFF	—	VERTE
—	Extérieur	VERTE

Entrée		Sortie
Etat du port RTC3PSM	CycleOffset à l'intérieur d'une période RED	TxPeriod
UP	Intérieur	ROUGE
RUN	Intérieur	ROUGE

4.4.5 Machines de Protocole de Mapping DLL

Il n'existe aucune Machine de Protocole de Mapping DLL (DMPM) définie pour ce protocole.

4.5 Synchronisation temporelle

L'IEEE 802.1AS s'applique à la synchronisation temporelle. La prise en charge de la Base d'Informations de Gestion IEEE 802.1AS est facultative.

4.6 Redondance de support

4.6.1 Redondance de support et prévention de boucles

Pour la redondance de support, la CEI 62439-2 s'applique. La prise en charge de la Base d'Informations de Gestion MRP (MIB) est facultative.

4.6.2 Redondance de support sans coupure

Pour la redondance de support sans coupure, MRPD s'applique. Ceci doit être effectué à l'aide du PPM, du CPM et du RED_RELAY.

MRDP doit fonctionner uniquement conjointement avec MRP.

4.7 Cyclique en temps réel

4.7.1 Description de la syntaxe FAL

4.7.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.7.1.2 Syntaxe abstraite RTC APDU

Le Tableau 159 définit la syntaxe abstraite des PDU de couche application appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 159 doivent représenter le contenu de la DLSDU dans le Tableau 5.

Tableau 159 – Syntaxe RTC APDU

Nom de l'APDU	Structure de l'APDU
RTC-PDU	C_SDU ^ CSF_SDU, [RTCPadding*] ^{a b} , APDU_Status
UDP-RTC-PDU	IPHeader, UDPHeader, FrameID, RTC-PDU
^a Dans le cas de RTC-PDU, le nombre d'octets de remplissage doit être compris entre 0 et 40, selon la taille Dataltem. Dans le cas d'UDP-RTC-PDU, le nombre d'octets de remplissage doit être compris entre 0 et 12, selon la taille Dataltem. ^b Dans le cas du transport RT_CLASS_3, le nombre d'octets de remplissage est donné par le système d'ingénierie.	

Le Tableau 160 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 159.

Tableau 160 – Substitutions RTC

Nom de substitution	Structure
C_SDU	{[DataItem], [GAP*]}* ^a
DataItem	{[IOCS*], [DataObjectElement]*}* ^a
SubstituteDataItem	{[IOCS*] ^b , [SubstituteDataObjectElement]*}* ^a
DataObjectElement	[Data*], IOPS
SubstituteDataObjectElement	[Data*], SubstituteDataValid
CSF_SDU	SFCRC16, SUBFRAME*, SFEndDelimiter ^a
SUBFRAME	SFPosition, SFDataLength, SFCycleCounter, DataStatus, C_SDU, SFCRC16
SFEndDelimiter	SFPosition(=0), SFDataLength(=0)
APDU_Status	CycleCounter, DataStatus, TransferStatus
^a La taille maximale d'un C_SDU ou CSF_SDU ne doit pas dépasser 1 440 octets.	
^b La valeur doit contenir la même valeur que le champ DataItem.IOCS.	

4.7.2 Syntaxe de transfert FAL

4.7.2.1 Section de codage des champs spécifiques à Data-RTC-PDU

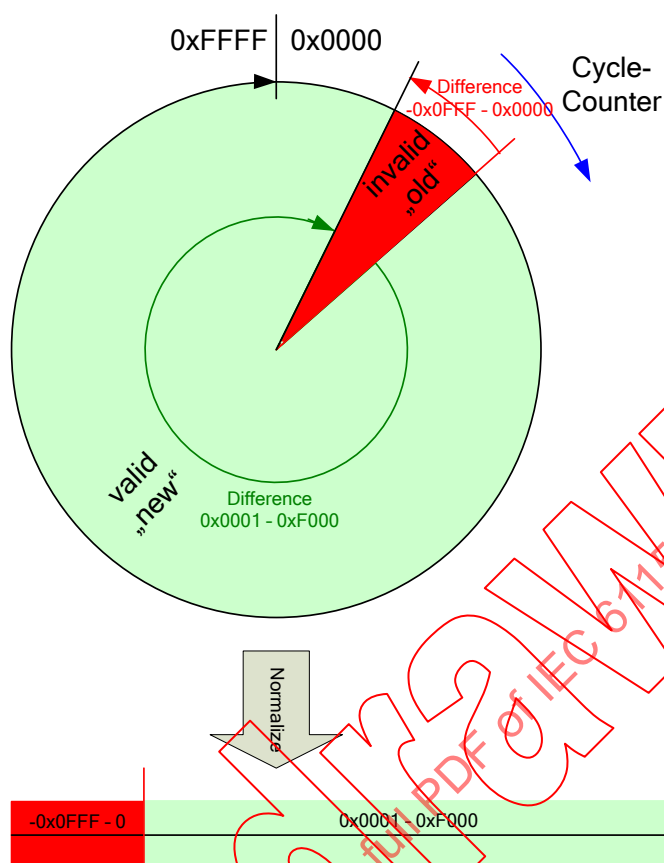
4.7.2.1.1 Vue d'ensemble

Les champs Data et IOxS doivent également être utilisés pour coder les données utilisateur des autres types PDU.

4.7.2.1.2 Codage du champ CycleCounter

Ce champ doit être codé en type de données Unsigned16. Un incrément représente une valeur de temps de 31,25 µs pour RTC-PDU et de 1 ms pour UDP-RTC-PDU. Chaque RTCPDU ou UDP-RTC-PDU contient son CycleCounter créé conformément à la Figure 49 par l'émetteur.

Le destinataire de RTC-PDU ou de UDP-RTC-PDU doit utiliser ce champ pour détecter les répétitions, les pertes de trame et l'opportunité des données. C'est la raison pour laquelle la plage de valeurs CycleCounter doit être divisée selon un rapport de 15/16 en cas de validité, et de 1/16 en cas de non-validité (voir Figure 48).



Légende

Anglais	Français
Difference	Différence
invalid "old"	"ancien" non valide
valid "new"	"nouveau" valide
Normalize	Normalisation

Figure 48 – Plage de valeurs CycleCounter

NOTE La normalisation peut être assurée à l'aide de la formule suivante:

$$\text{CycleCounter(Normalized)} = (((\text{CycleCounter(NEW)} + (0x10000 - \text{CycleCounter(STORED)}) - 1) \& 0xFFFF) - 0xF000) \times (-1).$$

Le Tableau 161 définit un laps de temps maximal entre l'ancienne et la nouvelle trame et l'action du consommateur, selon la différence entre le CycleCounter du dernier RTC-PDU ou UDP-RTC-PDU reçu et la valeur du PDU reçu en cours.

Tableau 161 – Différence CycleCounter

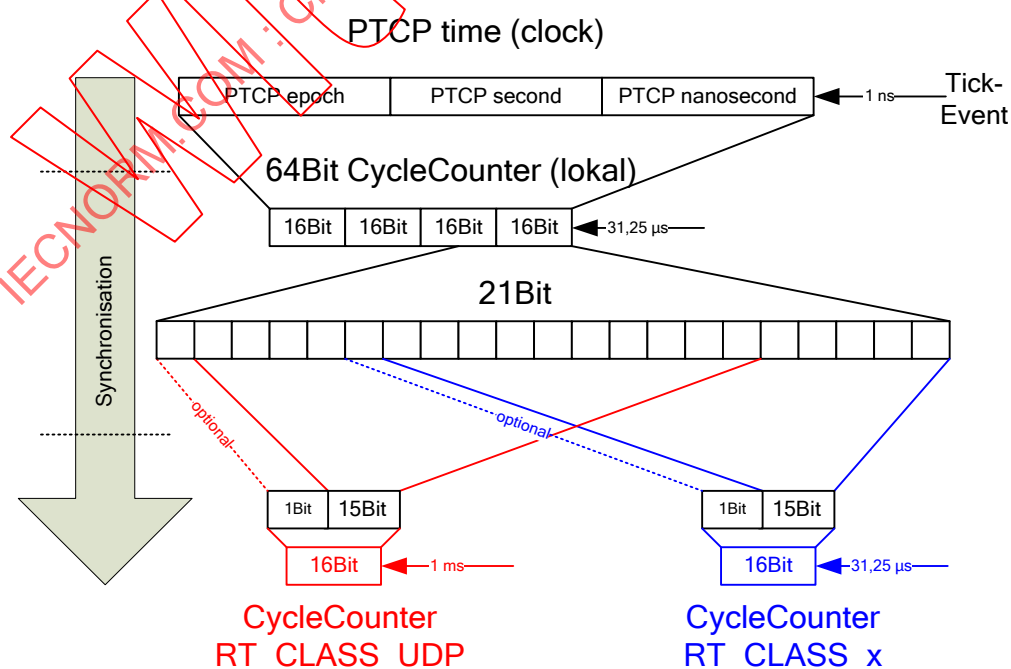
Valeur (hexadécimale)	Plage de temps RTC-PDU [s]	Plage de temps UDP-RTC-PDU [s]	Signification	Utilisation
-0x0FFF – 0x0000	0,128	4,096	Domaine rouge conformément à la Figure 48 1/16 de la plage CycleCounter Le consommateur approuve la trame reçue comme étant plus ancienne que la trame enregistrée	La trame est supprimée
0x0001 – 0xF000	1,92	61,44	Domaine vert conformément à la Figure 48 15/16 de la plage CycleCounter Le consommateur approuve la trame reçue comme étant plus récente que la trame enregistrée	La trame est traitée

Si MRPD est utilisé, le CycleCounter des trames supplémentaires doit être configuré de manière identique par l'émetteur.

Un appareil doit conserver un compteur 64 bits local avec un incrément de 31,25 µs (voir Figure 49).

Les bits 0 à 14 doivent être utilisés de la même manière pour le CycleCounter des RTC-PDU. Le bit 15 du CycleCounter à 16 bits des RTC-PDU peut également être identique au bit 15 du compteur 64 bits local.

Les bits 5 à 19 doivent être utilisés de la même manière que les bits 0 à 14 du CycleCounter des UDP-RTC-PDU. Le bit 15 du CycleCounter à 16 bits des UDP-RTC-PDU peut également être identique au bit 20 du compteur 64 bits local.



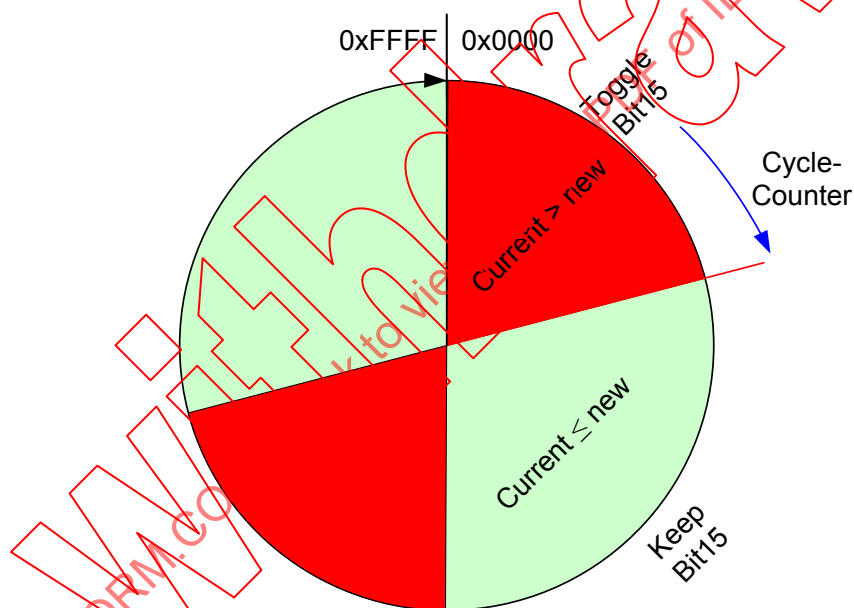
Légende

Anglais	Français
PTCP time (clock)	Heure PTCP (horloge)
PTCP epoch	Numéro epoch PTCP
PTCP second	seconde PTCP
PTCP nanosecond	nanoseconde PTCP
64Bit CycleCounter (local)	CycleCounter 64 bits (local)
Synchronisation	Synchronisation

Figure 49 – Structure de CycleCounter

En cas de synchronisation globale des nœuds, le bit 15 de la base de CycleCounter des RTCPDU et le bit 20 de la base de CycleCounter des UDP-RTC-PDU doivent être mis conformément à la Figure 49 et à la Figure 50 afin d'éviter une détection "plus ancienne" erronée par le destinataire.

Le CycleCounter doit être défini en raison de la synchronisation conformément au modèle présenté à la Figure 50.



Légende

Anglais	Français
Toggle Bit15	Basculer bit 15
Current > new	En cours > nouveau
Current ≤ new	En cours ≤ nouveau
Keep Bit15	Conserver bit 15

Figure 50 – Paramètre CycleCounter optimisé

CycleCounter(en cours)

Cela fait référence à la valeur de CycleCounter utilisée par l'émetteur avant l'exécution de la synchronisation.

CycleCounter(suivant)

Cela fait référence à la valeur de CycleCounter requise par l'émetteur conformément aux informations relatives à la synchronisation

Si CycleCounter(en cours)[0..14] <= CycleCounter(suivant)[0..14]

alors /* en cours ≤ suivant */

REGLER CycleCounter[0..14] avec CycleCounter(next)[0..14] et conserver CycleCounter[15]

sinon /* en cours > suivant */

REGLER CycleCounter[0..14] avec CycleCounter(next)[0..14] et conserver CycleCounter[15]

4.7.2.1.3 Codage du champ DataStatus

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0: DataStatus.State

Ce champ doit être codé avec les valeurs conformes au Tableau 162 et évalué après que l'AR soit à l'état InDATA.

Tableau 162 – DataStatus.State

Valeur (hexadécimale)	Signification
0x00	Etat IOCR de sauvegarde
0x01	Etat IOCR principal

Bit 1: DataStatus.DataValid

Ce champ doit être codé avec les valeurs conformes au Tableau 163 et évalué après que l'AR soit à l'état InDATA.

Tableau 163 – DataStatus.Redundancy

Valeur (hexadécimale)	Signification
0x00	Par défaut Une AR principale d'un ensemble d'AR donné est présente
0x01	Aucune AR principale d'un ensemble d'AR donné n'est présente

NOTE Un appareil de commande E/S met le fanion à la valeur par défaut indépendamment de l'ARType.

Bit 2: DataStatus.DataValid

Ce champ doit être codé avec les valeurs conformes au Tableau 164.

Tableau 164 – DataStatus.DataValid

Valeur (hexadécimale)	Signification
0x00	DataItem non valide
0x01	DataItem valide

Il est recommandé d'utiliser `DataStatus.DataValid == 'DataItem invalid'` pour DFP uniquement avec `IOxS.DataState == 'Bad'` pour tous les `DataItems`.

Bit 3: `DataStatus.Reserved_1`

Ce champ doit être mis à zéro mais ce dernier ne doit pas être vérifié par le destinataire

Bit 4: `DataStatus.ProviderState`

Ce champ doit être codé avec les valeurs conformes au Tableau 165.

Tableau 165 – `DataStatus.ProviderState`

Valeur (hexadécimale)	Signification
0x00	Arrêt
0x01	RUN

Bit 5: `DataStatus.StationProblemIndicator`

Ce champ doit être codé avec les valeurs conformes au Tableau 166.

Tableau 166 – `DataStatus.StationProblemIndicator`

Valeur (hexadécimale)	Signification
0x00	Problème détecté
0x01	Fonctionnement normal

Bit 6: `DataStatus.Reserved_2`

Ce champ doit être mis conformément à 3.4.2.2.

Bit 7: `DataStatus.Ignore`

Ce champ doit être codé avec les valeurs conformes au Tableau 167 et au Tableau 168.

Tableau 167 – `DataStatus.Ignore` d'une trame

Valeur (hexadécimale)	Signification
0x00	Evaluer <code>DataStatus</code>
0x01	Ignorer <code>DataStatus</code> Ne doit être mis en place que pour une trame contenant des sous-trames avec <code>SFIOCRProperties.SFCRC16:= 1</code>

Tableau 168 – `DataStatus.Ignore` d'une sous-trame

Valeur (hexadécimale)	Signification
0x00	Evaluer la sous-trame
0x01	Ignorer la sous-trame; il s'agit d'une sous-trame subrogée. Le CPM charge cette sous-trame comme invalide.

4.7.2.1.4 Codage du champ TransferStatus

Ce champ doit être codé en type de données Unsigned8. Ce champ doit être réglé à zéro pour les trames RT_CLASS_UDP, RT_CLASS_1 et RT_CLASS_2. Pour la RT_CLASS_3, la valeur doit être réglée conformément à Tableau 169.

Tableau 169 – TransferStatus pour RT_CLASS_3

Position du bit	Nom	Valeur	Signification
0	AlignmentOrFrameChecksumError	0	Pas d'erreur de somme de contrôle de trame ni d'erreur d'alignement
		1	Erreur de somme de contrôle de trame ou erreur d'alignement et aucun dépassement de capacité de mémoire tampon de réception MAC ne s'est produit
1	WrongLengthError	0	Pas d'erreur de longueur
		1	Une erreur de longueur s'est produite
2	MACReceiveBufferOverflow	0	Aucun dépassement de capacité de mémoire tampon de réception MAC
		1	Un dépassement de capacité de mémoire tampon de réception MAC s'est produit
3	RT_CLASS_3 Error	0	Aucune erreur RT_CLASS_3
		1	Erreur RT_CLASS_3 Par exemple, la trame n'a pas été reçue à temps, ne correspond pas au type de trame prévu, ne détient l'ID de trame prévu, ne correspond pas à l'adresse MAC source prévue (éventuellement vérifiée) ou en cas de sous-exploitation (le pont a reçu un en-tête, mais les données ne sont pas arrivées à temps, par exemple)
Autre	—	Réserve	Réserve

4.7.2.1.5 Section de codage relative à la périphérie décentralisée

4.7.2.1.5.1 Codage du champ Data

Les types de données définis dans la CEI 61158-5-10, Article 5 doivent être utilisés pour Data.

4.7.2.1.5.2 Section de codage relative à une sous-trame

4.7.2.1.5.2.1 Codage du champ SFCRC16

Ce champ doit être codé en type de données Unsigned16.

Le SFCRC16 est généré par la Formule (38) sous forme d'un polynôme de 16 bits.

$$\text{SFCRC16} = 1 + x^2 + x^{15} + x^{16} \quad (38)$$

où

x est le contenu de l'octet
SFCRC16 polynôme de 16 bits

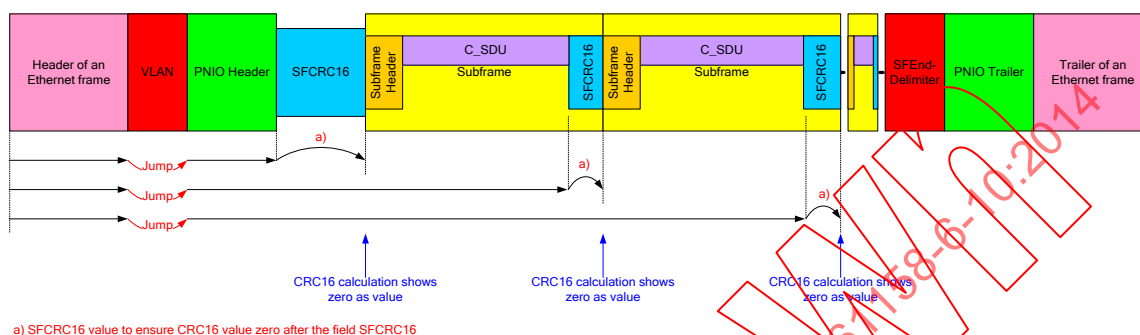
Chaque SFCRC16 commence par la valeur 0 et calcule tous les octets commençant par le premier octet de l'adresse MAC de destination jusqu'au SFCRC16.

La représentation de la valeur en 16 bits dans une trame est petit-boutiste (ce qui garantit que la valeur de CRC est 0 après les octets SFCRC de la sous-trame précédente).

NOTE La conception pourrait être vérifiée avec <http://www.easics.be/webtools/crc7ool>.

La génération du premier SFCRC16 commence par le premier octet de l'adresse MAC de destination, ignore l'étiquette VLAN (et les autres étiquettes) et se termine par le dernier octet de Frame ID.

La génération de la sous-trame SFCRC16, présentée à la Figure 51 commence par le premier octet de la SFPosition et se termine par le dernier octet du champ C_SDU.



Légende

Anglais	Français
Header of an Ethernet frame	En-tête d'une trame Ethernet
VLAN	VLAN
PNIO Header	En-tête PNIO
Subframe Header	En-tête de sous-trame
Subframe	Sous-trame
SFEnd-Delimiter	Délimiteur de fin de sous-trame
PNIO Trailer	Fin de trame PNIO
Trailer of an Ethernet frame	Fin d'une trame Ethernet
Jump	Saut
CRC16 calculation shows zero as value	Calcul CRC16 montre 0 en tant que valeur
SFCRC16 value to ensure CRC16 value zero after the field	Valeur SFCRC16 pour garantir la valeur zéro de CRC16 après le champ
SFCRC16	SFCRC16

Figure 51 – Règle de génération de SFCRC16

4.7.2.1.5.2.2 Codage du champ SFPosition

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0-6: SFPosition.Position

Ce champ doit être codé avec les valeurs conformes au Tableau 170.

Tableau 170 – SFPosition.Position

Valeur (hexadécimale)	Signification
0x00	Fin du codage de la Sous-trame
0x01 – 0x3F	Utilisable pour la mise en paquets et la désassociation des sous-trames lors du transport et de bout en bout

Valeur (hexadécimale)	Signification
0x40 – 0x7F	Utilisable pour la mise en paquets et la désassociation des sous-frames de bout en bout

NOTE "De bout en bout" signifie, que la source des données crée et concatène les sous-frames d'une trame. L'acheminement est effectué sans modification sur la trame. La destination des données désassocie les sous-frames et utilise les données.

Bit 7: SFPosition.Reserved

Ce champ doit être codé avec les valeurs conformes au Tableau 171.

Tableau 171 – SFPosition.Reserved

Valeur (hexadécimale)	Signification
0x00	Réservé

4.7.2.1.5.2.3 Codage du champ SFDataLength

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 172.

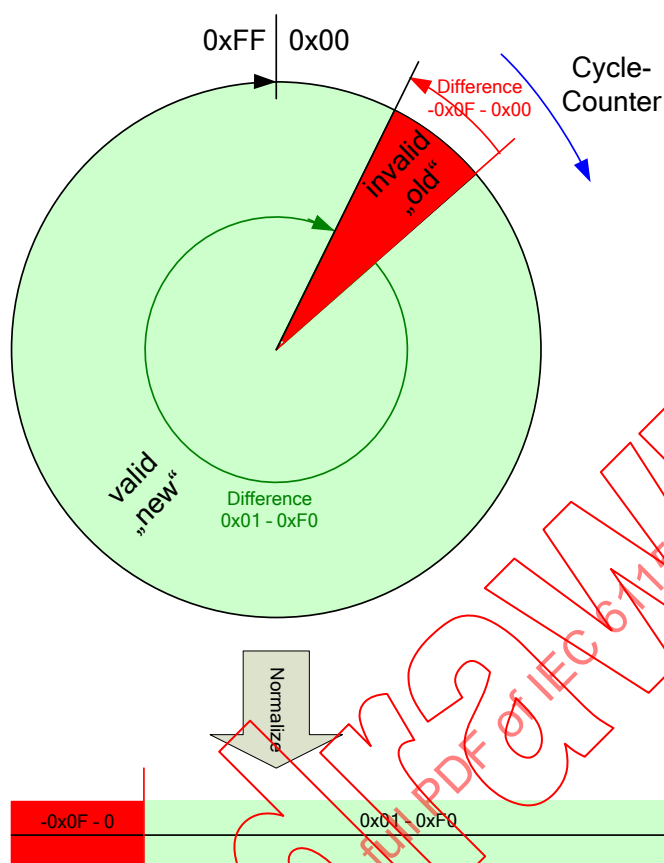
Tableau 172 – SFDataLength

Valeur (hexadécimale)	Signification
0x00	Fin du codage de la Sous-trame
0x01 – 0xFF	Nombre d'octets de la C_SDU

4.7.2.1.5.2.4 Codage du champ SFCycleCounter

Ce champ doit être codé en type de données Unsigned8. Chaque Sous-trame contient son SFCycleCounter incrémenté de un pour chaque transmission à partir du PPM.

Le destinataire de la Sous-trame doit utiliser ce champ pour détecter les répétitions, la perte de Sous-frames et l'opportunité des données. C'est la raison pour laquelle la plage de valeurs CycleCounter doit être divisée selon un rapport de 15/16 en cas de validité et de 1/16 en cas de non-validité (voir Figure 52).



Légende

Anglais	Français
Difference	Différence 0x0F – 0x00
Invalid "old"	"ancien" non valide
Valid "new"	"nouveau" valide
Normalize	Normalisation

Figure 52 – Plage de valeurs SFCycleCounter

Le Tableau 173 définit un laps de temps maximal entre l'ancien et le nouveau SFCycleCounter et l'action du consommateur, selon la différence entre le SFCycleCounter de la dernière Sous-trame reçue et la valeur de la Sous-trame actuellement reçue.

Tableau 173 – Différence entre SFCycleCounter

Valeur (hexadécimale)	Plage de temps RTC-PDU [s]	Plage de temps UDP-RTC-PDU [s]	Signification	Utilisation
-0x0F– 0x00	—	—	Domaine rouge conformément à la Figure 52 1/16 de la plage de SFCycleCounter Le consommateur considère la sous-trame reçue comme plus ancienne que celle stockée	La sous-trame est rejetée

Valeur (hexadécimale)	Plage de temps RTC- PDU [s]	Plage de temps UDP- RTC-PDU [s]	Signification	Utilisation
0x01 – 0xF0	—	—	Domaine verte conformément à la Figure 52 15/16 de la plage de SFCycleCounter Le consommateur considère la sous- trame reçue comme plus neuve que celle stockée	La sous-trame est traitée

NOTE La même valeur SFCycleCounter, qui signifie que la différence est de zéro, est attribuée à la plus ancienne.

Si MRPD est utilisé, le SFCycleCounter des sous-frames supplémentaires doit être configuré comme identique par l'émetteur.

4.7.2.1.5.3 Section de codage relative à IOxS (IOPS, IOCS, SubstituteDataValid)

4.7.2.1.5.3.1 Codage du champ IOPS

Le codage de ces champs doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0: IOxS.Extension

Ce champ doit être codé avec les valeurs conformes au Tableau 174.

Tableau 174 – IOxS.Extension

Valeur (hexadécimale)	Signification
0x00	Aucun octet IOxS ne suit
0x01	Un octet IOxS supplémentaire suit

Bit 1 à 4: IOxS.reserved

Ce champ doit être mis conformément à 3.4.2.2.

Bit 5 à 6: IOxS.Instance

Ce champ doit être codé à l'aide de valeurs indiquées au Tableau 175 si le bit IOxS.DataState est réglé à Bad. Si ce bit a pour valeur Good, les bits de l'instance IOxS ne sont pas pris en compte et doivent prendre une valeur nulle.

Tableau 175 – IOxS.Instance

Valeur (hexadécimale)	Signification
0x00	Déecté par la sous-baie
0x01	Déecté par la sous-baie
0x02	Déecté par l'appareil E/S
0x03	Déecté par l'appareil de commande E/S

Bit 7: IOxS.DataState

Ce champ doit être codé avec les valeurs conformes au Tableau 176.

Tableau 176 – IOxS.DataState

Valeur (hexadécimale)	Signification
0x00	Mauvais Le champ de données ne peut pas contenir de données utilisateur valides. Le destinataire doit utiliser ses valeurs préalablement configurées (zéro, la dernière valeur valide ou la valeur par défaut) à la place des valeurs transmises du champ de données SubstituteDataValid = FALSE
0x01	Bon Le champ de données doit contenir des données utilisateur valides SubstituteDataValid = TRUE

4.7.2.1.5.3.2 Codage du champ IOCS

Le codage de ces champs doit être conforme à 4.7.2.1.5.3.1.

4.7.2.1.5.3.3 Codage du champ SubstituteDataValid

Le codage de ces champs doit être conforme à 4.7.2.1.5.3.1.

4.7.3 Machines de Protocole de Service FAL

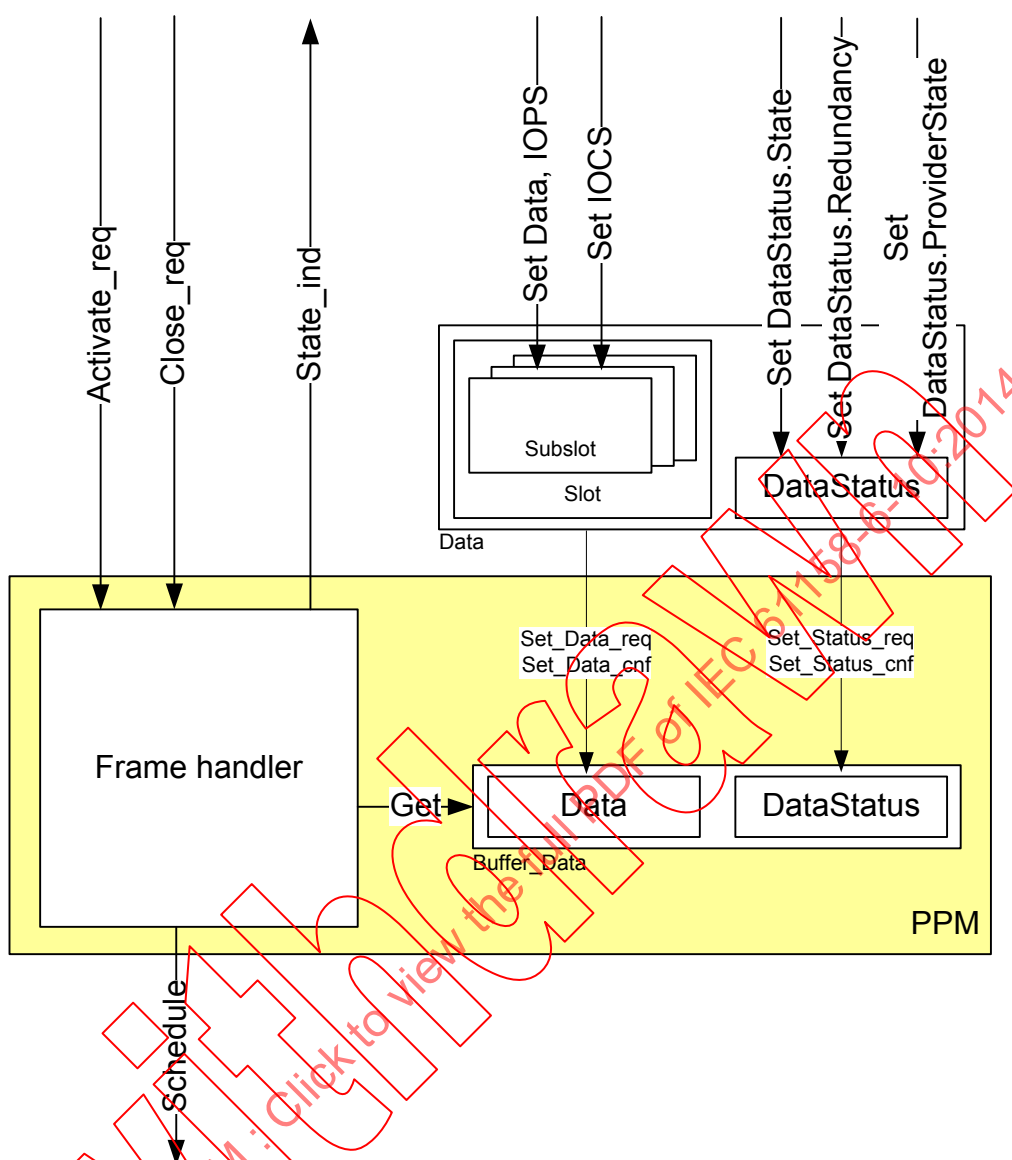
Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.7.4 Machines de protocole de relation entre applications

4.7.4.1 Machine de Protocole de Fournisseur

4.7.4.1.1 Généralités

La Figure 53 et la Figure 54 présentent la structure de base de la Machine de Protocole de Fournisseur (PPM).



Légende

Anglais	Français
Set Data, IOPS	Réglage de Data, IOPS
Set IOCS	Réglage d'IOCS
Set DataStatus.State	Réglage de DataStatus.State
Set DataStatus.Redundancy	Réglage de DataStatus.Redundancy
Set DataStatus.ProviderState	Réglage de DataStatus.ProviderState
Subslot	Sous-emplacement
Slot	Emplacement
Frame handler	Gestionnaire de trame
Get	Récupère
PPM	PPM
Schedule	Programme
Slot	Emplacement
Subslot	Sous-emplacement

Figure 53 – Structure de base d'une PPM comportant une structure de trame

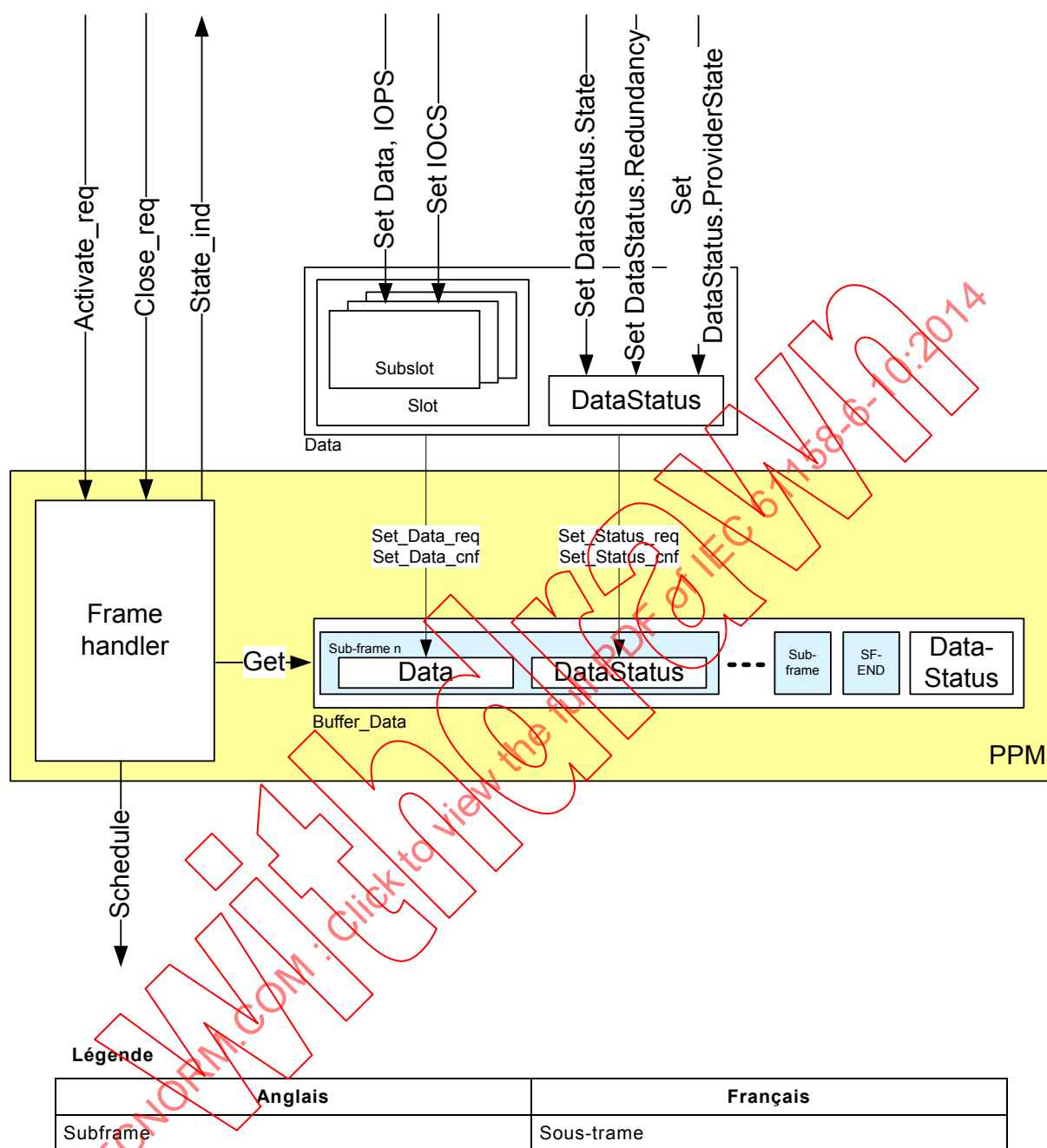


Figure 54 – Structure de base d'une PPM comportant une structure de sous-trame

4.7.4.1.2 Définitions de primitives

4.7.4.1.2.1 Primitives échangées entre les machines distantes

Les primitives de service distantes, y compris leurs paramètres associés, émises ou reçues par PPM sont décrites dans la définition de service et présentées dans le Tableau 177.

Tableau 177 – Primitives distantes émises ou reçues par PPM

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.7.4.1.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par PPM sont décrites dans la définition de service et présentées dans le Tableau 178.

Tableau 178 – Primitives locales émises ou reçues par PPM

Primitive	Source	Destination	Paramètres associés	Fonctions
PPM_Activate_cnf (-)	PPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
PPM_Activate_cnf (+)	PPM	CMSU CTLSU	CREP	—
PPM_Activate_req	CMSU CTLSU	PPM	CREP, DA, SA, FrameID, Prio, TxOption, ReductionRatio, Phase, Sequence, Default_values, Default_Status	—
PPM_Close_cnf (-)	PPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
PPM_Close_cnf (+)	PPM	CMSU CTLSU	CREP	—
PPM_Close_req	CMSU CTLSU	PPM	CREP	—
PPM_Error_ind	PPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
PPM_Set_Data_cnf (-)	PPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
PPM_Set_Data_cnf (+)	PPM	FSPMDEV FSPMCTL	CREP	—
PPM_Set_Data_req	FSPMDEV FSPMCTL	PPM	CREP, Data	—
PPM_Set_Status_cnf (-)	PPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
PPM_Set_Status_cnf (+)	PPM	FSPMDEV FSPMCTL	CREP	—
PPM_Set_Status_req	FSPMDEV FSPMCTL	PPM	CREP, Status	—
PPM_State_ind	PPM	local matter	CREP, State	—
Scheduler_add_cnf (-)	SCHEDULER	PPM	CREP	—
Scheduler_add_cnf (+)	SCHEDULER	PPM	CREP	—
SCHEDULER_add_req	PPM	SCHEDULER	CREP, ReductionRatio, Phase, Sequence, TxOption	—
Scheduler_remove_cnf (-)	SCHEDULER	PPM	CREP	—
Scheduler_remove_cnf (+)	SCHEDULER	PPM	CREP	—
SCHEDULER_remove_req	PPM	SCHEDULER	CREP	—

Primitive	Source	Destination	Paramètres associés	Fonctions
Scheduler_transmit_ind	SCHEDULER	PPM	CREP	—

4.7.4.1.3 Schéma de transition d'états

Le schéma de transition d'états du PPM est présenté à la Figure 55.

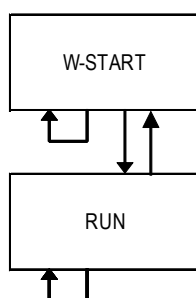


Figure 55 – Schéma de transition d'états of PPM

Etats du PPM

W-START

RUN

En attente de l'initialisation et de l'ajout de la trame au SCHEDULER

En cours d'exécution

4.7.4.1.4 Description du diagramme d'états

L'état W-START signale la nécessité d'une initialisation. Le service Activate place la machine à l'état RUN en attendant les demandes de service Provider Data et les événements temporels. Suite à la réception de la confirmation, la machine revient à l'état RUN. Une erreur ou une demande de service Close est nécessaire pour revenir à l'état W-START.

Chaque PPM gère jusqu'à deux trames avec un ou deux FrameID. Pour une compatibilité avec RTC3, deux trames avec le même FrameID sont gérées et pour MRPD, deux trames avec deux FrameIDs (FrameID et FrameID+1) sont gérées.

Cela sert à préparer les trames qui sont transmises par le Scheduler (Programmateur). Le contenu (Buffer_Data) comporte les données locales si la "frame structure" est utilisée ou les données locales ainsi que les sous-trames enregistrées si "subframe structure" est utilisée.

Le Buffer_Data is est récupéré d'abord à partir de la trame INBOUND DFP et de DFP_STORAGE si cette trame a trop de retard. Si la sous-trame en cours de réception et de transmission est détectée comme étant invalide, il convient que la trame soit raccourcie en ajoutant une TX-error (voir Annexe S) au port de transmission.

NOTE Le TagControlInformation.VLAN_Id est mis à zéro par le PPM mais peut être différent dans d'autres cas conformément à l'IEEE 802.1Q.

4.7.4.1.5 Table d'états PPM

Le Tableau 179 contient la description de la table d'états PPM.

Tableau 179 – Table d'états PPM

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	W-START	PPM_Activate_req () => FirstTransmit := FALSE Buffer_Data := Default_Values Buffer_Status := Default_Status SCHEDULER_add.req () PPM_Activate_cnf (+)	RUN
2	W-START	PPM_Set_Data_req (Data) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Data_cnf (-)	W-START
3	W-START	PPM_Set_Status_req (Status) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Status_cnf (-)	W-START
4	W-START	PPM_Close_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Close_cnf (-)	W-START
5	W-START	Scheduler_transmit.ind () => ignore	W-START
6	W-START	Scheduler_add.cnf (+) => ignore	W-START
7	W-START	Scheduler_remove.cnf (+) => ignore	W-START
8	W-START	Scheduler_add.cnf (-) => ignore	W-START
9	W-START	Scheduler_remove.cnf (-) => ignore	W-START
10	RUN	PPM_Activate_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Activate_cnf (-)	RUN
11	RUN	PPM_Set_Data_req (Data) => Buffer_Data := Data PPM_Set_Data_cnf (+)	RUN
12	RUN	PPM_Set_Status_req (Status) => Buffer_Status := Status PPM_Set_Status_cnf (+)	RUN
13	RUN	PPM_Close_req () => SCHEDULER_remove.req () PPM_Close_cnf (+)	W-START
14	RUN	Scheduler_transmit.ind () /FirstTransmit == FALSE => FirstTransmit := TRUE PPM_State_ind (Start)	RUN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
15	RUN	Scheduler_transmit.ind () /FirstTransmit == TRUE => ignore	RUN
16	RUN	Scheduler_add.cnf (+) => ignore	RUN
17	RUN	Scheduler_remove.cnf (+) => ignore	RUN
18	RUN	Scheduler_add.cnf (-) => ERRCLS := CTXT ERRCODE := INVALID PPM_State_ind (Error) PPM_Error_ind (Error)	W START
19	RUN	Scheduler_remove.cnf (-) => ignore	RUN

4.7.4.1.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 180 contient les fonctions, macros, temporisateurs et variables utilisés par le PPM ainsi que leurs arguments et leurs descriptions.

Tableau 180 – Fonctions, Macros, Temporisateurs et Variables utilisés par le PPM

Nom	Type	Fonction/Signification
FirstTransmit	Variable	Cette variable locale contient les informations stipulant si la trame a été acheminée au moins une fois.
Buffer_Data	Variable	Cette variable locale contient les données (codés en DataItem*) qui doivent être utilisées pour stocker les données prévues pour l'acheminement du Data-RTC-PDU ou UDP-RTC-PDU suivant.
Buffer_Status	Variable	Cette variable locale contient le statut (codés en APDU_Status.DataStatus) qui doit être utilisé pour stocker les statuts prévus pour l'acheminement du Data-RTC-PDU ou UDP-RTC-PDU suivant.
NOTE DataItem* inclut tous les DataItems pour la PDU dédiée.		

4.7.4.1.7 Tables de vérité

Le Tableau 181 contient les combinaisons valides, leur signification et leur description de TxOption.

Tableau 181 – Table de vérité utilisée par le PPM pour TxOption

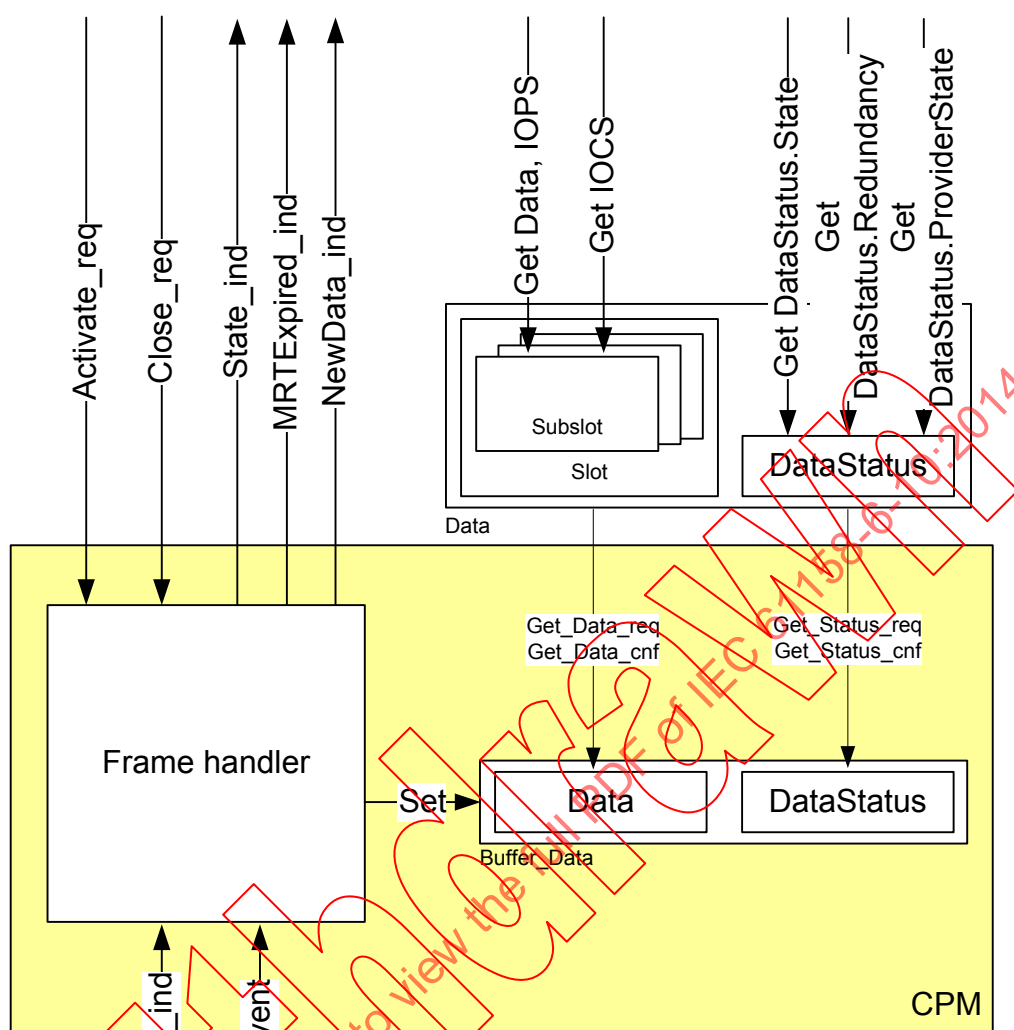
RT_CLASS_X	ARProperties. StartupMode	IRT	MRPD	DFP	Description
		FrameSendOffset, Port	FrameID, FrameSendOffset, Port	SFPosition	
RT_CLASS_1, RT_CLASS_2, RT_CLASS_UDP	—	—	—	—	Le PPM doit créer une trame

RT_CLASS_X	ARProperties. StartupMode	IRT	MRPD	DFP	Description
		FrameSendOffset, Port	FrameID, FrameSendOffset, Port	SFPosition	
RT_CLASS_3	Legacy	Valeurs données	—	—	Le PPM doit créer deux trames. Lors du démarrage (Voir Annexe B) une trame avec les valeurs données du service de connexion et une trame avec RR=128 envoient uniquement dans la PERIODE VERTE. Après ReadyForRTC3 la trame supplémentaire (VERTE) doit être supprimée.
RT_CLASS_3	Legacy	Valeurs données	Valeurs données	—	Non pris en charge
RT_CLASS_3	Legacy	Valeurs données	—	Valeurs données	Non pris en charge
RT_CLASS_3	Legacy	Valeurs données	Valeurs données	Valeurs données	Non pris en charge
RT_CLASS_3	Advanced	Valeurs données	—	—	Le PPM doit créer une trame (voir Annexe A)
RT_CLASS_3	Advanced	Valeurs données	Valeurs données	—	Le PPM doit créer deux trames (voir Annexe A et Annexe Q)
RT_CLASS_3	Advanced	Valeurs données	—	Valeurs données	Le PPM doit créer une trame (voir Annexe A)
RT_CLASS_3	Advanced	Valeurs données	Valeurs données	Valeurs données	Le PPM doit créer deux trames (voir Annexe A et Annexe Q)

4.7.4.2 Machine de Protocole de Consommateur

4.7.4.2.1 Généralités

La Figure 56 présente la structure de base de la Machine de Protocole de Consommateur (CPM).



Légende

Anglais	Français
Get Data, IOPS	Récupération de Data, d'IOPS
Get IOCS	Récupération d'IOCS
Get DataStatus.State	Récupération de DataStatus.State
Get DataStatus.Redundancy	Récupération de DataStatus.Redundancy
Get DataStatus.ProviderState	Récupération de DataStatus.ProviderState
Subslot	Sous-emplacement
Slot	Emplacement
Frame handler	Gestionnaire de trames
Set	Met en place

Figure 56 – Structure de base d'une CPM

4.7.4.2.2 Définitions de primitives

4.7.4.2.2.1 Primitives échangées entre les machines distantes

Les primitives de service distantes, y compris leurs paramètres associés, émises ou reçues par CPM sont décrites dans la définition de service et présentées dans le Tableau 182.

Tableau 182 – Primitives distantes émises ou reçues par une CPM

Primitive	Source	Destination	Paramètres associés	Fonctions
LMPM_C_Data.ind	LMPM	CPM	CREP, Port, DA, SA, Prio, C_SDU, APDU_Status	—
UDP_C_Data.ind	IP	CPM	CREP, IPPort, DA, SA, Prio, C_SDU, APDU_Status	—

4.7.4.2.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par CPM sont décrites dans la définition de service et présentées dans le Tableau 185.

Tableau 183 – Primitives locales émises ou reçues par CPM

Primitive	Source	Destination	Paramètres associés	Fonctions
CPM_Activate_cnf (-)	CPM	CMSU CMDMC CTLSU	CREP, ERRCLS, ERRCODE	—
CPM_Activate_cnf (*)	CPM	CMSU CMDMC CTLSU	CREP	—
CPM_Activate_req	CMSU CMDMC CTLSU	CPM	CREP, DA, SA, FrameID, RxOption, Exp_Length, DataHoldFactor, Default_Value, Default_Status	—
CPM_Close_cnf (+)	CPM	CMSU CMDMC CTLSU	CREP	—
CPM_Close_req	CMSU CMDMC CTLSU	CPM	CREP	—
CPM_Error_ind	CPM	CMSU CTLSU	CREP, ERRCLS, ERRCODE	Locally generated event if something strange happens
CPM_Get_Data_cnf (-)	CPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
CPM_Get_Data_cnf (+)	CPM	FSPMDEV FSPMCTL	CREP, Data, New_Flag	—

Primitive	Source	Destination	Paramètres associés	Fonctions
CPM_Get_Data_req	FSPMDEV FSPMCTL	CPM	CREP	—
CPM_Get_Status_cnf (-)	CPM	FSPMDEV FSPMCTL	CREP, ERRCLS, ERRCODE	—
CPM_Get_Status_cnf (+)	CPM	FSPMDEV FSPMCTL	CREP, Status, RecvCounter	—
CPM_Get_Status_req	FSPMDEV FSPMCTL	CPM	CREP	—
CPM_MRTEexpired_ind	CPM	Diagnosis Module	info	—
CPM_NewData_ind	CPM	FSPMDEV FSPMCTL CMIO CTLIO CMDMC	AREP, CREP, APDU_Status, data {Data, NoData}	—
CPM_State_ind	CPM	CMIO CTLIO CMDMC	state {Start, Stop}	—

4.7.4.2.3 Schéma de transition d'états

Le schéma de transition d'états du CPM est présenté à la Figure 57.

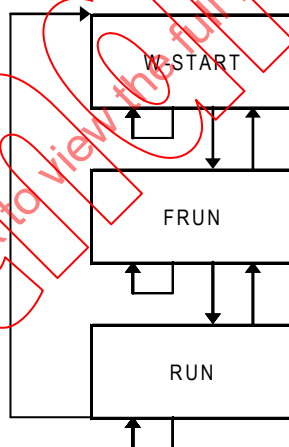


Figure 57 – Schéma de transition d'états de CPM

Etats du CPM

W-START

L'état W-START signale la nécessité d'une initialisation. Le service Activate place la machine à l'état RUN ou FRUN (première exécution).

FRUN

Le diagramme d'états attend la première indication de service C_Data_ind dans ces états. Le DataHoldTime est ignoré dans cet état. Il passe à l'état RUN après avoir transmis l'indication Start à l'utilisateur.

RUN

Dans cet état, la surveillance à l'aide de DataHoldtime est active. Une temporisation forcera la transition à l'état FRUN (combiné à une indication Stop). Une demande de service Close est nécessaire pour retourner à l'état W-START.

4.7.4.2.4 Description du diagramme d'états

L'état W-START signale la nécessité d'une initialisation. Le service Activate place la machine à l'état RUN ou FRUN (première exécution). Le diagramme d'états attend la première indication de service LMPM_C_Data dans ces états. Il passe à l'état RUN après avoir transmis l'indication Start à l'utilisateur.

Une temporisation forcera la transition à l'état FRUN (combiné à une indication Stop). Une demande de service Close est nécessaire pour retourner à l'état W-START.

A chaque réception de C_Data.ind valide, l'heure est lue et enregistrée dans une variable locale avec un fanion.

Chaque CPM gère jusqu'à deux trames avec un ou deux FrameID. Pour une compatibilité avec RTC3, deux trames avec le même FrameID sont gérées et pour MRPD, deux trames avec deux FrameIDs (FrameID et FrameID+1) sont gérées.

4.7.4.2.5 Table d'états CPM

Le Tableau 184 contient la description exhaustive de la table d'états CPM.

Tableau 184 – Table d'états CPM

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	W-START	CPM_Activate_req () => Store DA, SA, FrameID, RxOption, DataHoldFactor, Default_Value, Default_Status New_Data := FALSE for all ports of interface do: RxA(FrameID) := Invalid DHT := 0 Cycle := 0 RecvCnt := 0 Buffer_Data := Default_Value Buffer_Status := Default_Status Primary := TRUE ControlInterval := SendClockFactor x ReductionRatio x 31,25 µs StartTimer (ControlInterval) CPM_Activate_cnf (+)	FRUN
2	W-START	CPM_Close_req () => StopTimer (ControlInterval) CPM_Close_cnf (+)	W-START
3	W-START	CPM_Get_Status_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Get_Status_cnf (-)	W-START
4	W-START	CPM_Get_Data_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Get_Data_cnf (-)	W-START
5	W-START	C_Data_ind => ignore	W-START
6	FRUN	CPM_Activate_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Activate_cnf (-)	FRUN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
7	FRUN	CPM_Close_req () => StopTimer () CPM_Close_cnf (+)	W-START
8	FRUN	CPM_Get_Status_req () => Status := Cycle, Buffer_Status RecvCounter := RecvCnt RecvCnt := 0 CPM_Get_Status_cnf (+)	FRUN
9	FRUN	CPM_Get_Data_req () => Data := Buffer_Data New_Flag := New_Data New_Data := FALSE CPM_Get_Data_cnf (+)	FRUN
10	FRUN	C_Data_ind /FrameGuard () == INVALID => ignore	FRUN
11	FRUN	C_Data_ind /FrameGuard () == VALID, DHTReload, UpdatedData => New_Data := TRUE RxA(Port, FrameID) := APDU_Status.Cycle_Counter Dht := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Data := Filter (Data) Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_State_ind (Start) CPM_NewData_ind (Data) If MRPD active then StartTimer (MRPDt) //NOTE DataStatus.State == don't care and DataStatus.DataValid == Valid	RUN
12	FRUN	C_Data_ind /FrameGuard () == VALID, DHTReload => RxA(Port, FrameID) := APDU_Status.Cycle_Counter Dht := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_State_ind (Start) CPM_NewData_ind (NoData) If MRPD active then StartTimer (MRPDt) //NOTE DataStatus.State == Backup	RUN
13	FRUN	TimerExpired (ControllInterval) => ignore	FRUN
14	FRUN	TimerExpired (MRWDt) => ignore	FRUN
15	RUN	CPM_Activate_req () => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Activate_cnf (-)	RUN
16	RUN	CPM_Close_req () => StopTimer (ControllInterval) CPM_Close_cnf (+)	W-START

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
17	RUN	CPM_Get_Status_req () => Status := Cycle, Buffer_Status, RecvCounter := RecvCnt, RecvCnt := 0 CPM_Get_Status_cnf (+)	RUN
18	RUN	CPM_Get_Data_req () => Data := Buffer_Data, New_Flag := New_Data New_Data := FALSE CPM_Get_Data_cnf (+)	RUN
19	RUN	C_Data_ind /FrameGuard () == INVALID => ignore	RUN
20	RUN	C_Data_ind /FrameGuard () == VALID, DHTReload, UpdateData => New_Data := TRUE, RxA(Port, FrameID) := APDU_Status.Cycle_Counter DHT := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Data := Filter (Data) Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_NewData_ind (Data) //NOTE DataStatus.State == don't care and DataStatus.DataValid == Valid	RUN
21	RUN	C_Data_ind /FrameGuard () == VALID, DHTReload => RxA(Port, FrameID) := APDU_Status.Cycle_Counter DHT := 0 Cycle := APDU_Status.Cycle_Counter Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_NewData_ind (NoData) //NOTE DataStatus.State == Backup	RUN
22	RUN	TimerExpired (ControllInterval) /DHT > (DataHoldFactor-1) => DHT := 0 CPM_State_ind (Stop) //NOTE DHT a expiré	FRUN
23	RUN	TimerExpired (ControllInterval) /DHT < (DataHoldFactor) => DHT := DHT + 1	RUN
24	RUN	TimerExpired (MRWDt) /MRPD active && MRWDt expired => CPM_MRTEexpired_ind (Fault)	RUN
25	RUN	TimerExpired (MRWDt) /MRPD active && MRPDGuard () == Adjunctive frame received => for all ports, FrameIDs RxA(Port, FrameID) := Invalid StopTimer (MRWDt) CPM_MRTEexpired_ind (Ok)	RUN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
26	RUN	TimerExpired (MRWDt) /MRPD active && MRPDGuard () == No adjunctive frame received => for all ports, FrameIDs RxA(Port, FrameID) := Invalid StartTimer (MRWDt)	RUN

4.7.4.2.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 185 contient les fonctions, macros, temporisateurs et variables utilisés par le CPM ainsi que leurs arguments et leurs descriptions.

Tableau 185 – Fonctions, Macros, Temporisateurs et Variables utilisés par le CPM

Nom	Type	Fonction/Signification
Filtre	Fonction	Cette fonction locale filtre à partir des données reçues celles étant la partie stockée pour Buffer_Data
FrameGuard	Fonction	Cette fonction locale vérifie la trame reçue par rapport à la structure et à la sémantique attendues conformément au codage et aux tables de vérité. VALID / INVALID: Ce fanion indique si la trame reçue est valide. DHTReload: Ce fanion indique que le DataHoldTimer doit être à nouveau déclenché. UpdateData: Ce fanion indique que Buffer_Data et Buffer_Status doivent être mis à jour à l'aide des données reçues.
MRPDGuard	Fonction	Cette fonction locale vérifie si la trame supplémentaire définie par l'activation de MRPD est reçue ou non.
StartTimer	Fonction	Elle utilise la variable locale RxA comme base de données.
StopTimer	Fonction	Cette fonction locale lance ou relance un temporisateur.
TimerExpired	Fonction	Cette fonction locale arrête un temporisateur.
C_Data_ind	Macro	Cette fonction locale est émise si un temporisateur expire. si (RxOption == RTC) LMPM_C_Data.ind (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) si (RxOption == UDP) UDP_C_Data.ind (IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status)
RxOption	Macro	Contient - RT_CLASS_x - FrameID, - if MRPD - FrameID (chemin redondant), - Media redundancy watchdog (MRPD actif) - if DFP - subaddress SFPosition
ControllInterval	Temporisateur	Ce temporisateur local contient l'intervalle de surveillance pour DataHoldFactor. ControllInterval:= SendClockFactor x ReductionRatio x 31,25 µs Ce temporisateur doit utiliser une fonction d'autorecharge pour se lancer après son expiration. Il doit être aligné au PTCP ou à SendClock fonctionnant localement.

Nom	Type	Fonction/Signification
MRWDt	Temporisateur	Ce temporisateur local contient l'intervalle de surveillance du délai de WatchDog de redondance de support. Il est utilisé pour surveiller la réception des trames supplémentaires définies par MRDP et pour émettre un événement s'il a expiré.
Buffer_Data	Variable	Cette variable locale contient la dernière unité de données de service valide acheminée avec une indication LMPM_C_Data.
Buffer_Status	Variable	Cette variable locale contient le dernier APDU_Status valide acheminé avec une indication LMPM_C_Data.
Cycle	Variable	Cette variable locale contient le dernier Cycle Counter valide acheminé avec une indication LMPM_C_Data contenant des données valides.
Default_Status	Variable	Le CycleCounter est réglé à invalide, le TransferStatus et le DataStatus sont réglés à zéro.
Default_Value	Variable	Data est réglé à zéro et le IOxS = BAD.
DHt	Variable	Cette variable locale contient la valeur de compteur DataHoldFactor des événements temporels depuis la dernière indication LMPM_C_Data.
ERRCLS	Variable	Cette variable locale contient l'entité signalant l'erreur.
ERRCODE	Variable	Cette variable locale contient la raison de l'erreur dans le contexte du ERRCLS.
New_Data	Variable	Cette variable locale signale la réception d'une indication LMPM_C_Data valide (définie sur FALSE par une demande de GetData).
RecvCnt	Variable	Cette variable locale contient la valeur de compteur du PDU cyclique reçu depuis la dernière demande de GetStatus.
RxA	Variable	Cette variable locale contient les informations enregistrées concernant les trames reçues pour la vérification MRPD. Principe: RxA(m, n) == RxA(0, n+1) // les APDU_Status.CycleCounter stockés sont égaux // élément m, 0 {1..MaxPort} // élément n, n+1 {FrameID, FrameID+1}

Le Tableau 186 contient les combinaisons valides, leur signification et leur description pour RxOption.

Tableau 186 – Table de vérité utilisée par le CPM pour RxOption

RT_CLASS_X	ARProperties. StartupMode	MRPD	DFP	Description
		FrameID, Chien de garde de redondance	SFPosition	
RT_CLASS_1, RT_CLASS_2, RT_CLASS_UDP	—	—	—	Le CPM doit créer un consommateur pour une trame
RT_CLASS_3	FALSE	—	—	Le CPM doit créer un consommateur pour une trame. Lors du démarrage (voir Annexe B) les valeurs DHT sont pour RR=256 et après ReadyForRTC3 comme données lors du service de connexion.
RT_CLASS_3	FALSE	Valeurs données	—	Non pris en charge
RT_CLASS_3	FALSE	—	Valeurs données	Non pris en charge

RT_CLASS_X	ARProperties. StartupMode	MRPD	DFP	Description
		FrameID, Chien de garde de redondance	SFPosition	
RT_CLASS_3	FALSE	Valeurs données	Valeurs données	Non pris en charge
RT_CLASS_3	TRUE	—	—	Le CPM doit créer un consommateur pour une trame (voir Annexe A)
RT_CLASS_3	TRUE	Valeurs données	—	Le CPM doit créer un consommateur pour deux trames (voir Annexe A et Annexe Q)
RT_CLASS_3	TRUE	—	Valeurs données	Le CPM doit créer un consommateur pour une trame (voir Annexe A)
RT_CLASS_3	TRUE	Valeurs données	Valeurs données	Le CPM doit créer un consommateur pour deux trames (voir Annexe A et Annexe Q)

4.7.4.2.7 Tables de vérité

4.7.4.2.7.1 Règles de vérification du CPM pour une trame

Le Tableau 187, le Tableau 188, le Tableau 189 et le Tableau 190 contiennent la table de vérité utilisée par le CPM pour les trames.

Tableau 187 – Table de vérité pour une trame utilisant RT_CLASS_x

Entrée					Sortie
Trame reçue					Local
Received-InRED ^c	APDU_Status. TransferStatus	SourceMAC- Address ^b	FrameID, redundant FrameID ^a	C_SDU. Longueur	Structure de la trame
FALSE	—	—	—	—	Non valide
TRUE	Non valide	—	—	—	Non valide
TRUE	Valide	Incorrect	—	—	Non valide
TRUE	Valide	Ok	Incorrect	—	Non valide
TRUE	Valide	Ok	Ok	Incorrect	Non valide
TRUE	Valide	Ok	Ok	Ok	Valide
^a La SFPosition.Position (configuration de trame différente) est gérée par le DFP_RELAY.					
^b Pour les trames utilisant DFP facultatif.					
^c Uniquement vérifié pour RT_CLASS_3.					

Tableau 188 – Table de vérité pour une trame utilisant RT_CLASS_UDP

Entrée					Sortie
Trame reçue					Local
APDU_Status. TransferStatus	SourceIP- Address	SourceIP- Port	FrameID, redundant FrameID	C_SDU. Longueur	Structure de la trame

Entrée					Sortie
Trame reçue					Local
Non valide	—	—	—	—	Non valide
Valide	Incorrect	—	—	—	Non valide
Valide	Ok	Incorrect	—	—	Non valide
Valide	Ok	Ok	Incorrect	—	Non valide
Valide	Ok	Ok	Ok	Incorrect	Non valide
Valide	Ok	Ok	Ok	Ok	Valide

Tableau 189 – Table de vérité pour C_SDU

Entrée	Sortie
Trame reçue	Local
APDU_Status.CycleCounter	C_SDU structure
Ancien ^a	Non valide
Nouveau ^b	Valide
^a Voir 4.7.2.1.2 et le Tableau 161. Lors de l'activation d'une CPM, la CycleCounterValue stockée est mise sur non valide pour rendre la prochaine trame reçue "Nouvelle" dans tous les cas. ^b Voir 4.7.2.1.2 et le Tableau 161.	

Tableau 190 – Table de vérité pour l'agencement de DHT et données

Entrée				Sortie		
Trame reçue				Local		
Structure de la trame	C_SDU structure	APDU_Status.DataStatus.DataValid	APDU_Status.DataStatus.Etat	Temporisateur du maintien des données	Mise à jour des données	FrameGuard
Non valide	—	—	—	—	Non	Non valide
Valide	Non valide	—	—	—	Non	Non valide
Valide	Valide	Non valide	Primaire	—	Non	Non valide
Valide	Valide	Non valide	Sauvegarde	Recharge	Non	Valide
Valide	Valide	Valide	Primaire	Recharge	Oui	Valide
Valide	Valide	Valide	Sauvegarde	Recharge	Oui	Valide

4.7.4.2.7.2 Règles de vérification du CPM pour une sous-trame

Le Tableau 191, le Tableau 192, le Tableau 193 et le Tableau 194 contiennent la table de vérité utilisée par le CPM pour les sous-trames.

Tableau 191 – Table de vérité pour la sous-trame – vérification de la trame

Entrée			Sortie
FrameID	SourceMAC-address	En-tête Subframe. SFCRC16	Vérification de trame
FALSE	—	—	Non valide
TRUE	FALSE	—	Non valide
TRUE	TRUE	Non valide	Non valide

TRUE	TRUE	Valide	Valide
TRUE	TRUE	Inutile	Valide

Tableau 192 – Table de vérité pour la sous-trame – vérification de la sous-trame

Entrée			Sortie
Subframe. N° de la	Subframe. Longueur	Subframe. SFCRC16	Vérification de sous-trame
FALSE	—	—	Non valide
TRUE	FALSE	—	Non valide
TRUE	TRUE	Non valide	Non valide
TRUE	TRUE	Valide	Valide
TRUE	TRUE	Inutile	Valide

Tableau 193 – Table de vérité pour la sous-trame – vérification des données de la sous-trame

Entrée		Sortie
Subframe. DataStatus. Ignorer	Subframe. SFCycleCounter	Vérification des données de sous-trame
Ignorer	—	Non valide
Ok	Ancien	Non valide
Ok	Nouveau	Valide

Tableau 194 – Table de vérité de la sous-trame – DHT et données

Entrée					Sortie		
Trame reçue					Local		
Vérification de trame	Vérification de sous-trame	Vérification des données de sous-trame	Subframe. DataStatus. Valide	Subframe. DataStatus. Etat	Temporisateur du maintien des données	Mise à jour des données	Frame-Guard
Non valide	—	—	—	—	—	Non	Non valide
Valide	Non valide	—	—	—	—	Non	Non valide
Valide	Valide	Non valide	—	—	—	Non	Non valide
Valide	Valide	Valide	Non valide	Sauvegarde	Recharge	Non	Valide
Valide	Valide	Valide	Valide	Primaire	Recharge	Oui	Valide
Valide	Valide	Valide	Valide	Sauvegarde	Recharge	Oui	Valide

4.7.5 Machines de Protocole de Mapping DLL

Il n'existe aucune Machine de Protocole de Mapping DLL (DMPM) définie pour ce Protocole.

4.8 Acyclique en temps-réel

4.8.1 Description de syntaxe RTA

4.8.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.8.1.2 Syntaxe abstraite RTA APDU

Le Tableau 195 définit la syntaxe abstraite des PDU de couche application appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 195 doivent représenter le contenu de la DLSDU dans le Tableau 5.

Tableau 195 – Syntaxe RTA APDU

Nom de l'APDU	Structure de l'APDU
RTA-PDU	DATA-RTA-PDU ^ ACK-RTA-PDU ^ NACK-RTA-PDU ^ ERR-RTA-PDU
UDP-RTA-PDU	IPHeader, UDPHeader, FrameID, RTA-PDU
NOTE Le DLPDU_Padding s'applique à RTA-PDU et à UDP-RTA-PDU.	

Le Tableau 196 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 195.

Tableau 196 – Substitutions RTA

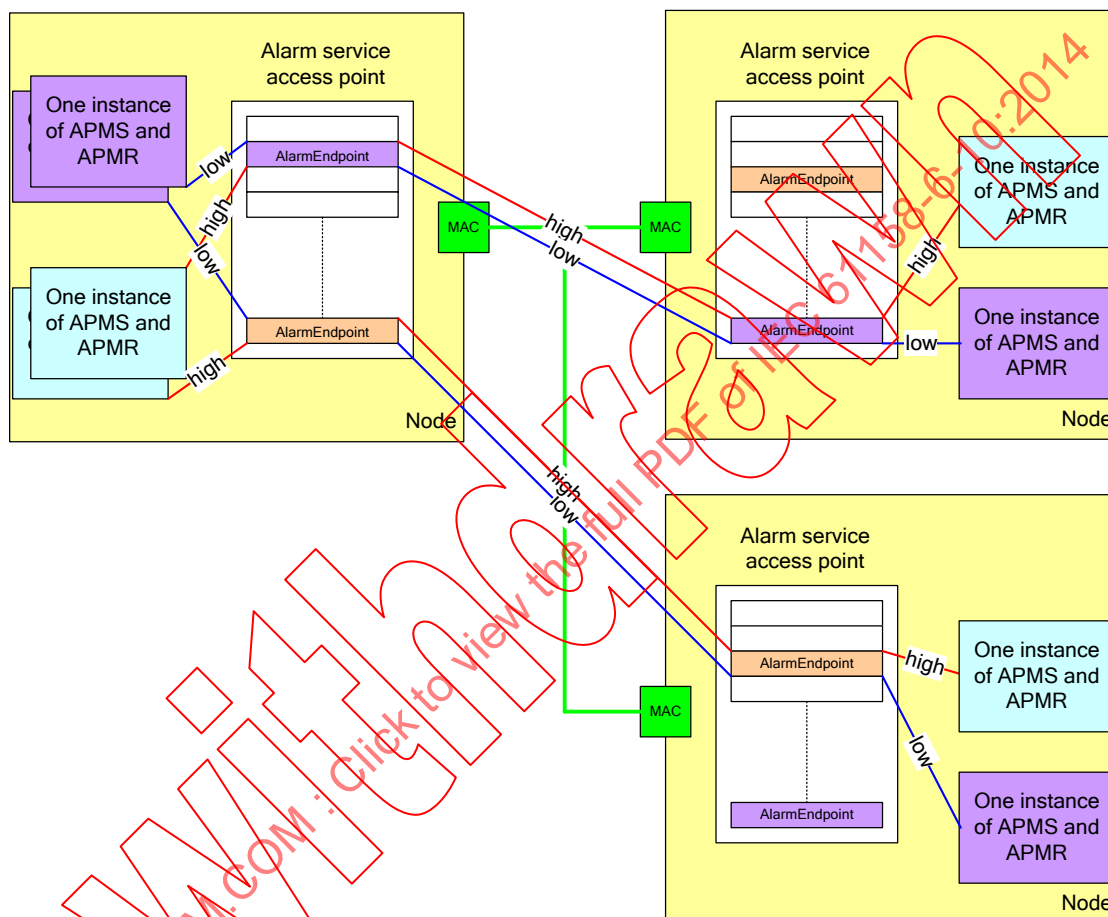
Nom de substitution	Structure
Référence	DestinationServiceAccessPoint, SourceServiceAccessPoint
Destination-ServiceAccess-Point	AlarmEndpoint
Source-ServiceAccess-Point	AlarmEndpoint
FlagsSequence	AddFlags, SendSeqNum, AckSeqNum
VendorDevice-ErrorInfo	VendorID, DeviceID, Data*
DATA-RTA-PDU	Reference, PDUType (=1), FlagsSequence, VarPartLen (1 – 1 432) ^a , RTA-SDU
ACK-RTA-PDU	Reference, PDUType (=3), FlagsSequence, VarPartLen (=0)
NACK-RTA-PDU	Reference, PDUType (=2), FlagsSequence, VarPartLen (=0)
ERR-RTA-PDU	Reference, PDUType (=4), FlagsSequence, VarPartLen (=4), PNIOStatus, [VendorDeviceErrorInfo] ^b
^a Le VarPartLen maximal est calculé pour être conforme à RTA-PDU et à UDP-RTA-PDU.	
^b Le nombre d'octets pour cette substitution ne doit pas dépasser le DLPDU_Padding. Voir la taille minimale de la trame dans IEEE 802.3.	

4.8.2 Syntaxe de transfert RTA

4.8.2.1 Section de codage relative aux champs spécifiques de RTA-PDU

4.8.2.1.1 Codage du champ AlarmEndpoint

Ce champ doit être codé en type de données Unsigned16. Il identifie en liaison avec Host-Adress les points d'accès au service d'une connexion bidirectionnelle comme le montre la Figure 58. Les valeurs pour la source et la destination doivent être échangées lors d'une opération de gestion de contexte.



Légende

Anglais	Français
Alarm service access point	Point d'accès au service d'alarme
One instance of APMS and APMR	Une instance de APMS et APMR
Alarm EndPoint	Extrémité de l'alarme
Node	Nœud
High	Élevé
Low	Faible
MAC	MAC

Figure 58 – Diagramme d'adressage de RTA

NOTE L'appareil de commande E/S envoie un AlarmEndpoint pendant qu'il se connecte à l'appareil d'E/S en utilisant le champ LocalAlarmReference dans AlarmCRBlockReq. L'appareil E/S répond avec son AlarmEndpoint en utilisant le champ LocalAlarmReference dans AlarmCRBlockRes.

L'appareil de commande E/S utilise son AlarmEndpoint comme SourceServiceAccessPoint et le AlarmEndpoint des appareils E/S comme DestinationServiceAccessPoint. L'appareil E/S utilise son AlarmEndpoint comme SourceServiceAccessPoint et le AlarmEndpoint de l'appareil de commande E/S comme DestinationServiceAccessPoint.

4.8.2.1.2 Codage du champ PDUType

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0 – 3: PDUType.Type

Ce champ doit être codé avec les valeurs conformes au Tableau 197.

Tableau 197 – PDUType.Type

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	—
0x01	RTA_TYPE_DATA	Ne doit être utilisé que pour encoder DATA-RTA-PDU
0x02	RTA_TYPE_NACK	Ne doit être utilisé que pour encoder NACK-RTA-PDU
0x03	RTA_TYPE_ACK	Ne doit être utilisé que pour encoder ACK-RTA-PDU
0x04	RTA_TYPE_ERR	Ne doit être utilisé que pour encoder ERR-RTA-PDU
0x05 – 0x0F	Réservé	—

Bit 4 – 7: PDUType.Version

Ce champ doit être codé avec les valeurs conformes au Tableau 198.

Tableau 198 – PDUType.Version

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Version 1 du protocole
0x02 – 0x0F	Réservé

4.8.2.1.3 Codage du champ AddFlags

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0 – 3: AddFlags.WindowSize

Ce champ doit être codé avec les valeurs conformes au Tableau 199.

Tableau 199 – AddFlags.WindowSize

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Taille de la fenêtre un
0x02 – 0x07	Réservé

Bit 4: AddFlags.TACK

Ce champ doit être codé avec les valeurs conformes au Tableau 200 par l'émetteur au sein d'une RTA-PDU pour contrôler le comportement d'acquittement du destinataire.

Tableau 200 – AddFlags.TACK

Valeur (hexadécimale)	Signification	Utilisation
0x00	Acquittement non immédiat	Doit être mis en place pour ERR-RTA-PDU, ACK-RTA-PDU et NACK-RTA-PDU et ne doit pas être vérifié par le destinataire.
0x01	Acquittement immédiat	Doit être mis en place pour l'acquittement immédiat de DATA-RTA-PDU.

Bit 5 – 7: AddFlags.reserved

Ce champ doit être mis conformément à 3.4.2.2.

4.8.2.1.4 Codage du champ SendSeqNum

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 201.

Tableau 201 – SendSeqNum

Valeur (hexadécimale)	Utilisation
0x0000 – 0x7FFF	La première DATA-RTA-PDU émis doit contenir la valeur zéro.
autre	Réservé
0xFFFE	Cette valeur doit être utilisée pour synchroniser l'émetteur et le destinataire après établissement de la relation entre applications. Cette valeur indique qu'aucune DATA-RTA-PDU n'a été reçue auparavant.
0xFFFF	Cette valeur doit être utilisée pour synchroniser l'émetteur et le destinataire après établissement de la relation entre applications. Cette valeur indique la première DATA-RTA-PDU.

Ce champ contient le nombre de DATA-RTA-PDU. L'incrément et la comparaison de ce nombre sont assurées par des opérations de modulo 2^{15} .

NOTE La première DATA-RTA-PDU définit SendSeqNum=0xFFFF et AckSeqNum=0xFFFE. Elle est acquittée avec SendSeqNum=0xFFFE et AckSeqNum=0xFFFF. La deuxième DATA-RTA-PDU définit SendSeqNum=0 et AckSeqNum=0xFFFE. La synchronisation est indispensable, car le protocole acyclique ne définit aucun contrôle de connexion.

4.8.2.1.5 Codage du champ AckSeqNum

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 202.

Tableau 202 – AckSeqNum

Valeur (hexadécimale)	Utilisation
0x0000 – 0x7FFF	La première DATA-RTA-PDU émis doit contenir la valeur zéro.
autre	Réservé
0xFFFE	Cette valeur doit être utilisée pour synchroniser l'émetteur et le destinataire après établissement de la relation entre applications. Cette valeur indique qu'aucune DATA-RTA-PDU n'a été reçue auparavant.
0xFFFF	Cette valeur doit être utilisée pour synchroniser l'émetteur et le destinataire après établissement de la relation entre applications. La valeur fait office d'acquittement de la première DATA-RTA-PDU.

Ce champ contient le nombre des DATA-RTA-PDU dont l'acquittement est prévu ou qui ont été acquittées.

4.8.2.1.6 Codage du champ VarPartLen

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 203. Le VarPartLen ne doit pas contenir DLPDU_Padding.

Tableau 203 – VarPartLen

Valeur (hexadécimale)	Signification
0x0000	Aucune RTA-SDU n'existe
0x0001 – 0x0598	Une RTA-SDU avec des octets VarPartLen existe
0x0599 – 0xFFFF	Réservé

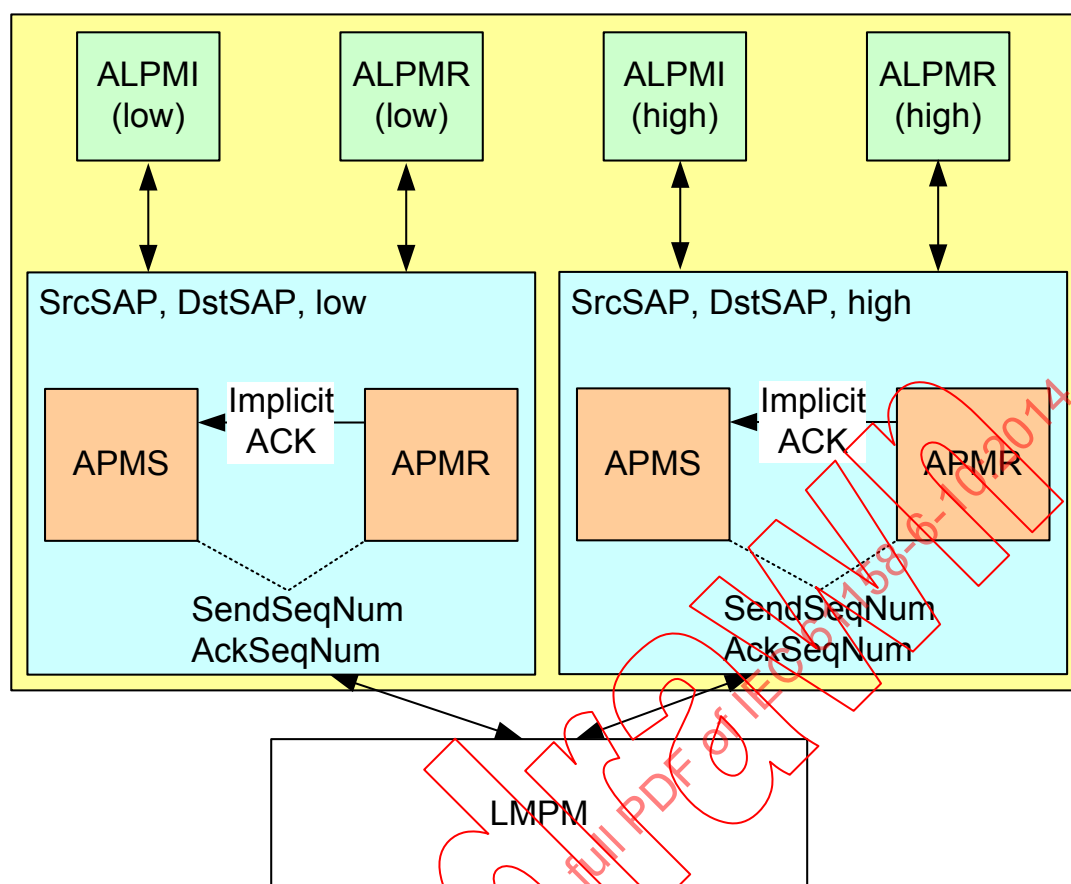
4.8.3 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.8.4 Machines de protocole de relation entre applications

4.8.4.1 Généralités

Le transport RTA symétrique est utilisé par la Machine de Protocole Acyclique (APM) symétrique. L'APM est élaborée comme décrit à la Figure 59. Un canal ALPMI / ALPMR est adressé par Source-Node, Destination-Node, Source-SAP, Destination-SAP et FrameID high ou low.



Légende

Anglais	Français
ALPMI (low)	ALPMI (bas)
ALPMR (low)	ALPMR (bas)
ALPMI (high)	ALPMI (élevé)
SrcSAP, DstSAP, low	SrcSAP, DstSAP, bas
APMS	APMS
Implicit ACK	ACK implicite
APMR	APMR
SrcSAP, DstSAP, high	SrcSAP, DstSAP, élevé
LMPM	LMPM

Figure 59 – Structure de l'APM

4.8.4.2 Acyclic Protocol Machine Sender (Expéditeur de machine de protocole acyclique)

4.8.4.2.1 Généralités

La structure de l'Emetteur de Machine de Protocole Acyclique (APMS) apparaît ci-dessous (voir Figure 60).

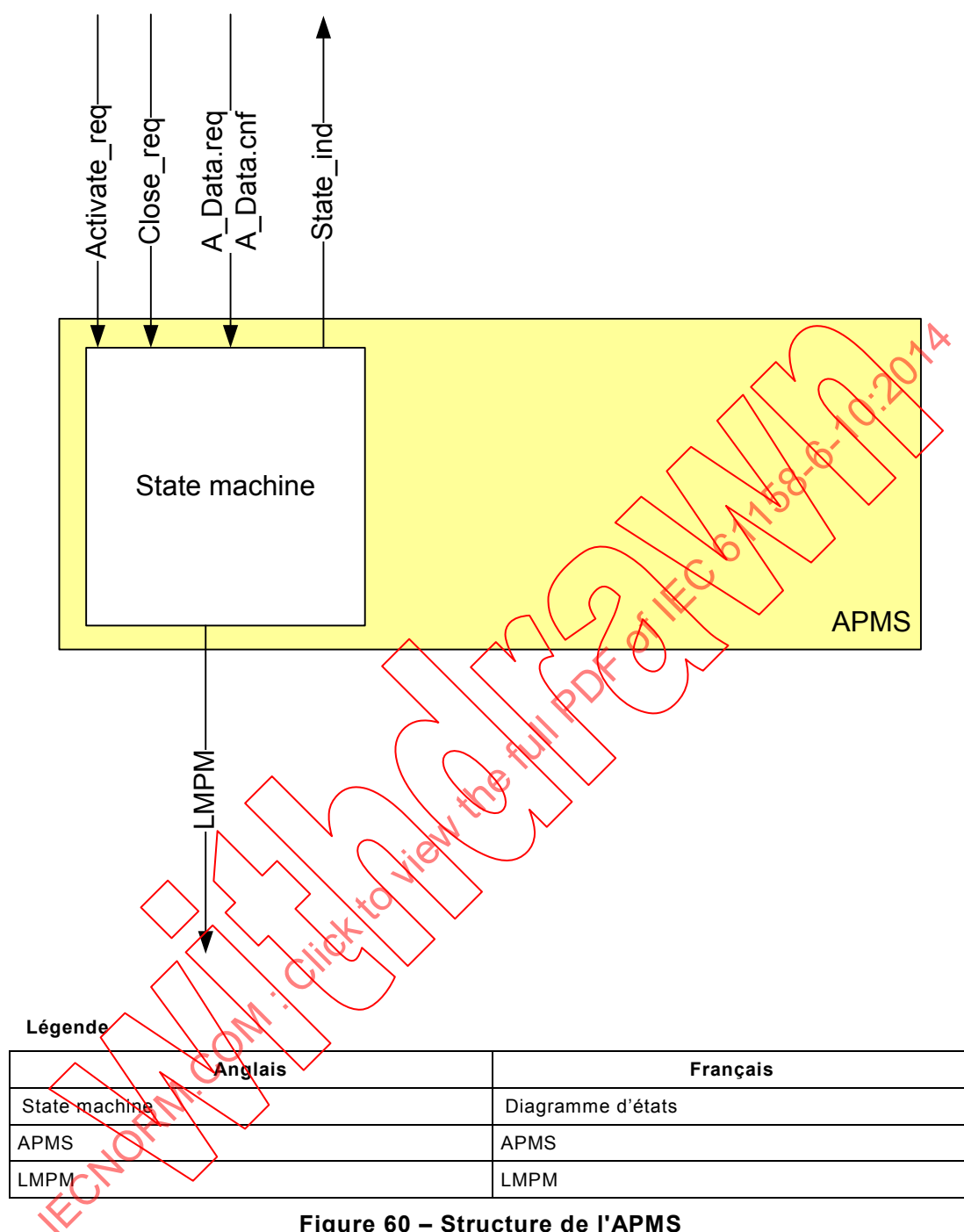


Figure 60 – Structure de l'APMS

4.8.4.2.2 Définitions de primitives

4.8.4.2.2.1 Primitives échangées entre les machines distantes

Les primitives de service distantes, y compris leurs paramètres associés, émises ou reçues par APMS sont décrites dans la définition de service et présentées dans le Tableau 204.

Tableau 204 – Primitives distantes émises ou reçues par APMS

Primitive	Source	Destination	Paramètres associés	Fonctions
APMS_A_Data.cnf (-)	APMS	ALPMI ALPMR	CREP, ERRCLS, ERRCODE	—
APMS_A_Data.cnf (+)	APMS	ALPMI ALPMR	CREP	—
APMS_A_Data.req	ALPMI ALPMR	APMS	CREP, Data	—
LMPM_A_Data.cnf	LMPM	APMS	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	APMS	CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_N_Data.cnf	LMPM	APMS	CREP, LMPM_status	—
LMPM_N_Data.ind	LMPM	APMS	CREP, DA, SA, VLANPrio, VLANId, N_SDU	—
LMPM_A_Data.req	APMS	LMPM	CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_N_Data.req	APMS	LMPM	CREP, DA, SA, VLANPrio, VLANId, N_SDU	—

4.8.4.2.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés émises ou reçues par APMS sont décrites dans la définition de service et présentées dans le Tableau 205.

Tableau 205 – Primitives locales émises ou reçues par APMS

Primitive	Source	Destination	Paramètres associés	Fonctions
APMS_Activate_req	ALPMI ALPMR	APMS	CREP, DA, SA, FrameID, VLANPrio, VLANID, RTATimeoutFactor, RTARetries, Transport, DstIP, SrcIP	—
APMS_Close_req	ALPMI ALPMR	APMS	CREP, ErrCode	—

Primitive	Source	Destination	Paramètres associés	Fonctions
APMS_Close_cnf	APMS	ALPMI ALPMR	CREP	—
APMS_Activate_cnf (-)	APMS	ALPMI ALPMR	CREP, ERRCLS, ERRCODE	—
APMS_Activate_cnf (+)	APMS	ALPMI ALPMR	CREP	—
APMS_Error_ind	APMS	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—

4.8.4.2.3 Schéma de transition d'états

Le schéma de transition d'états de l'APMS est présenté à la Figure 61.

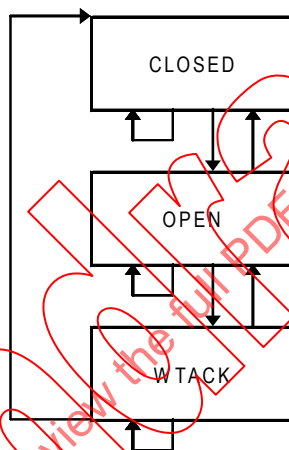


Figure 61 – Schéma de transition d'états d'APMS

Etats de l'APMS

- CLOSED L'état CLOSED indique la nécessité d'une initialisation. Le service Activate met la machine à l'état OPEN.
- OPEN Attend une demande du service A-Data
- WTACK A l'issue de la transmission des données, l'état WTACK permet d'attendre un acquittement de transport. Suite à la réception de l'acquittement, la machine revient à l'état OPEN.

4.8.4.2.4 Description du diagramme d'états

Ce diagramme d'états, adressé par Source-Host, Source-ServiceAccessPoint, Destination-Host, DestinationAccessPoint et Priority, traite le service A-Data. Chaque priorité (Haute et Basse) est gérée de façon indépendante. Ce diagramme d'états fait partie de la paire APMS et APMR utilisée en tant qu'émetteur et répondeur pour les services A-Data d'acquittement des transports. Le service A-Data est utilisé pour le transport de l'Alarme.

4.8.4.2.5 Table d'états APMS

Le Tableau 206 contient la description exhaustive du diagramme d'états d'APMS. Une primitive LPMI_A_Data.ind primitive sera acceptée si la valeur du champ PDUType.Type est DATA-RTA-PDU, ACK-RTA-PDU ou NACK-RTA-PDU et si PDUType.Version est RTA_VERS=1.

Le diagramme d'états d'APMS spécifie la séquence d'arrêt prématuré et montre l'ERR-RTA-PDU correspondant. Ces PDU ne doivent être envoyées que de l'APMS pour les alarmes basse priorité à partir des deux côtés de la connexion.

Tableau 206 – Table d'états APMS

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	CLOSED	APMS_Activate.req () => AlarmInstance[SrcSAP, DstSAP, Prio] := DA, RTATimeoutFactor, RTARetries, ... // NOTE Information used for ALPMI, ALPMR, APMS and APMR Send_Seq_Count:=0xffff Send_Seq_CountO:=0xfffe Use CREP to identify the AlarmInstance APMS_Activate.cnf (+)	OPEN
2	CLOSED	APMS_Close.req () => APMS_Close.cnf ()	CLOSED
3	CLOSED	APMS_A_Data.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_A_Data.cnf (-)	CLOSED
4	CLOSED	A_Data_ind => ignore	CLOSED
5	CLOSED	A_Data_cnf => ignore	CLOSED
6	OPEN	APMS_Activate.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_Activate.cnf (-)	OPEN
7	OPEN	APMS_Close.req () /VLANPrio == Low => PDU.Version:=1 PDU.PDUType:=ERR PDU.AddFlags.TACK:=0 PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.PNIOStatus.ErrorCode = 0xCF PDU.PNIOStatus.ErrorDecode = 0x81 PDU.PNIOStatus.ErrorCode1 = 0xFD PDU.PNIOStatus.ErrorCode2 = ErrCode from APMS_Close request A_Data_req StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
8	OPEN	APMS_Close.req () /VLANPrio == High => //NOTE Error PDU only for the APMS instance "low" StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
9	OPEN	APMS_A_Data.req () => PDU.Version:=1 PDU.PDUType:=DATA PDU.AddFlags.TACK:=Tack PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.RTA-SDU:=Data Store PDU.Flags, PDU.RTA-SDU Retry:= RTARetries A_Data_req	WTACK
10	OPEN	A_Data_ind /PDU.PDUType == DATA PDU.PDUType == NAK PDU.PDUType == ACK => ignore	OPEN
11	OPEN	A_Data_ind /PDU.Type == ERR => ERRCLS := PDU.PNIOStatus.ErrorCode1 ERRCODE := PDU.PNIOStatus.ErrorCode2 APMS_Error_ind ()	OPEN
12	OPEN	A_Data_cnf => ignore	OPEN
13	OPEN	TimerExpired => ignore	OPEN
14	WTACK	APMS_Activate.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_Activate.cnf (-)	WTACK
15	WTACK	APMS_Close.req () /VLANPrio == Low => PDU.Version:=1 PDU.PDUType:=ERR PDU.AddFlags:=0 PDU.Window:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.PNIOStatus.ErrorCode = 0xCF PDU.PNIOStatus.ErrorDecode = 0x81 PDU.PNIOStatus.ErrorCode1 = 0xFD PDU.PNIOStatus.ErrorCode2 = ErrCode from APMS_Close request A_Data_req StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
16	WTACK	APMS_Close.req () /VLANPrio == High => StopTimer (RTATimeout) APMS_Close.cnf ()	CLOSED
17	WTACK	APMS_A_Data.req () => ERRCLS := APMS ERRCODE := INVALID_STATE APMS_A_Data.cnf (-)	WTACK

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
18	WTACK	A_Data_ind /(PDU.PDUType == DATA PDU.PDUType == ACK) && PDU.AckSeqNum == Send_Seq_Count => Send_Seq_CountO:=Send_Seq_Count Send_Seq_Count:= (Send_Seq_Count+1) & 0x7fff //NOTE TACK shall be done by ACK or DATA APMS_A_Data.cnf (+)	OPEN
19	WTACK	A_Data_ind /(PDU.PDUType == DATA PDU.PDUType == ACK) && PDU.AckSeqNum != Send_Seq_Count => ignore	WTACK
21	WTACK	A_Data_ind /PDU.Type == ERR => APMS_Error_ind ()	WTACK
21	WTACK	A_Data_ind /PDU.PDUType == NAK => ignore	WTACK
22	WTACK	A_Data_cnf => StartTimer (RTATimeout)	WTACK
23	WTACK	TimerExpired / Retry != 0 => PDU.Version:=1 PDU.PDUType:=DATA PDU.AddFlags.TACK:=from Stored PDU.AddFlags PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_Count PDU.AckSeqNum:=Expected_Seq_CountO (from APMR) PDU.RTA-SDU:=from Stored PDU.RTA-SDU Retry:= Retry-1 A_Data_req	WTACK
24	WTACK	TimerExpired /Retry == 0 => ERRCLS = APMS ERRCODE := TIMEOUT APMS_Error_ind () APMS_A_Data.cnf (-)	OPEN

4.8.4.2.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 207 contient les fonctions, macros, temporisateurs et variables utilisés par l'APMS, ainsi que leurs arguments et leurs descriptions.

Tableau 207 – Fonctions, Macros, Temporisateur et Variables utilisés par l'APMS

Nom	Type	Fonction/Signification
StartTimer	Fonction	Cette fonction est utilisée pour lancer ou relancer un temporisateur.
StopTimer	Fonction	Cette fonction est utilisée pour arrêter un temporisateur.
TimerExpired	Fonction	Cette fonction signale qu'un temporisateur a expiré.

Nom	Type	Fonction/Signification
A_Data_cnf	Macro	si (Transport == RTC) LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) si (Transport == UDP) LMPM_N_Data.cnf (CREP, D_Port, Tstamp, LMPM_status)
A_Data_ind	Macro	si (Transport == RTC) LMPM_A_Data.ind (CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU) PDU := A_SDU si (Transport == UDP) LMPM_N_Data.ind (CREP, DA, SA, VLANPrio, VLANID, N_SDU) PDU := N_SDU
A_Data_req	Macro	si (Transport == RTC) A_SDU := PDU LMPM_A_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, A_SDU) si (Transport == UDP) N_SDU := PDU LMPM_N_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, N_SDU)
RTATimeout	Temporisateur	Ce temporisateur suit l'acquittement de transport du répondeur
ERRCLS	Variable	Cette variable locale contient l'entité signalant l'erreur.
ERRCODE	Variable	Cette variable locale contient la raison de l'erreur dans le contexte du ERRCLS.
Expected_Seq_CountO	Variable	Cette variable locale contient la précédente valeur du compteur de la variable Seq_Count.
RTARetries	Variable	Cette variable locale contient le facteur de retransmission maximal d'ASE RTA. La valeur par défaut est 3.
Retry	Variable	Cette variable locale est utilisée comme compteur de travail pour les retransmissions.
RTATimeoutFactor	Variable	Cette variable locale contient la valeur provenant d'ASE RTA. La valeur par défaut est 100 ms.
Send_Seq_Count	Variable	Cette variable locale contient la valeur de compteur qui doit être utilisée lors de l'acheminement de la DATA-RTA-PDU suivante.

4.8.4.3 Acyclic Protocol Machine Receiver (Destinataire de machine de protocole acyclique)

4.8.4.3.1 Généralités

La structure du Destinataire de Machine de Protocole Acyclique (APMR) apparaît ci-dessous (voir Figure 62).

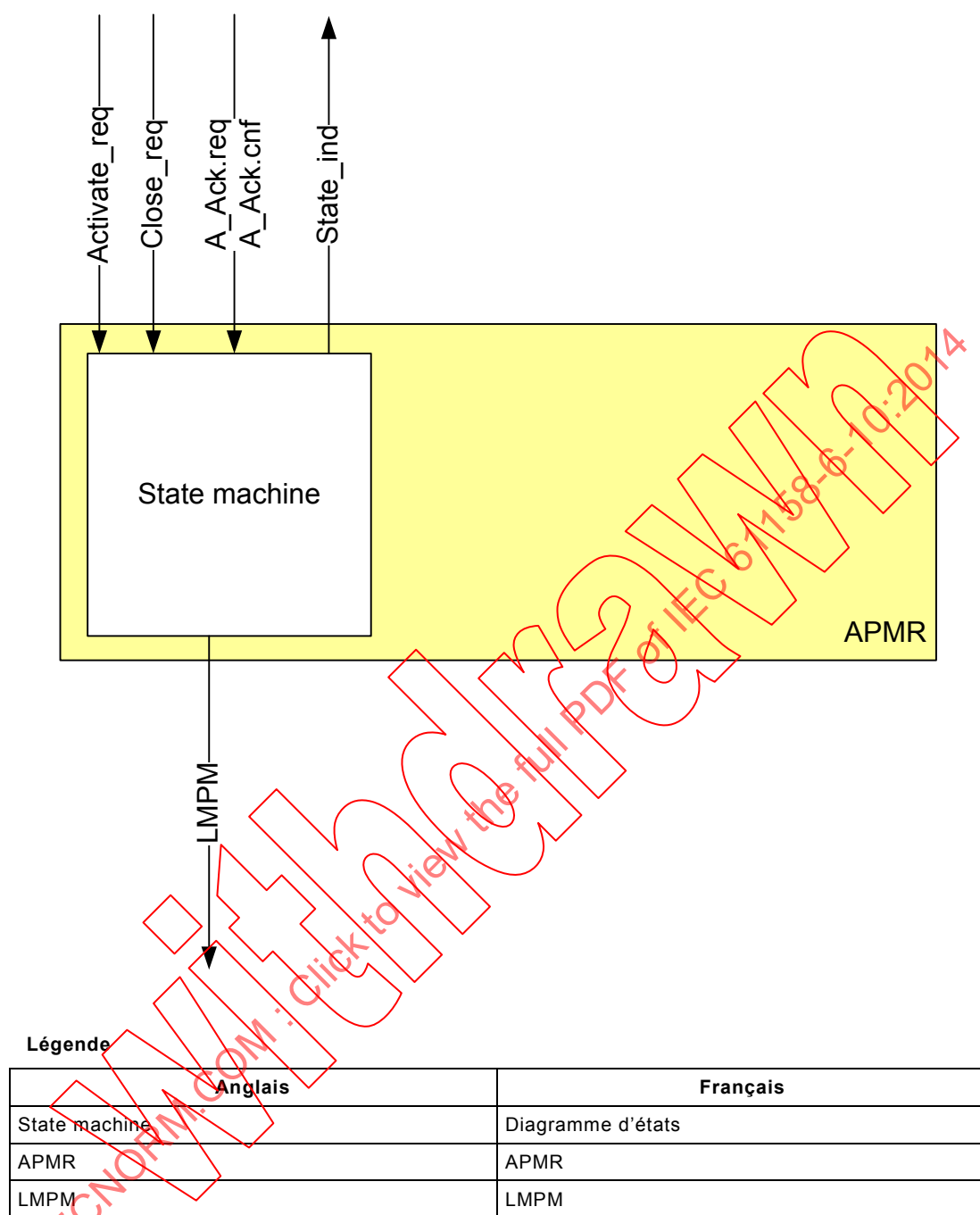


Figure 62 – Structure de l'APMR

4.8.4.3.2 Définitions de primitives

4.8.4.3.2.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par APMR sont décrites dans la définition de service et présentées dans le Tableau 209.

Tableau 208 – Primitives distantes émises ou reçues par APMR

Primitive	Source	Destination	Paramètres associés	Fonctions
APMR_A_Data.ind	APMR	ALPMR	CREP, Data	Destination utilisée si Data == AlarmNotification-PDU Data == unknown
APMR_A_Data.ind	APMR	ALPMI	CREP, Data	Destination utilisée si Data == Alarm-Ack-PDU Data == unknown
LMPM_A_Data.cnf	APMR	LMPM	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_A_Data.ind	APMR	LMPM	CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_A_Data.req	APMR	LMPM	CREP, D_Port:=AUTO, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_N_Data.cnf	APMR	LMPM	CREP, D_Port, Tstamp, LMPM_status	—
LMPM_N_Data.ind	APMR	LMPM	CREP, DA, SA, VLANPrio, VLANID, N_SDU	—
LMPM_N_Data.req	APMR	LMPM	CREP, D_Port:=AUTO, DA, SA, VLANPrio, VLANID, N_SDU	—

4.8.4.3.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par CPM sont décrites dans la définition de service et présentées dans le Tableau 209.

Tableau 209 – Primitives locales émises ou reçues par APMR

Primitive	Source	Destination	Paramètres associés	Fonctions
APMR_Activate_req	ALPMI	APMR	CREP, DA, SA, FrameID, VLANPrio, VLANID, Transport, DstIP, SrcIP	—
APMR_Close_req	ALPMI	APMR	CREP	—

Primitive	Source	Destination	Paramètres associés	Fonctions
APMR_Activate_cnf (-)	APMR	ALPMI	CREP, ERRCLS, ERRCODE	—
APMR_Activate_cnf (+)	APMR	ALPMI	CREP	—
APMR_Close_cnf (+)	APMR	ALPMI	CREP	—
APMR_Error_ind	APMR	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—

4.8.4.3.3 Schéma de transition d'états

Le diagramme de transition d'états de l'APMR est présenté à la Figure 63.

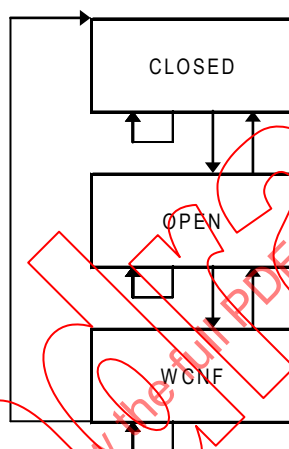


Figure 63 – Schéma de transition d'états d'APMR

Etats de l'APMR

CLOSED	L'état CLOSED indique la nécessité d'une initialisation. Le service Activate met la machine à l'état OPEN.
OPEN	Attend une indication de service LMPM_A_Data ou UDP_A_Data et achemine un acquittement de transport.
WCNF	Attend la confirmation de l'acquittement de transport et émet une indication de APMR_Data.

4.8.4.3.4 Description du diagramme d'états

Ce diagramme d'états, adressé par Source-Host, Source-ServiceAccessPoint, Destination-Host, DestinationAccessPoint et Priority, traite le service A-Data. Chaque priorité (Haute et Basse) est gérée de façon indépendante. Ce diagramme d'états fait partie de la paire APMS et APMR utilisée en tant qu'émetteur et répondeur pour les services A-Data d'acquittement des transports. Le service A-Data est utilisé pour le transport de l'Alarme.

4.8.4.3.5 Table d'états APMR

Le Tableau 210 contient la description exhaustive du diagramme d'états d'APMR. Une primitive LMPM_A_Data.ind sera acceptée:

- Si le champ PDUType.Type est DATA-RTA-PDU, le champ PDUType.Version est un, VarPartLen supérieur à zéro et que AddFlagsWindow est un.

- Si le champ PDUType.Type est ERR-RTA-PDU, le champ PDUType.Version est un et que VarPartLen est 4.

Tableau 210 – Table d'états APMR

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	CLOSED	APMR_Activate.req () => AlarmInstance[SrcSAP, DstSAP, Prio] := DA, RTATimeoutFactor, RTARetries, ... // NOTE Information used for ALPMI, ALPMR, APMS and APMR Expected_Seq_Count:=0xffff Expected_Seq_CountO:=0xfffe Use CREP to identify the AlarmInstance APMR_Activate.cnf (+)	OPEN
2	CLOSED	APMR_Close.req () => APMR_Close.cnf (+)	CLOSED
3	CLOSED	A_Data_ind => ignore	CLOSED
4	CLOSED	A_Data_cnf => ignore	CLOSED
5	OPEN	APMR_Activate.req () => ERRCLS := APMR ERRCODE :=INVALID_STATE APMR_Activate.cnf (-)	OPEN
6	OPEN	APMR_Close.req () => APMR_Close.cnf (+)	CLOSED
7	OPEN	A_Data_ind /PDU.Type == DATA && PDU.AddFlags.Tack && PDU.SendSeqNum == Expected_Seq_Count => Store Data := PDU.RTA-SDU Expected_Seq_CountO :=Expected_Seq_Count Expected_Seq_Count := (Expected_Seq_Count+1) & 0x7fff PDU.Version :=1 PDU.Type := ACK PDU.AddFlags.TACK :=0 PDU.AddFlags.WindowSize :=1 PDU.SendSeqNum := Send_Seq_CountO (from the APMS) PDU.AckSeqNum := Expected_Seq_CountO //NOTE Le PDU.SendSeqNum doit être mis en place avec la valeur de l'APMS correspondant A_Data_req	WCNF
8	OPEN	A_Data_ind /PDU.Type == DATA && PDU.AddFlags.Tack && PDU.SendSeqNum == Expected_Seq_CountO => PDU.Version:=1 PDU.Type:= ACK PDU.AddFlags.TACK:=0 PDU.AddFlags.WindowSize:=1 PDU.SendSeqNum:=Send_Seq_CountO (from the APMS) PDU.AckSeqNum:=Expected_Seq_CountO A_Data_req	OPEN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
9	OPEN	A_Data_ind /PDU.Type == DATA && PDU.AddFlags.Tack && PDU.SendSeqNum != Expected_Seq_Count && PDU.SendSeqNum != Expected_Seq_CountO => ERRCLS := RTA_ERR_CLS_PROTOCOL ERRCODE := RTA_ERR_CODE_SEQ PDU.Version:=1 PDU.Type:= NAK PDU.AddFlags.TACK:=0 PDU.AddFlags.Window:=1 PDU.SendSeqNum:=Send_Seq_CountO (from the APMS) PDU.AckSeqNum:=Expected_Seq_CountO A_Data_req	OPEN
10	OPEN	A_Data_ind /PDU.Type == ERR => ERRCLS := PDU.PNIOStatus.ErrorCode1 ERRCODE := PDU.PNIOStatus.ErrorCode2 APMR_Error_ind ()	OPEN
11	OPEN	A_Data_ind /(PDU.Type == DATA && !PDU.AddFlags.Tack) => ignore	OPEN
12	OPEN	A_Data_ind /PDU.Type == ACK PDU.Type == NAK => ignore	OPEN
13	OPEN	A_Data_cnf => ignore	OPEN
14	WCNF	APMR_Activate.req () => ERRCLS := APMR ERRCODE :=INVALID_STATE APMR_Activate.cnf (-)	WCNF
15	WCNF	APMR_Close.req () => APMR_Close.cnf (+)	CLOSED
16	WCNF	A_Data_cnf => Retrieve Data := PDU.RTA-SDU APMR_A_Data.ind ()	OPEN
17	WCNF	A_Data_ind /PDU.Type != ERR => ignore	WCNF
18	WCNF	A_Data_ind /PDU.Type == ERR => APMR_Error_ind ()	WCNF

4.8.4.3.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 211 contient les fonctions, macros, temporisateurs et variables utilisés par l'APMR ainsi que leurs arguments et leurs descriptions.

Tableau 211 – Fonctions, Macros, Temporisateurs et Variables utilisés par l'APMR

Nom	Type	Signification
A_Data_ind	Macro	si (Transport == RTC) LMPM_A_Data.ind(CREP, S_Port, Tstamp, DA, SA, VLANPrio, VLANID, A_SDU) PDU := A_SDU si (Transport == UDP) LMPM_N_Data.ind(CREP, DA, SA, VLANPrio, VLANid, N_SDU) PDU := N_SDU
A_Data_cnf	Macro	si (Transport == RTC) LMPM_A_Data.cnf (CREP, D_Port, Tstamp, LMPM_status) si (Transport == UDP) LMPM_N_Data.cnf (CREP, D_Port, Tstamp, LMPM_status)
A_Data_req	Macro	si (Transport == RTC) A_SDU := PDU LMPM_A_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, A_SDU) si (Transport == UDP) N_SDU := PDU LMPM_N_Data.req (CREP, D_Port := AUTO, DA, SA, VLANPrio, VLANID, N_SDU)
PDU	Variable	Variable locale représentant UDP-RTA-PDU ou RTA-PDU en fonction du transport. Utilisée pour afficher la configuration spécifique au protocole et la validation des champs de RTA. La configuration et la validation de PDU-Parameters supplémentaires (IP-header, par exemple) ne sont pas présentées. Le PDUType apparaît au Tableau 202.
Expected_Seq_Count	Variable	Cette variable locale contient la valeur du compteur qui doit être utilisée à la réception du prochain DATA PDU.
Expected_Seq_CountO	Variable	Cette variable locale contient la précédente valeur du compteur de la variable Seq_Count.

4.8.5 Machines de Protocole de Mapping DLL

Il n'existe aucune Machine de Protocole de Mapping DLL (DMPM) définie pour ce Protocole.

4.9 Fragmentation

4.9.1 Généralités

La LengthOfPeriod est limitée sur l'extrémité inférieure par la taille maximale d'une trame Ethernet. La fragmentation réduit, avec une approche entre homologues, la taille acheminée d'une trame Ethernet et permet d'utiliser une LengthOfPeriod jusqu'à 31,25 µs.

Elle utilise TagControllInformation.Priority défini dans l'IEEE 802.1Q pour encoder et décoder jusqu'à 8 (au minimum un doit être pris en charge) flux de fragmentation parallèles. En raison du groupage avec l'IEEE 802.1Q la granularité des flux pris en charge doit être d'une, de deux, de quatre ou de huit et l'affectation à TagControllInformation.Priority apparaît dans le Tableau 212.

Tableau 212 – TagControllInformation.Priority en fonction des flux

Priorité	Un flux	Deux flux	Quatre flux	Huit flux
7	Par défaut Toutes les priorités sont affectées à un flux	Les quatre priorités les plus hautes sont affectées à un flux	Deux priorités sont affectées à un flux	Une priorité est un flux
6				Une priorité est un flux
5			Deux priorités sont affectées à un flux	Une priorité est un flux
4				Une priorité est un flux

Priorité	Un flux	Deux flux	Quatre flux	Huit flux
3		Les quatre priorités les plus basses sont affectées à un flux	Deux priorités sont affectées à un flux	Une priorité est un flux
2				Une priorité est un flux
1			Deux priorités sont affectées à un flux	Une priorité est un flux
0				Une priorité est un flux

Même pour les trames, générées par la fragmentation, la limite de taille de trame la plus petite définie dans l'IEEE 802.3 s'applique. Par conséquent, cela limite l'utilisation du principe de fragmentation conformément à la Figure O.2, à la Formule (39), à la Formule (40), à la Formule (41) et à la Formule (42), telle que présentée au Tableau 213.

La Formule (39) suivante présente le calcul pour une seule trame fragmentée dans un premier et un dernier fragment.

NOTE La LowestFractionizeableFrameSize possède deux valeurs différentes pour les trames avec ou sans VLAN. Par conséquent, les données utiles de fragmentation sont définies et utilisées comme une simplification.

$$\text{LowestFractionizeableFrameSize} = \text{FrameDataFromFirstFragment} + \text{FrameDataFromLastFragment} + \text{ProtocolOverhead} \quad (39)$$

Avec VLAN

$$\text{LowestFractionizeableFrameSize} = 40 \text{ octets} + 38 \text{ octets} + 20 \text{ octets}$$

$$\text{LowestFractionizeableFrameSize} = 98 \text{ octets}$$

Sans VLAN

$$\text{LowestFractionizeableFrameSize} = 48 \text{ octets} + 42 \text{ octets} + 16 \text{ octets}$$

$$\text{LowestFractionizeableFrameSize} = 106 \text{ octets}$$

où

LowestFractionizeableFrameSize	est la longueur de trame minimale pouvant être fractionnée
FrameDataFromFirstFragment	est la quantité de données de la trame d'origine acheminée par le premier fragment
FrameDataFromLastFragment	est la quantité de données de la trame d'origine acheminée par le dernier fragment
ProtocolOverhead	est la quantité de données de la trame de protocole Ethernet

La Formule (40) suivante présente le calcul des données utiles restantes minimales d'une trame fragmentée dans un fragment médian ou dans un dernier fragment.

NOTE La LowestFractionizeableFrameSize possède deux valeurs différentes pour les trames avec ou sans VLAN. Par souci de simplicité, seule la valeur avec VLAN est utilisée.

$$\text{LowestFractionizeableFrameSize} = \text{FrameDataFromMiddleFragment} + \text{FrameDataFromLastFragment} + \text{ProtocolOverhead} \quad (40)$$

Avec VLAN

$$\text{LowestFractionizeableFrameSize} = 40 \text{ octets} + 38 \text{ octets} + 26 \text{ octets}$$

LowestFractionizeableFrameSize = 104 octets

Sans VLAN

LowestFractionizeableFrameSize = 48 octets + 42 octets + 22 octets

LowestFractionizeableFrameSize = 112 octets

où

LowestFractionizeableFrameSize	est la longueur de trame minimale pouvant être fractionnée
FrameDataFromMiddleFragment	est la quantité de données résiduelles de la trame d'origine acheminée par un fragment médian
FrameDataFromLastFragment	est la quantité de données de la trame d'origine acheminée par le dernier fragment
ProtocolOverhead	est la quantité de données de la trame de protocole Ethernet

La Formule (41) présente une trame utilisée pour le premier fragment, le fragment médian ou un dernier fragment avec une étiquette VLAN.

Taille de trame minimale (avec VLAN) = DestinationMAC + SourceMAC + VLAN + FragmentationHeader + Payload + FCS (41)

Payload (avec VLAN) = Taille de trame minimale – (DestinationMAC + SourceMAC + VLAN + FragmentationHeader + FCS)

Payload (avec VLAN) = 64 – (6 + 6 + 4 + 6 + 4)

Payload (avec VLAN) = 38

conformément aux règles pour FragStatus.MoreFollows:=D'autres fragments suivent

Payload (avec VLAN) = 40

conformément aux règles pour FragStatus.MoreFollows:=Dernier fragment

Payload (avec VLAN) = 38

où

Lowest frame size	est la longueur minimale autorisée d'une trame (64 octets)
DestinationMAC	est la longueur de l'adresse MAC (6 octets)
SourceMAC	est la longueur de l'adresse MAC (6 octets)
VLAN	est la longueur de l'étiquette VLAN (4 octets)
FragmentationHeader	est la longueur de l'en-tête de fragmentation (6 octets)
Payload	est la longueur des données utiles
FCS	est la longueur de FCS (4 octets)

La Formule (42) présente une trame utilisée pour le premier fragment, le fragment médian ou un dernier fragment avec une étiquette VLAN.

Taille de trame minimale(sans VLAN) = DestinationMAC + SourceMAC + FragmentationHeader + Payload + FCS (42)

Payload (sans VLAN) = Taille de trame minimale – (DestinationMAC + SourceMAC + FragmentationHeader + FCS)

Payload (sans VLAN) = 64 – (6 + 6 + 6 + 4)

Payload (sans VLAN) = 42

conformément aux règles pour FragStatus.MoreFollows:=D'autres fragments suivent

Payload (sans VLAN) = 48

conformément aux règles pour FragStatus.MoreFollows:=Dernier fragment

Payload (sans VLAN) = 42

où

Lowest frame size	est la longueur minimale autorisée d'une trame (64 octets)
DestinationMAC	est la longueur de l'adresse MAC (6 octets)
SourceMAC	est la longueur de l'adresse MAC (6 octets)
VLAN	est la longueur de l'étiquette VLAN (4 octets)
FragmentationHeader	est la longueur de l'en-tête de fragmentation (6 octets)
Payload	est la longueur des données utiles
FCS	est la longueur de FCS (4 octets)

Tableau 213 – Limite inférieure des fragments

Cas	VLAN	Taille de Payload (données utiles) des fragments	Signification
Premier + dernier	Avec	Données utiles du premier fragment:= 40 octets Données utiles du dernier fragment:= 38 octets Partie du protocole:= 20 octets	Plus petite taille de trame du fragment capable:= DA + SA + VLAN + Données utiles + FCS:= 98 octets
	Sans	Données utiles du premier fragment:= 48 octets Données utiles du dernier fragment:= 42 octets Partie du protocole:= 16 octets	Plus petite taille de trame du fragment capable:= DA + SA + Données utiles + FCS:= 106 octets
Médian + dernier	Avec	Données utiles du fragment médian:= 40 octets Données utiles du dernier fragment:= 38 octets Partie du protocole:= 26 octets	Plus petite taille de données utiles résiduelles du fragment capable:= DA + SA + VLAN + FragHeader + Données utiles + FCS:= 104 octets
	Sans	Données utiles du fragment médian:= 48 octets Données utiles du dernier fragment:= 42 octets Partie du protocole:= 22 octets	Plus petite taille de données utiles résiduelles du fragment capable:= DA + SA + FragHeader + Données utiles + FCS:= 112 octets

Des données utiles de fragmentation de 90 octets pour tous les types de trames (avec ou sans VLAN) peuvent être utilisées pour simplifier la mise en œuvre de la fragmentation. Par conséquent, la taille de trame minimale qui pourrait encore être fragmentée doit être de 110 octets avec VLAN et de 106 octets sans VLAN et les données utiles de fragmentation résiduelles minimales qui pourraient encore être fragmentées doivent être de 116 octets avec VLAN et de 112 octets sans VLAN.

4.9.2 Description de la syntaxe FRAG

4.9.2.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.9.2.2 Syntaxe abstraite FRAG APDU

Le Tableau 214 définit la syntaxe abstraite des PDU de couche application appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 214 doivent représenter le contenu de la DLSU dans le Tableau 5.

Tableau 214 – Syntaxe FRAG APDU

Nom de l'APDU	Structure de l'APDU
FRAG-PDU	FRAG-DATA-PDU

Le Tableau 215 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 214.

Tableau 215 – Substitutions FRAG

Nom de substitution	Structure
FRAG-DATA-PDU	FragDataLength, FragStatus, FragData

4.9.3 Syntaxe de transfert FRAG

4.9.3.1 Section de codage relative aux champs spécifiques de FRAG-PDU

4.9.3.1.1 Codage du champ FragDataLength

Le champ FragDataLength est codé en Unsigned8 et contient le nombre de parties de 8 octets des données du fragment. Pour le dernier fragment (FragStatus.MoreFollows:= 0), jusqu'à 7 octets "restants" peuvent être ajoutés.

FragDataLength doit être calculée à l'aide de la Formule (43).

$$\text{FragDataLength} = (\text{Length of FragData} \text{ DIV } 8) \quad (43)$$

où

FragDataLength	est la longueur du champ FragData en parties de 8 octets
Length of FragData	est la longueur du champ FragData en octets
8	est la valeur définie d'une incrémentation de FragData

Si la fragmentation d'une trame en plusieurs parties de 8 octets n'a aucun reste, alors la taille du dernier fragment contenant FragData doit constituer le reste, comportant les octets restants.

Le destinataire doit accepter une trame fragmentée avec ce type de FragData si FragStatus.MoreFollows:= 0.

Ce champ doit être codé en Unsigned8 avec les valeurs conformes au Tableau 216.

Tableau 216 – FragDataLength

Valeur (hexadécimale)	Signification	Utilisation
0x00, ..., 0x04	—	Réservé
0x05 ou 0x06 ^a	Minimum Si FragStatus.MoreFollows:= 1, alors la longueur de FragData est de 0x30 octets Si FragStatus.MoreFollows:= 0, alors la longueur de FragData est de 0x30 à 0x37 octets (jusqu'à 7 octets restants)	Obligatoire
0x08, ..., 0xBE	...	Obligatoire
0xBF ^b	Maximum Si FragStatus.MoreFollows:= 1, alors la longueur de FragData est de 0x5CF octets Si FragStatus.MoreFollows:= 0, alors la longueur de FragData est de 0x5F8 à 0x5FF octets	Obligatoire
0xC0, ..., 0xFF	La longueur maximale de FragData est 0x7FF	Facultatif
^a La FragDataLength minimale qui doit être prise en charge par la DEFRAG est de 0x05 avec VLAN et de 0x06 sans VLAN. ^b La FragDataLength maximale qui doit être prise en charge par la DEFRAG est définie par l'utilisation d'au moins deux fragments pour une trame maximale définie dans l'IEEE 802.3. La prise en charge de la fragmentation en un fragment n'est pas voulue.		

4.9.3.1.2 Codage du champ FragStatus

Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0 – 5: FragStatus.FragmentNumber

Ce champ contient le numéro du fragment et doit être codé conformément au Tableau 217.

Tableau 217 – FragStatus.FragmentNumber

Valeur (hexadécimale)	Signification
0x00	Premier fragment d'une trame
0x01	Deuxième fragment d'une trame
...	...
0x3F	64ème fragment d'une trame

Bit 6: FragStatus.Reserved

Ce champ doit être codé conformément à Tableau 218.

Tableau 218 – FragStatus.Reserved

Valeur (hexadécimale)	Signification
0x00	Réservé

Bit 7: FragStatus.MoreFollows

Ce champ doit être codé conformément à Tableau 219.

Tableau 219 – FragStatus.MoreFollows

Valeur (hexadécimale)	Signification
0x00	Dernier fragment
0x01	D'autres fragments suivent

4.9.3.1.3 Codage du champ FragData

Ce champ doit être codé en type de données OctetString et doit contenir la part de données transmise de la trame fragmentée.

4.9.4 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.9.5 Machines de protocole de relation entre applications

Il n'existe aucune Machine de Protocole de relation entre applications (ARPM) définie pour ce Protocole.

4.9.6 Machines de Protocole de Mapping DLL**4.9.6.1 Fragmentation****4.9.6.1.1 Fragmentation dynamique****4.9.6.1.1.1 Définitions de primitives****4.9.6.1.1.1.1 Primitives échangées entre les machines distantes**

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par FRAG_D sont décrites dans la définition de service et présentées dans le Tableau 220.

Tableau 220 – Primitives distantes émises ou reçues par FRAG_D

Primitive	Source	Destination	Paramètres associés	Fonctions
QueueHandler.req	FRAG_D	QueueHandler	Queue, Frame	—

4.9.6.1.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par FRAG_D sont décrites dans la définition de service et présentées dans le Tableau 221.

Tableau 221 – Primitives locales émises ou reçues par FRAG_D

Primitive	Source	Destination	Paramètres associés	Fonctions
SCHEDULER_StartOfPeriod	SCHEDULER	FRAG	—	Cette fonction locale est émise si un nouveau cycle de communication commence.

4.9.6.1.1.2 Schéma de transition d'états

Le schéma de transition d'états du FRAG_D est présenté à la Figure 64.

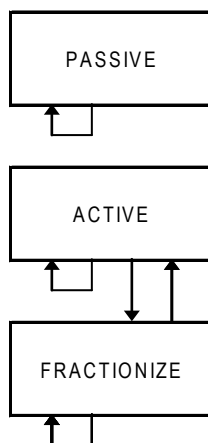


Figure 64 – Schéma de transition d'états de FRAG_D

Etats du FRAG_D

PASSIVE

L'unité FRAG n'est pas utilisée. Transmissions des trames non affectées.

ACTIVE

L'unité FRAG pour le flux défini est active et attend les trames pour les fractionner si nécessaire.

FRACTIONIZE

Une trame est fragmentée et acheminée. Les trames à haute priorité qui ne nécessitent pas de fragmentation peuvent dépasser les fragments. Après l'envoi du dernier fragment, l'état ACTIVE est atteint. Si un temps résidentiel est disponible, la fragmentation commence immédiatement avec la trame suivante.

4.9.6.1.1.3 Description du diagramme d'états

Le FRAG est chargé de la fragmentation des trames selon la période VERTE résiduelle. Il récupère les trames de QueueHandler, les fragmente, si cela est nécessaire et permis par le protocole, et stocke les fragments dans les files d'attente inférieures de QueueHandler.

A l'état inactif, FRAG doit être évité par QueueHandler.

La fragmentation peut être effectuée selon deux stratégies différentes. La première est la fragmentation dynamique, qui utilise la phase VERTE résidentielle alors que la seconde est la fragmentation statique, qui utilise une taille de fragment prédéfinie.

4.9.6.1.1.4 Table d'états FRAG_D (fragmentation dynamique)

Le Tableau 222 contient la table d'états FRAG_D.

Tableau 222 – Table d'états FRAG_D (dynamique)

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	PASSIVE	=> ignore	PASSIVE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
2	ACTIVE	QueuesNotEmpty () /Period == RED => ignore	ACTIVE
3	ACTIVE	QueuesNotEmpty () /Period == GREEN => GetFrameFromQueue () StartFragStream ()	FRACTIONIZE
4	FRACTIONIZE	QueuesNotEmpty () => ignore	FRACTIONIZE
5	FRACTIONIZE	StartFragStream () /Period == GREEN && FrameFitsIntoResidualPeriod == TRUE => QueueHandler.req () //NOTE Envoi de la trame	ACTIVE
6	FRACTIONIZE	StartFragStream () /Period == GREEN && FrameFitsIntoResidualPeriod == FALSE && FrameSizeAllowsFragmentation == TRUE && ResidualPeriodShorterThanMin == TRUE => ignore //NOTE Attend jusqu'à ce que la prochaine période VERTE commence	FRACTIONIZE
7	FRACTIONIZE	SCHEDULER_StartOfPeriod () /Period == GREEN && ResidualPeriodShorterThanMin == TRUE => ignore	FRACTIONIZE
8	FRACTIONIZE	StartFragStream () /Period == GREEN && FrameFitsIntoResidualPeriod == FALSE && FrameSizeAllowsFragmentation == TRUE && ResidualPeriodShorterThanMin == FALSE && LastFragment == FALSE => CalculateFragSize () CreateFrag () QueueHandler.req () //NOTE Envoi du fragment de trame	FRACTIONIZE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
9	FRACTIONIZE	SCHEDULER_StartOfPeriod () /Period == GREEN && ResidualPeriodShorterThanMin == FALSE && LastFragment == TRUE => CalculateFragSize () CreateFrag () QueueHandler.req () //NOTE Envoi du fragment de trame	ACTIVE
10	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameFitsIntoResidualPeriod == TRUE => GetFrameFromQueue () QueueHandler.req ()	FRACTIONIZE
11	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameFitsIntoResidualPeriod == FALSE => ignore	FRACTIONIZE

4.9.6.1.1.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 223 contient les fonctions, macros, temporisateurs et variables utilisés par le FRAG_D ainsi que leurs arguments et leurs descriptions.

Tableau 223 – Fonctions, Macros, Temporisateurs et Variables utilisés par le FRAG_D (dynamique)

Nom	Type	Fonction/Signification
CalculateFragSize	Fonction	Cette fonction locale calcule la taille des fragments en fonction de la période VERTE résiduelle et des règles de fragmentation. Si la fragmentation dynamique n'est pas prise en charge, utiliser une longueur par défaut donnée par l'ingénierie selon la période VERTE minimale de toutes les phases.
CheckForFrameWithHigherPriority	Fonction	Cette fonction locale vérifie si une trame avec une priorité plus haute que celle fragmentée est stockée dans les files d'attente.
CreateFrag	Fonction	Cette fonction locale crée un fragment selon les règles de codage et les règles de fragmentation. Soit premier, soit suivant soit dernier. Garder à l'esprit que les tailles de trame minimale et maximale d'une trame de l'IEEE 802.3 sont toujours valides, même pour les fragments.
FrameFitsIntoResidualPeriod	Fonction	Cette fonction locale vérifie si la période VERTE résiduelle est plus longue que celle requise par la trame pour l'acheminement
FrameSizeAllowsFragmentation	Fonction	Cette fonction locale vérifie si la trame est suffisamment grande pour la fragmentation
GetFrameFromQueue	Fonction	Cette fonction locale extrait la trame suivante selon la quantité de flux FRAG et les priorités.
QueuesNotEmpty	Fonction	Cette fonction locale est exécutée si une entrée est stockée dans les files d'attente.

Nom	Type	Fonction/Signification
ResidualPeriodShorterThanMin	Fonction	Cette fonction locale vérifie si la période VERTE résiduelle est plus courte que celle requise par la trame minimale pour l'acheminement
StartFragStream	Fonction	Cette fonction locale déclenche la création des fragments de FRAG.
LastFragment	Macro	Cette macro locale contient les informations stipulant si le fragment réel est le dernier fragment. Le flux FRAG se termine en même temps que l'acheminement du dernier fragment
Period	Macro	Cette macro locale contient les informations stipulant la période active (Verte ou Rouge).

4.9.6.1.2 Fragmentation statique

4.9.6.1.2.1 Définitions de primitives

4.9.6.1.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par FRAG_S sont décrites dans la définition de service et présentées dans le Tableau 224.

Tableau 224 – Primitives distantes émises ou reçues par FRAG_S

Primitive	Source	Destination	Paramètres associés	Fonctions
QueueHandler.req	FRAG_S	QueueHandler	Queue, Frame	—

4.9.6.1.2.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par FRAG_S sont décrites dans la définition de service et présentées dans le Tableau 225.

Tableau 225 – Primitives locales émises ou reçues par FRAG_S

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.9.6.1.2.2 Schéma de transition d'états

Le schéma de transition d'états du FRAG_S est présenté à la Figure 65.

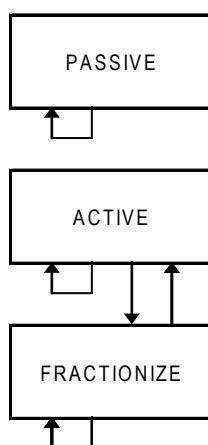


Figure 65 – Schéma de transition d'états de FRAG_S

Etats du FRAG_S

PASSIVE

L'unité FRAG n'est pas utilisée. Transmissions des trames non affectées.

ACTIVE

L'unité FRAG pour le flux défini est active et attend les trames pour les fractionner si nécessaire.

FRACTIONIZE

Une trame est fragmentée et acheminée. Les trames à haute priorité qui ne nécessitent pas de fragmentation peuvent dépasser les fragments. Après l'envoi du dernier fragment, l'état ACTIVE est atteint. Si un temps résidentiel est disponible, la fragmentation commence immédiatement avec la trame suivante.

4.9.6.1.2.3 Description du diagramme d'états

Le FRAG est chargé de la fragmentation des trames selon la période VERTE résiduelle. Il récupère les trames de QueueHandler, les fragmente, si cela est nécessaire et permis par le protocole, et stocke les fragments dans les files d'attente inférieures de QueueHandler.

A l'état inactif, FRAG doit être évité par QueueHandler.

La fragmentation peut être effectuée selon deux stratégies différentes. La première est la fragmentation dynamique, qui utilise la phase VERTE résidentielle alors que la seconde est la fragmentation statique, qui utilise une taille de fragment prédéfinie.

4.9.6.1.2.4 Table d'états FRAG_S (fragmentation statique)

Le Tableau 226 contient la table d'états FRAG_S.

Tableau 226 – Table d'états FRAG_S (statique)

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	PASSIVE	=> ignore	PASSIVE
2	ACTIVE	QueuesNotEmpty () /Period == RED => ignore	ACTIVE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
3	ACTIVE	QueuesNotEmpty () /Period == GREEN && FrameSizeAllowsFragmentation == FALSE => GetFrameFromQueue () // Adapt the assigned queues if more than one Frag stream exists QueueHandler.req () //NOTE Envoi de la trame	ACTIVE
4	ACTIVE	QueuesNotEmpty () /Period == GREEN && FrameSizeAllowsFragmentation == TRUE => GetFrameFromQueue () // Adapt the assigned queues if more than one Frag stream exists StartFragStream ()	FRACTIONIZE
5	FRACTIONIZE	QueuesNotEmpty () => ignore	FRACTIONIZE
6	FRACTIONIZE	StartFragStream () => CalculateFragSize () CreateAllFrag () For all fragment frames QueueHandler.req () //NOTE Envoi du fragment de trame	FRACTIONIZE
7	FRACTIONIZE	LastFragmentTransmitted () =>	ACTIVE
8	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameSizeAllowsFragmentation == FALSE => GetFrameFromQueue () // Adapt the assigned queues if more than one Frag stream exists QueueHandler.req () //NOTE Envoi de la trame	FRACTIONIZE
9	FRACTIONIZE	QueuesNotEmpty () /CheckForFrameWithHigherPriority () == EXISTS && FrameSizeAllowsFragmentation == TRUE => ignore	FRACTIONIZE

4.9.6.1.2.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 227 contient les fonctions, macros, temporisateurs et variables utilisés par le FRAG_S ainsi que leurs arguments et leurs descriptions.

Tableau 227 – Fonctions, Macros, Temporisateurs et Variables utilisés par le FRAG_S (statique)

Nom	Type	Fonction/Signification
CalculateFragSize	Fonction	Cette fonction locale calcule la taille des fragments en fonction de la taille de fragment donnée et des règles de fragmentation. La longueur par défaut est donnée par l'ingénierie en fonction de la période VERTE minimale de toutes les phases.
CheckForFrameWithHigherPriority	Fonction	Cette fonction locale vérifie si une trame avec une priorité plus haute que celle fragmentée est stockée dans les files d'attente.
CreateAllFrag	Fonction	Cette fonction locale crée tous les fragments selon les règles de codage et les règles de fragmentation. Soit premier, soit suivant soit dernier. Garder à l'esprit que les tailles de trame minimale et maximale d'une trame de l'IEEE 802.3 sont toujours valides, même pour les fragments.
FrameSizeAllowsFragmentation	Fonction	Cette fonction locale vérifie si la trame est suffisamment grande pour la fragmentation
GetFrameFromQueue	Fonction	Cette fonction locale extrait la trame suivante selon la quantité de flux FRAG et les priorités.
QueuesNotEmpty	Fonction	Cette fonction locale est exécutée si une entrée est stockée dans les files d'attente.
StartFragStream	Fonction	Cette fonction locale déclenche la création des fragments de FRAG.
LastFragmentTransmitted	Fonction	Cette fonction locale est exécutée si le dernier fragment est acheminé.
Period	Macro	Cette macro locale contient les informations stipulant la période active (Verte ou Rouge).

4.9.6.2 Défragmentation

4.9.6.2.1 Définitions de primitives

4.9.6.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par DEFRAG sont décrites dans la définition de service et présentées dans le Tableau 228.

Tableau 228 – Primitives distantes émises ou reçues par DEFRAG

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_DEFRAG.ind	DEMUX	DEFRAG	Port, Tstamp, DA, SA, VLAN, N_SDU	—
DMUX_MAC_RELAY.ind	DEFRAG	MAC_RELAY	Port, Tstamp, DA, SA, N_SDU	—

4.9.6.2.1.2 Primitives échangées entre les machines locales

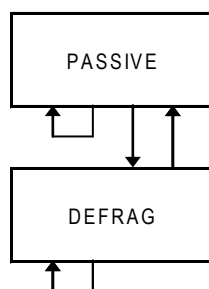
Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DEFRAG sont décrites dans la définition de service et présentées dans le Tableau 229.

Tableau 229 – Primitives locales émises ou reçues par DEFRAG

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.9.6.2.2 Schéma de transition d'états

Le schéma de transition d'états du DEFRAG est présenté à la Figure 66.

**Figure 66 – Schéma de transition d'états de DEFRAG**

Etats du DEFRAG

PASSIVE

En attente du premier fragment d'un flux. Pour chaque flux pris en charge, il existe une unité DEFRAG.

Un flux a été lancé par le premier fragment. Après réception du dernier fragment du flux actuel, la machine atteint de nouveau l'état PASSIVE.

DEFRAG

Si la quantité de données reçue lors de DEFRAG dépasse la taille de trame maximale définie dans l'IEEE 802.3, la défragmentation est arrêtée prématurément et l'état PASSIVE est saisi.

4.9.6.2.3 Description du diagramme d'états

Un diagramme d'états de DEFRAG doit exister pour chaque flux pris en charge. Au moins un flux doit être pris en charge, si la fragmentation est prise en charge. Dans ce cas, le DEFRAG attend un FRAGMENT après que la liaison est active.

4.9.6.2.4 Table d'états DEFRAG

Le Tableau 230 contient la table d'états DEFRAG qui existe pour chaque port.

Tableau 230 – Table d'états DEFRAG

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	PASSIVE	DMUX_DEFRAG.ind () /DefragGuard () == FirstFragment => Store fragment	DEFRAG
2	PASSIVE	DMUX_DEFRAG.ind () /DefragGuard () == Invalid => ignore	PASSIVE
3	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == NextFragment && FrameLimitReached => Delete stored fragments	PASSIVE
4	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == NextFragment && ! FrameLimitReached => Store fragment	DEFRAG
5	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == LastFragment && FrameLimitReached => Delete stored fragments	PASSIVE
6	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == LastFragment && ! FrameLimitReached => Create frame Delete stored fragments DMUX_MAC_RELAY.ind ()	PASSIVE
7	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == FirstFragment => Delete stored fragments Store fragment	DEFRAG
8	DEFRAG	DMUX_DEFRAG.ind () /DefragGuard () == Invalid => Delete stored fragments	PASSIVE

4.9.6.2.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 231 contient les fonctions utilisées par le DEFRAG, ainsi que leurs arguments et leurs descriptions.

Tableau 231 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DEFRAG

Nom	Type	Fonction/Signification
DefragGuard	Fonction	Cette fonction locale vérifie les fragments par rapport aux tables de vérité.
Store fragment	Macro	Cette macro locale stocke le fragment pour une défragmentation ultérieure
Delete stored fragment	Macro	Cette macro locale supprime les fragments stockés précédemment
Create frame	Macro	Cette macro locale crée une trame en dehors des fragments stockés
FrameLimitReached	Macro	Cette macro locale vérifie si les limites de trame définies dans l'IEEE 802.3 sont dépassées.

4.9.6.2.6 DefragGuard

DefragGuard vérifie la trame reçue selon le Tableau 232, le Tableau 233 et le Tableau 234. Le tableau utilisé dépend de la fonction prise en charge en local.

La défragmentation doit accepter une trame indépendamment de la valeur de FrameID.FragSequence conjointement avec FragStatus.FragmentNumber:=0 comme le début d'une nouvelle séquence de défragmentation.

Tableau 232 – Table de vérité pour DefragGuard – premier fragment

Entrée				Sortie
FragmentationFrameID. FragSequence	FragStatus. FragmentNumber	FragStatus. MoreFollows	FragDataLength	Local
Non valide	—	—	—	Non valide
Valide	Non valide	—	—	Non valide
Valide	Valide	FALSE ^a	—	Non valide
Valide	Valide	TRUE	Non valide	Non valide
Valide	Valide	TRUE	Valide	Valide

^a MoreFollows pour le premier fragment ne doit jamais être mis à FALSE par l'émetteur.

Tableau 233 – Table de vérité pour DefragGuard – fragment suivant

Entrée				Sortie
FragmentationFrameID. FragSequence	FragStatus. FragmentNumber	FragStatus. MoreFollows	FragDataLength	Local
Non valide	—	—	—	Non valide
Valide ^a	Non valide	—	—	Non valide
Valide	Valide ^b	FALSE	—	Non valide
Valide	Valide ^b	TRUE	Non valide	Non valide
Valide	Valide	TRUE	Valide	Valide

^a Identique à celui stocké dans le premier fragment.
^b Celui stocké dans le dernier fragment reçu de cette FragSequence (FragmentNumber =+ 1).

Tableau 234 – Table de vérité pour DefragGuard – dernier fragment

Entrée				Sortie
FragmentationFrameID. FragSequence	FragStatus. FragmentNumber	FragStatus. MoreFollows	FragDataLength	Local
Non valide	—	—	—	Non valide
Valide	Non valide	—	—	Non valide
Valide	Valide	TRUE	—	Non valide
Valide	Valide	FALSE ^a	Non valide	Non valide
Valide	Valide	FALSE ^a	Valide	Valide

^a MoreFollows pour le dernier fragment doit être mis à FALSE par l'émetteur.

4.10 Appel de procédure à distance

4.10.1 Description de la syntaxe RPC

4.10.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.10.1.2 Syntaxe abstraite RPC APDU

Le Tableau 235 présente l'utilisation de la Publication C706 de The Open Group et définit la syntaxe abstraite des PDU de couche application appelés APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 235 doivent représenter le contenu de la DLSDU dans le Tableau 5.

Tableau 235 – syntaxe RPC APDU

Nom de l'APDU	Structure de l'APDU
CL-RPC-PDU	IPHeader, UDPHeader, RPCHeader, NDRData
NDRData	NDRDataRequest ^ NDRDataResponse ^ NDREPMAPLookupReq ^ NDREPMAPLookupRes ^ NDREPMAPLookupFreeReq ^ NDREPMAPLookupFreeRes ^ NDRAck ^ NDRQuAck ^ NDRQuit ^ NDRFack ^ NDRFault ^ NDRNoCall ^ NDRWorking ^ NDRPing ^ NDRReject

Le Tableau 236 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 235.

Tableau 236 – Substitutions RPC

Nom de substitution	Structure
RPCHeader	RPCVersion (4), RPCPacketType, RPCFlags, RPCFlags2, RPCDRep, RPCSerialHigh, RPCObjectUUID ^a , RPCInterfaceUUID ^b , RPCActivityUUID, RPCServerBootTime, RPCInterfaceVersion, RPCSequenceNmb, RPCOperationNmb ^c , RPCInterfaceHint, RPCActivityHint, RPCLengthOfBody, RPCFragmentNmb, RPCAuthenticationProtocol, RPCSerialLow
NDRDataRequest	ArgsMaximum, ArgsLength, MaximumCount, Offset(0), ActualCount, PROFINETIOServiceReqPDU
NDRDataResponse	PNIOStatus, ArgsLength, MaximumCount, Offset(0), ActualCount, [PROFINETIOServiceResPDU]
NDRFault	RPCNCAFaultStatus
NDRAck	NULL
NDRQuAck	NULL ^ (RPCCancelVersion(0), RPCCancelID, RPCServerIsAccepting)
NDRQuit	NULL ^ (RPCCancelVersion(0), RPCCancelID)
NDRFack	NULL ^ (RPCVersionFack(0), RPCPad1, RPCWindowSize, RPCMaxTsdu, RPCMaxFragSize, RPCSerialNumber, RPCSelAckLen, RPCArrayOfSelAck*)
NDRNoCall	NULL ^ NDRFack
NDRWorking	NULL
NDRPing	NULL
NDRReject	RPCNCARjectStatus
NDREPMAPLookupReq	RPCInquiryType, RPCObjectReference(1), RPCObjectUUID, RPCInterfaceReference(2), RPCInterfaceUUID, RPCInterfaceVersionMajor, RPCInterfaceVersionMinor, RPCVersionOption(1), RPCEntryHandleAttribute(0), RPCEntryHandleUUID, RPCMaxEntries

Nom de substitution	Structure
NDREMapLookupRes	RPCEntryHandleAttribute(0), RPCEntryHandleUUID, RPCNumberOfEntries, RPCMaxEntries, RPCEntriesOffset, RPCEntriesCount, [(RPCObjectUUID, RPCTowerReference, RPCAnnotationOffset, RPCAnnotationLength, RPCAnnotation, [RPCGap*], RPCTowerLength, RPCTowerOctetStringLength, [RPCTowerOctetString*], [RPCGap*])*], RPCEPMapStatus
RPCTowerOctetString	RPCFloorCount, [RPCFloor*]
RPCFloor	RPCLHSByteCount, [RPCProtocolIdentifierString*], RPCRHSByteCount, [RPCRelatedData*]
RPCProtocolIdentifier-String	(RPCID, RPCInterfaceUUID ^a , RPCInterfaceVersionMajor ^a) ^ (RPCID, RPCDataRepresentationUUID ^a , RPCInterfaceVersionMajor) ^ RPCProtocolIdentifier ^ RPCServerUDPPort ^ RPCHostAddress ^a Doit être codé au format petit-boutiste
RPCRelatedData	RPCInterfaceVersionMinor ^a ^ RPCPortNumber ^ RPCIPAddress ^a Doit être codé au format petit-boutiste
NDREMapLookupFree Req	RPCEntryHandleAttribute, RPCEntryHandleUUID
NDREMapLookupFree Res	RPCEntryHandleAttribute, RPCEntryHandleUUID, RPCEPMapStatus
RPCAnnotation	DeviceType, Blank, OrderID, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision, EndTerm
^a Pour identifier l'appareil E/S, l'appareil de commande E/S, le superviseur E/S. ^b Pour identifier le type d'interface PNIO. ^c Pour identifier le type de service (Connect, Release, Read, Write, Control, par exemple).	

4.10.2 Syntaxe de transfert RPC

4.10.2.1 Section de codage relative à CL-RPC-PDU

4.10.2.1.1 Codage du champ RPCVersion

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 237.

Tableau 237 – RPCVersion

Valeur (hexadécimale)	Signification
0x04	Version de RPC utilisée
Autre	Réservé

4.10.2.1.2 Codage du champ RPCPacketType

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 238.

Tableau 238 – RPCPacketType

Valeur (hexadécimale)	Signification
0x00	Demande
0x01	Ping
0x02	Réponse
0x03	Défaut

Valeur (hexadécimale)	Signification
0x04	En fonctionnement
0x05	Pas d'appel, réponse au ping
0x06	Rejet
0x07	Acquittement
0x08	Annulation sans connexion
0x09	Acquittement de fragment (FACK-PDU)
0x0A	Annuler l'acquittement
0x0B – 0xFF	Réservé

Le décodeur doit uniquement vérifier les 5 bits de poids faible.

4.10.2.1.3 Codage du champ RPCFlags

Ce champ doit être conforme à 3.4.2.3 avec les valeurs conformes au Tableau 239.

Tableau 239 – RPCFlags

Bit	Signification si la valeur 1 est attribuée
Bit 0	Spécifique à la mise en œuvre, doit être mis à zéro
Bit 1	Dernier fragment
Bit 2	Fragment
Bit 3	Aucun acquittement de fragment demandé
Bit 4	Maybe (Eventuellement)
Bit 5	Idempotent
Bit 6	Broadcast (Diffusion)
Bit 7	Spécifique à la mise en œuvre, doit être mis à zéro

4.10.2.1.4 Codage du champ RPCFlags2

Ce champ doit être conforme à 3.4.2.3 avec les valeurs conformes au Tableau 240.

Tableau 240 – RPCFlags2

Bit	Signification si la valeur 1 est attribuée
Bit 0	Spécifique à la mise en œuvre, doit être mis à 0
Bit 1	Annulation en attente à la fin de l'appel
Bit 2	Réservé
Bit 3	Réservé
Bit 4	Réservé
Bit 5	Réservé
Bit 6	Réservé
Bit 7	Réservé

4.10.2.1.5 Codage du champ RPCDRep

Ce champ doit être codé en type de données OctetString[3].

RPCDRep Octet 1

Le codage du premier octet doit être conforme à 3.4.2.3 et la signification des bits individuels conforme au Tableau 241.

Tableau 241 – RPCDRep.Character- et IntegerEncoding

Bit	Nom du champ	Valeur	Signification
0 - 3	RPCDRep.CharacterEncoding ^a	0	ASCII
		1	EBCDIC
4 - 7	RPCDRep.IntegerEncoding ^b	0	Gros-boutiste ^c
		1	Petit-boutiste ^c

^a Non utilisé avec le Type 10.

^b A l'exception de 3.4.2.5, les règles de codage RPC s'appliquent à RPCHeader, aux substitutions NDR et aux substitutions NDREMap du Type 10. Seul Unsigned8, Unsigned16 ou Unsigned32 est utilisé ici. Les données utilisateur des définitions de service de Type 10 ne sont pas affectées, car tous les éléments correspondent à un OctetString opaque.

^c Voir The Open Group — Publication C706.

RPCDRep Octet 2

Les valeurs du deuxième octet doivent être codées conformément au Tableau 242.

Tableau 242 – RPCDRep Octet 2 – Représentation à virgule flottante

Valeur (hexadécimale)	Signification
0x00	IEEE
0x01	VAX
0x02	CRAY
0x03	IBM
0x04 – 0xFF	Réservé

RPCDRep Octet 3

La valeur du troisième octet doit être nulle.

4.10.2.1.6 Codage du champ RPCSerialHigh

Ce champ doit être codé en type de données Unsigned8. La valeur contient l'octet supérieur du numéro de fragment de l'appel.

4.10.2.1.7 Codage du champ RPCSerialLow

Ce champ doit être codé en type de données Unsigned8. La valeur contient l'octet inférieur du numéro de fragment de l'appel.

NOTE La valeur du champ RPCSerial est incrémentée à chaque transmission, même avec une répétition de transmission, pour se distinguer du champ FragmentNmb.

4.10.2.1.8 Codage du champ RPCObjectUUID

Ce champ doit être codé sous la forme d'une structure de type de données contenant les éléments suivants:

- Data1 en Unsigned32
- Data2 en Unsigned16
- Data3 en Unsigned16
- Data4 sous la forme d'une matrice de huit Unsigned8 (Octet 1 à Octet 8)

L'ordre des octets doit être conforme à la valeur du champ RPCDRep (petit-boutiste ou gros-boutiste).

L'ordre des octets de Data4 est présenté au Tableau 243.

Tableau 243 – RPCObjectUUID.Data4

Octet	Signification
1	Valeurs voir Tableau 244
2	
3	Instance High
4	Instance Low
5	DeviceID High
6	DeviceID Low
7	VendorID High
8	VendorID Low

Les valeurs définies pour les éléments PNIO sont présentées dans le Tableau 244 et au Tableau 245. Elles doivent être utilisées conjointement avec les valeurs du Tableau 246.

Tableau 244 – RPCObjectUUID pour PNIO

Valeur	Description
UUID_IO_ObjectInstance_XYZ DEA00000-6C97-11D1-8271-{xxxxyyyyzzzz}	Identifie une instance d'objet particulière d'un appareil physique en présence de plusieurs instances, où
xxxx	Représente le numéro d'instance ou de nœud Voir Tableau 245
YYYY	Identifie l'ID d'appareil sous la forme d'un numéro spécifique au fournisseur pour la classe d'appareil
zzzz	Représente l'ID fournisseur sous la forme d'un numéro administratif central attribué par l'entité cliente responsable

Tableau 245 – RPCObjectUUID pour PNIO avec plusieurs interfaces

Valeur	Description
xxxx	Identifie une instance d'objet particulière d'un appareil physique en présence de plusieurs instances, où
Bit 0 – Bit 11	Représente le numéro d'instance ou de nœud
Bit 12 – Bit 15	Identifie l'interface 0x0: Interface 0 (Par défaut) ... 0xF: Interface 15 Voir "I" dans la Formule (45)

4.10.2.1.9 Codage du champ RPCInterfaceUUID

Ce champ doit être codé sous la forme d'une structure de type de données contenant les éléments suivants:

- Data1 en Unsigned32
- Data2 en Unsigned16
- Data3 en Unsigned16
- Data4 sous la forme d'une matrice de huit Unsigned8

L'ordre des octets doit être conforme à la valeur du champ RPCDRep (petit-boutiste ou gros-boutiste).

Les valeurs définies pour les éléments PNIO sont présentées dans le Tableau 246.

Tableau 246 – RPCInterfaceUUID for PNIO

Valeur	Description
UUID_IO_DeviceInterface DEA00001-6C97-11D1-8271-00A02442DF7D	Identifie l'interface d'un appareil E/S de manière unique. RPCInterfaceVersion doit être: Version majeure: 1 Version mineure: 0
UUID_IO_ControllerInterface DEA00002-6C97-11D1-8271-00A02442DF7D	Identifie l'interface d'un appareil de commande E/S de manière unique. RPCInterfaceVersion doit être: Version majeure: 1 Version mineure: 0
UUID_IO_SupervisorInterface DEA00003-6C97-11D1-8271-00A02442DF7D	Identifie l'interface d'un superviseur E/S de manière unique. RPCInterfaceVersion doit être: Version majeure: 1 Version mineure: 0
UUID_IO_ParameterServerInterface DEA00004-6C97-11D1-8271-00A02442DF7D	Identifie l'interface d'un serveur de paramètre E/S de manière unique. RPCInterfaceVersion doit être: Version majeure: 1 Version mineure: 0

Les valeurs définies pour le mappeur de point d'extrémité RPC sont présentées dans le Tableau 247.

Tableau 247 – RPCInterfaceUUID pour le mappeur de point d'extrémité RPC

Valeur	Description
UUID_EPMap_Interface E1AF8308-5D1F-11C9-91A4-08002B14A0FA UUID_EPMap_Object 00000000-0000-0000-0000-000000000000	Identifie l'interface du mappeur de point d'extrémité. RPCInterfaceVersion doit être: Version majeure: 3 Version mineure: 0

4.10.2.1.10 Codage du champ RPCActivityUUID

Ce champ doit être codé sous la forme d'une structure de type de données contenant les éléments suivants:

- Data1 en Unsigned32
- Data2 en Unsigned16

- Data3 en Unsigned16
- Data4 sous la forme d'une matrice de huit Unsigned8

NOTE La réponse reflète le contenu du champ de la demande.

Le caractère boutiste est conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

L'IOC et l'IOD génèrent un RPCActivityUUID pour chaque AR et les utilisent tant que l'AR existe.

4.10.2.1.11 Codage du champ RPCServerBootTime

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

NOTE Il n'est pas utile que les fonctions Idempotent conservent cette valeur. Seul le mappeur de point d'extrémité peut répondre par une valeur nulle.

4.10.2.1.12 Codage du champ RPCInterfaceVersion

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0 – 15: RPCInterfaceVersion.Major

Ce champ doit être mis conformément au Tableau 248.

Tableau 248 – RPCInterfaceVersion.Major

Valeur (hexadécimale)	Signification
0x0000	Réserve
0x0001 – 0xFFFF	Information de version valide

Bit 16 – 31: RPCInterfaceVersion.Minor

Ce champ doit être mis conformément au Tableau 249.

Tableau 249 – RPCInterfaceVersion.Minor

Valeur (hexadécimale)	Signification
0x0000 – 0xFFFF	Information de version valide

Le caractère boutiste est conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.13 Codage du champ RPCSequenceNmb

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.14 Codage du champ RPCOperationNmb

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

RPCOperationNmb identifie le service PNIO pris en charge par les interfaces PNIO

- appareil E/S,

- appareil de commande E/S et
- superviseur E/S.

Les valeurs des services PNIO sont définies au Tableau 250.

Tableau 250 – RPCOperationNmb (appareil E/S, appareil de commande E/S et superviseur E/S)

Valeur (décimale)	Service	Utilisation
0	Connexion	—
1	Release	—
2	Lecture	Valide uniquement avec ARUUIID<>0
3	Ecriture	Valide uniquement avec ARUUIID<>0
4	Control	—
5	Read Implicit	Valide uniquement avec ARUUIID=0
6 – 65 535	Réservé	—

Les valeurs des services du mappeur de point d'extrémité conformes au Tableau 251 doivent être utilisées.

Tableau 251 – RPCOperationNmb pour le mappeur de point d'extrémité

Valeur (décimale)	Utilisation	Service
0	Facultatif	Insérer
1	Facultatif	Delete
2	Obligatoire	Lookup
3	Facultatif	Map
4	Obligatoire	LookupHandleFree
5	Facultatif	InqObject
6	Facultatif	MgmtDelete
7 – 65 535	Réservé	—

4.10.2.1.15 Codage du champ RPCInterfaceHint

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il convient d'attribuer la valeur 0xFFFF (pas d'indication) pour cette version.

Le client doit commencer par "no hint" (pas d'indication) pour le premier appel et il est recommandé qu'il utilise la réponse du serveur pour les appels suivants afin de permettre l'optimisation du serveur.

4.10.2.1.16 Codage du champ RPCActivityHint

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il convient d'attribuer la valeur 0xFFFF (pas d'indication) pour cette version.

Le client doit commencer par "no hint" (pas d'indication) pour le premier appel et il est recommandé qu'il utilise la réponse du serveur pour les appels suivants afin de permettre l'optimisation du serveur.

4.10.2.1.17 Codage du champ RPCLengthOfBody

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

La valeur doit être définie sur le nombre d'octets de NDRData de la trame en cours.

L'acheminement de NDRData peut impliquer l'utilisation de plusieurs trames. Dans ce cas, RPCLengthOfBody contient uniquement le nombre d'octets pour la trame en cours.

4.10.2.1.18 Codage du champ RPCFragmentNmb

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

La valeur doit être définie sur le nombre de fragments en cours.

4.10.2.1.19 Codage du champ RPCAuthenticationProtocol

Ce champ doit être codé en type de données Unsigned8.

La valeur doit être nulle pour indiquer l'absence d'authentification.

4.10.2.1.20 Codage du champ RPCCancelVersion

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

La valeur doit être mis à zéro.

4.10.2.1.21 Codage du champ RPCCancelID

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.22 Codage du champ RPCServerIsAccepting

Ce champ doit être codé en type de données Unsigned8.

4.10.2.1.23 Codage du champ RPCVersionFack

Ce champ doit être codé en type de données Unsigned8.

La valeur doit être mis à zéro.

4.10.2.1.24 Codage du champ RPCPad1

Ce champ doit être codé en type de données Unsigned8.

4.10.2.1.25 Codage du champ RPCWindowSize

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep (petit-boutiste ou gros-boutiste).

4.10.2.1.26 Codage du champ RPCMaxTsdu

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.27 Codage du champ RPCMaxFragSize

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.28 Codage du champ RPCSerialNumber

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.29 Codage du champ RPCSelAckLen

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.30 Codage du champ RPCArrayOfSelAck

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.1.31 Codage du champ RPCDataRepresentationUUID

Ce champ doit être codé sous la forme d'une structure de type de données contenant les éléments suivants:

- Data1 en Unsigned32
- Data2 en Unsigned16
- Data3 en Unsigned16
- Data4 sous la forme d'une matrice de huit Unsigned8

L'ordre des octets doit être petit-boutiste avec une valeur conforme au Tableau 252.

Tableau 252 – RPCDataRepresentationUUID – valeurs définies

Value [UUID]	Valeur	Description
8A885D04-1CEB-11C9-9FE8-08002B104860	Représentation des données	Identifie la représentation de données RPC de manière unique.

4.10.2.2 Section de codage relative à NDREPMAPDU**4.10.2.2.1 Codage du champ DeviceType**

Ce champ doit être codé en type de données VisibleString[25]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de blancs si sa longueur est inférieure à 25 octets.

NOTE Voir également le champ DeviceVendorValue.

4.10.2.2.2 Codage du champ OrderID

Ce champ doit être codé en type de données VisibleString[20]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de blancs si sa longueur est inférieure à 20 octets.

4.10.2.2.3 Codage du champ HWRevision

Ce champ doit être codé en type de données VisibleString[5]. La valeur doit être mise spécifique à un fabricant à l'aide de caractères "0" à "9" et "<Blank>" (espace). La plage est comprise entre <Blank><Blank><Blank><Blank>0 et 99999.

NOTE Par exemple, HWRevision peut être défini comme suit:<Blank><Blank><Blank><Blank>2, <Blank>5714 ou 61214.

Ce champ doit présenter la même révision que le champ IM_Hardware_Revision.

4.10.2.2.4 Codage du champ SWRevisionPrefix

Ce champ doit être codé en type de données VisibleString[1]. La valeur doit être définie en fonction du fabricant à l'aide des caractères suivants:

- V pour une version éditée officiellement
- R pour Revision (Révision)
- P pour Prototype
- U pour Under Test (Essai pratique)
- T pour Test Device (Appareil d'essai)

Il est recommandé d'utiliser "V" conjointement avec le champ SWRevision contenant "<Blank><Blank>0<Blank><Blank>0<Blank><Blank>0" si un sous-module ne contient pas de logiciel / microprogramme.

4.10.2.2.5 Codage du champ SWRevision

Ce champ doit être codé en type de données VisibleString[9]. La valeur doit être mise spécifique à un fabricant à l'aide de caractères "0" à "9" et "<Blank>" (espace). La plage est comprise entre <Blank><Blank>0<Blank><Blank>0<Blank><Blank>0 et 999999999.

Par exemple, SWRevision pourrait être défini comme suit: <Blank><Blank>1<Blank><Blank>0<Blank><Blank>0. La couche de présentation peut insérer un "." après chaque groupe de trois octets pour une meilleure visualisation.

4.10.2.2.6 Codage du champ Blank

Ce champ doit être codé en type de données OctetString avec 1 octet.

Il doit être mis à 0x20.

4.10.2.2.7 Codage du champ RPCInquiryType

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Le codage doit être conforme au Tableau 253.

Tableau 253 – RPCInquiryType

Valeur (hexadécimale)	Signification	Utilisation
0x00000000	Lire toutes les interfaces enregistrées avec les objets.	Obligatoire
0x00000001	Lire tous les objets d'une interface enregistrée.	Facultatif
0x00000002	Lire toutes les interfaces contenant un objet dédié.	Facultatif
0x00000003	Lire une interface dédiée avec un objet dédié.	Facultatif
0x00000004 – 0xFFFFFFFF	Réservé	—

4.10.2.2.8 Codage du champ RPCObjectReference

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à 1.

4.10.2.2.9 Codage du champ RPCInterfaceReference

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep (petit-boutiste ou gros-boutiste).

Il doit être mis à 2.

4.10.2.2.10 Codage du champ RPCInterfaceVersionMajor

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.11 Codage du champ RPCInterfaceVersionMinor

Ce champ doit être codé en type de données Unsigned16. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.12 Codage du champ RPCVersionOption

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à 1.

4.10.2.2.13 Codage du champ RPCEntryHandleAttribute

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à 0.

4.10.2.2.14 Codage du champ RPCEntryHandleUUID

Ce champ doit être codé en type de données UUID. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.15 Codage du champ RPCMaxEntries

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être au moins à 1.

4.10.2.2.16 Codage du champ RPCNumberOfEntries

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.17 Codage du champ RPCEntriesOffset

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à 0.

4.10.2.2.18 Codage du champ RPCEntriesCount

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.19 Codage du champ RPCTowerReference

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à 3.

4.10.2.2.20 Codage du champ RPCAnnotationOffset

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à 0.

4.10.2.2.21 Codage du champ RPCAnnotationLength

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Il doit être mis à une valeur comprise entre 0 et 64.

4.10.2.2.22 Codage du champ EndTerm

Ce champ doit être codé en type de données Unsigned8. Il doit être mis à 0.

4.10.2.2.23 Codage du champ RPCGap

Ce champ doit être codé en type de données Unsigned8.

4.10.2.2.24 Codage du champ RPCTowerLength

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.25 Codage du champ RPCTowerOctetStringLength

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

4.10.2.2.26 Codage du champ RPCEMapStatus

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Le codage doit être conforme au Tableau 254.

Tableau 254 – RPCEMapStatus

Valeur (hexadécimale)	Signification
0x00000000	RPC correct
0x16C9A0D6	Point d'extrémité non enregistré
OTHERS	Réservé

4.10.2.2.27 Codage du champ RPCFloorCount

Ce champ doit être codé en type de données Unsigned16. Le codage doit être indépendant du champ RPCDRep et toujours petit-boutiste.

Il doit être mis à 5.

4.10.2.2.28 Codage du champ RPCLHSByteCount

Ce champ doit être codé en type de données Unsigned16. Le codage doit être indépendant du champ RPCDRep et toujours petit-boutiste.

4.10.2.2.29 Codage du champ RPCRHSByteCount

Ce champ doit être codé en type de données Unsigned16. Le codage doit être indépendant du champ RPCDRep et toujours petit-boutiste.

4.10.2.2.30 Codage du champ RPCID

Ce champ doit être codé en type de données Unsigned8.

Il doit être mis à 0x0D.

4.10.2.2.31 Codage du champ RPCProtocolIdentifier

Ce champ doit être codé en type de données Unsigned8.

Il doit être mis à 0x0A.

4.10.2.2.32 Codage du champ RPCServerUDPPort

Ce champ doit être codé en type de données Unsigned8.

Il doit être mis à 0x08.

4.10.2.2.33 Codage du champ RPCHostAddress

Ce champ doit être codé en type de données Unsigned8.

Il doit être mis à 0x09.

4.10.2.2.34 Codage du champ RPCPortNumber

Ce champ doit être codé en type de données Unsigned16. Le codage doit être indépendant du champ RPCDRep et toujours gros-boutiste.

4.10.2.2.35 Codage du champ RPCIPAddress

Ce champ doit être codé en type de données Unsigned32. Le codage doit être indépendant du champ RPCDRep et toujours gros-boutiste.

L'élément peut être zéro.

4.10.2.3 Section de codage relative à NDRData

4.10.2.3.1 Codage du champ ArgsMaximum

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Sa valeur doit être mise à la taille maximale de la mémoire tampon disponible pour la réponse.

4.10.2.3.2 Codage du champ ArgsLength

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Sa valeur doit être mise au nombre d'octets de PROFINETIOServiceReqPDU ou de PROFINETIOServiceResPDU.

4.10.2.3.3 Codage du champ MaximumCount

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

En cas de demande, la valeur doit être identique à celle du champ ArgsMaximum. Dans la réponse, la valeur doit être issue du champ ArgsMaximum de la demande correspondante.

En cas d'incohérence dans le codage de la "Matrice Variable Conforme Unidimensionnelle" il convient que le paquet soit rejeté du côté serveur et qu'une erreur soit rapportée localement du côté client, comme stipulé dans The Open Group — Publication C706.

4.10.2.3.4 Codage du champ Offset

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

La valeur doit être mis à zéro.

4.10.2.3.5 Codage du champ ActualCount

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

La valeur doit être identique à celle du champ ArgsLength.

4.10.2.3.6 Codage du champ PNIOStatus

Ce champ doit être codé en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste) dans le premier champ de NDRDataResponse. Dans tous les autres cas, l'ordre des octets doit être gros-boutiste.

Le contenu est défini en 5.2.6. PNIOStatus doit être calculé à l'aide de la formule suivante

$$\begin{aligned} \text{PNIOStatus} = & \text{ErrorCode} \times 16\,777\,216 + \\ & \text{ErrorDecode} \times 65\,536 + \\ & \text{ErrorCode1} \times 256 + \\ & \text{ErrorCode2} \end{aligned} \quad (44)$$

où

ErrorCode	est la valeur du code de l'erreur
ErrorDecode	est la valeur du décodage de l'erreur
ErrorCode1	est la valeur du code1 de l'erreur
ErrorCode2	est la valeur du code2 de l'erreur

4.10.2.4 Section de codage relative à RPC (codes NCA)

4.10.2.4.1 Codage du champ RPCNCAFaultStatus

Les codes d'erreur NCA spécifiques à RPC doivent être codés en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Les valeurs doivent être conformes au Tableau 255.

Tableau 255 – Valeurs de NCAFaultStatus

Valeur (hexadécimale)	Définition
0x1C000001	NCA_s_fault_int_div_by_zero
0x1C000002	NCA_s_fault_addr_error
0x1C000003	NCA_s_fault_fp_div_zero
0x1C000004	NCA_s_fault_fp_underflow
0x1C000005	NCA_s_fault_fp_overflow
0x1C000006	NCA_s_fault_invalid_tag
0x1C000007	NCA_s_fault_invalid_bound
0x1C000008	NCA_s_rpc_version_mismatch
0x1C000009	NCA_s_unspec_reject
0x1C00000A	NCA_s_bad_actid
0x1C00000B	NCA_s_who_are_you_failed
0x1C00000C	NCA_s_manager_not_entered
0x1C00000D	NCA_s_fault_chancel
0x1C00000E	NCA_s_fault_ill_inst
0x1C00000F	NCA_s_fault_fp_error
0x1C000010	NCA_s_fault_int_overflow

Valeur (hexadécimale)	Définition
0x1C000012	NCA_s_fault_unspec
0x1C000013	NCA_s_fault_remote_comm_failure
0x1C000014	NCA_s_fault_pipe_empty
0x1C000015	NCA_s_fault_pipe_closed
0x1C000016	NCA_s_fault_pipe_order
0x1C000017	NCA_s_fault_pipe_discipline
0x1C000018	NCA_s_fault_pipe_comm_error
0x1C000019	NCA_s_fault_pipe_memory
0x1C00001A	NCA_s_fault_context_mismatch
0x1C00001B	NCA_s_fault_remote_no_memory
0x1C00001C	NCA_s_invalid_pres_context_id
0x1C00001D	NCA_s_unsupported_authn_level
0x1C00001F	NCA_s_invalid_checksum
0x1C000020	NCA_s_invalid_crc
0x1C000021	NCA_s_fault_user_defined
0x1C000022	NCA_s_fault_tx_open_failed
0x1C000023	NCA_s_fault_codeset_conv_error
0x1C010001	NCA_s_comm_failure
0x1C010015	NCA_s_fault_string_too_long

4.10.2.4.2 Codage du champ **RPCNCARejectStatus**

Les codes d'erreur NCA spécifiques à RPC doivent être codés en type de données Unsigned32. L'ordre des octets doit être conforme à la valeur du champ RPCDRep.IntegerEncoding (petit-boutiste ou gros-boutiste).

Les valeurs doivent être conformes au Tableau 256.

Tableau 256 – Valeurs de NCARejectStatus

Valeur (hexadécimale)	Définition
0x1C000008	NCA_rpc_version_mismatch
0x1C000009	NCA_unspec_reject
0x1C00000A	NCA_s_bad_actid
0x1C00000B	NCA_who_are_you_failed
0x1C00000C	NCA_manager_not_entered
0x1C010002	NCA_op_rng_error
0x1C010003	NCA_unk_if
0x1C010006	NCA_wrong_boot_time
0x1C010009	NCA_s_you_crashed
0x1C01000B	NCA_proto_error
0x1C010013	NCA_out_args_too_big
0x1C010014	NCA_server_too_busy
0x1C010017	NCA_unsupported_type

0x1C00001D	NCA_unsupported_authn_level
0x1C00001F	NCA_invalid_checksum
0x1C000020	NCA_invalid_crc

4.10.3 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.10.4 Machines de protocole de relation entre applications

4.10.4.1 Machine de protocole RPC

4.10.4.1.1 Définitions de primitives

4.10.4.1.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par RPC sont décrites dans la définition de service et présentées dans le Tableau 257.

Tableau 257 – Primitives distantes émises ou reçues par RPC

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.10.4.1.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par RPC sont décrites dans la définition de service et présentées dans le Tableau 258.

Tableau 258 – Primitives locales émises ou reçues par RPC

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.10.4.1.2 Schéma de transition d'états

Le schéma de transition d'états du RPC est défini dans la norme Technique DCE1.1.

4.10.4.1.3 Description du diagramme d'états

La description du diagramme d'états est définie dans la norme Technique DCE1.1.

4.10.4.1.4 Table d'états RPC

La table d'états est définie dans la norme Technique DCE1.1.

4.10.4.1.5 Fonctions, Macros, Temporisateurs et Variables

Les fonctions sont définies dans la norme Technique DCE1.1.

4.10.4.2 Surveillance des services

Le service Ping initié par un demandeur de service est appelé pour surveiller la santé du répondeur. Le service Cancel peut être répété trois fois au maximum.

Timeout Ping

La valeur doit être 2 s.

NOTE 1 Le service ping est utilisé si la demande a été intégralement envoyée et si la réponse est toujours en attente.

Timeout FRAG

La valeur doit être 2 s.

NOTE 2 La transmission d'un fragment est répétée trois fois au maximum toutes les 2 s en l'absence d'acquittement.

Timeout Cancel

La valeur doit être 1 s.

NOTE 3 Le service Cancel est répété trois fois au maximum toutes les 1 s en l'absence de réaction.

Timeout Resend, Timeout Ack, Timeout Broadcast, Timeout IDLE, Timeout WAIT

Peu importe la valeur.

NOTE 4 Les paramètres décrits ci-dessus sont définis avec leurs valeurs dans DCE RRC.

4.10.5 Machines de Protocole de Mapping DLL

Il n'existe aucune Machine de Protocole de Mapping DLL (DMPM) définie pour ce Protocole.

4.11 Reconnaissance de couche de liaison

4.11.1 Description de la syntaxe commune FAL

4.11.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.11.1.2 Syntaxe abstraite LLDP APDU

Le Tableau 259 définit la syntaxe abstraite des LLDP PDU appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 259 doivent représenter le contenu de la DLSDU dans le Tableau 5.

Tableau 259 – Syntaxe LLDP APDU

Nom de l'APDU	Structure de l'APDU
LLDP-PDU	LLDPChassis, LLDPPort, LLDP TTL, LLDP-PNIO-PDU, LLDPEnd
LLDP-PNIO-PDU	{ LLDP_PNIO_DELAY ^a , LLDP_PNIO_PORTSTATUS, [LLDP_PNIO_ALIAS], LLDP_PNIO_MRPPORTSTATUS ^b , LLDP_PNIO_CHASSIS_MAC, LLDP8023MACPHY ^d , LLDPManagement, LLDP_PNIO_PTCPSTATUS ^c , [LLDPOption*] ^e , [LLDP8021*], [LLDP8023*] }
^a Doit uniquement exister si la mesure LineDelay est prise en charge. ^b Doit uniquement exister si MRP est activé pour ce port. ^c Doit uniquement exister si PTCIP est activé au moyen de PDSyncData Record. ^d Doit uniquement exister si IEEE 802.3 est utilisé. ^e D'autres options LLDP peuvent être utilisées simultanément.	

Le Tableau 260 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 259.

Tableau 260 – Substitutions LLDP

Nom de substitution	Structure
LLDPChassis with MultipleInterfaceMode.- NameOfDevice:=0	LLDPChassisStationName ^ LLDPChassisMacAddress ^a
LLDPChassis with MultipleInterfaceMode.- NameOfDevice:=1	LLDPChassisStationName ^ LLDPChassisMacAddress ^c
LLDPChassisStationName	LLDP_TLVHeader ^b , LLDP_ChassisIDSubType(7) ^b , LLDP_ChassisID
LLDPChassisMacAddress	LLDP_TLVHeader ^b , LLDP_ChassisIDSubType(4) ^b , (CMResponderMacAdd ^ CMInitiatorMacAdd) ^d
LLDPPort	LLDP_TLVHeader ^b , LLDP_PortIDSubType(7) ^b , LLDP_PortID
LLDPTTL	LLDP_TLVHeader ^b , LLDP_TimeToLive(20) ^b
LLDP_PNIO_Header	LLDP_TLVHeader ^b , LLDP_OUI(00-0E-CF)
LLDP_PNIO_DELAY	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x01), PTCP_PortRxDelayLocal, PTCP_PortRxDelayRemote, PTCP_PortTxDelayLocal, PTCP_PortTxDelayRemote, CableDelayLocal
LLDP_PNIO_PORTSTATUS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x02), RTClass2_PortStatus, RTClass3_PortStatus
LLDP_PNIO_ALIAS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x03), AliasNameValue
LLDP_PNIO_MRPPORTSTATUS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x04), MRP_DomainUUID, MRRT_PortStatus
LLDP_PNIO_CHASSIS_MAC	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x05), (CMResponderMacAdd ^ CMInitiatorMacAdd) ^d
LLDP_PNIO_PTCPSTATUS	LLDP_PNIO_Header, LLDP_PNIO_SubType(0x06), PTCP_MasterSourceAddress ^e , PTCP_SubdomainUUID ^f , IRDataUUID ^g , LLDP_LengthOfPeriod ^g , LLDP_RedOrangePeriodBegin ^g , LLDP_OrangePeriodBegin ^g , LLDP_GreenPeriodBegin ^g
LLDPEnd	LLDP_TLVHeader ^b (0)
LLDP8021	LLDP_TLVHeader ^b , LLDP_OUI(00-80-C2) ^h , LLDP_8021_SubType ^h , Data ^h
LLDP8023	LLDP_TLVHeader ^b , LLDP_OUI(00-12-0F) ⁱ , LLDP_8023_SubType ⁱ , Data ⁱ
LLDP8023MACPHY	LLDP_TLVHeader ^b , LLDP_OUI(00-12-0F) ⁱ , LLDP_8023_SubType(1) ⁱ , LLDP_8023_AUTONEG ⁱ , LLDP_8023_PMDCAP ⁱ , LLDP_8023_OPMAU ⁱ
LLDPManagement	LLDP_TLVHeader ^b , LLDP_ManagementData ^j
^a LLDPChassisMacAddress doit être utilisé si aucun NameOfStation n'est attribué. ^b Le codage des champs doit être conforme à l'IEEE 802.1AB-2005. ^c LLDPChassisMacAddress doit être utilisé uniquement si NameOfDevice est égal à NameOfStation et qu'aucun NameOfStation n'est attribué. ^d Doit être l'adresse MAC de l'interface du nœud de transmission ^e Doit être nul si inconnu. ^f Doit être PTCP_SubdomainUUID de PTCP_SyncID:= 0. Sinon, la valeur doit être nulle. ^g Doit être nul si inconnu. ^h Doit être conforme à l'annexe F de l'IEEE 802.1AB-2005. ⁱ Doit être conforme à l'annexe G de l'IEEE 802.1AB-2005. ^j Doit être conforme à l'IEEE 802.1AB-2005, 9.5.9. Il est recommandé de régler l'identificateur d'objet conformément à 4.17.2.	
NOTE Il existe différentes sortes d'adresse MAC: l'adresse MAC du port utilisée comme SourceAddress et l'adresse MAC de l'interface utilisée comme CMResponderMacAdd ou CMInitiatorMacAdd.	

4.11.2 Syntaxe de transfert LLDP

4.11.2.1 Généralités

Dans le prolongement de l'IEEE 802.1AB-2005, 10.5.3.1, la présente norme stipule qu'une valeur nulle doit être attribuée au champ txDelayWhile. Dans ce cas, chaque nœud transmet les LLDP PDU à chaque modification de données.

4.11.2.2 Codage du champ LLDP_ChassisID

Ce champ doit être codé en type de données OctetString conformément au Tableau 261.

NOTE Le champ LLDP_ChassisID ne se termine pas par zéro.

Tableau 261 – LLDP_ChassisID conjointement avec MultipleInterfaceMode.NameOfDevice et NameOfStation

Multiple-InterfaceMode.NameOfDevice	NameOfStation exist	Type de données	Signification
0	Oui	OctetString[]	Ce champ doit être codé avec 1 à 240 octets conformément à 4.3.1.4.15.
0	Non	—	—
1	Oui	OctetString[]	Obligatoire ^a Il convient que ce champ soit codé avec "DeviceType, Blank, OrderID, Blank, IM_Serial_Number, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision"
			Facultatif ^b Ce champ doit être codé avec 1 à 240 octets conformément à 4.3.1.4.15.
1	Non	OctetString[]	Obligatoire ^a Il convient que ce champ soit codé avec "DeviceType, Blank, OrderID, Blank, IM_Serial_Number, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision"
			—

^a Obligatoire pour les nœuds à plusieurs interfaces et recommandé pour tous les autres nœuds.
^b Facultatif pour les nœuds à une seule interface qui ne posséderont jamais plusieurs interfaces.

4.11.2.3 Codage du champ LLDP_PortID

Ce champ doit être codé en type de données OctetString conformément au Tableau 262.

NOTE Le champ LLDP_PortID ne se termine pas par zéro.

Tableau 262 – LLDP_PortID conjointement avec MultipleInterfaceMode.NameOfDevice

Multiple-InterfaceMode.NameOfDevice	Type de données	Signification
0	OctetString[8] OctetString[14]	Le champ LLDP_PortID doit être codé conformément à 4.3.1.4.16
1	OctetString[]	Le champ LLDP_PortID doit être codé conformément à 4.3.1.4.17.1

4.11.2.4 Codage du champ LLDP_PNIO_SubType

Ce champ doit être codé en type de données Unsigned8 et doit être mis conformément au Tableau 263.

Tableau 263 – LLDP_PNIO_SubType

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Valeurs mesurées du délai
0x02	Statut du port
0x03	Alias
0x04	Statut du port MRP
0x05	Adresse MAC de l'interface
0x06	Statut PTCP
0x07 – 0xFF	Réservé

4.11.2.5 Codage du champ PTCP_PortRxDelayLocal

Ce champ doit être codé en type de données Unsigned32, la base de temps doit être de 1 ns et doit être conforme au Tableau 264.

Tableau 264 – PTCP_PortRxDelayLocal

Valeur (hexadécimale)	Signification
0x00	Inconnu
0x01 – 0x00000FFF	Délai du port RX local
0x00001000 – 0xFFFFFFFF	Réservé

4.11.2.6 Codage du champ PTCP_PortRxDelayRemote

Ce champ doit être codé en type de données Unsigned32, la base de temps doit être de 1 ns et doit être conforme au Tableau 265.

Tableau 265 – PTCP_PortRxDelayRemote

Valeur (hexadécimale)	Signification
0x00	Inconnu
0x01 – 0x00000FFF	Délai du port RX distant
0x00001000 – 0xFFFFFFFF	Réservé

4.11.2.7 Codage du champ PTCP_PortTxDelayLocal

Ce champ doit être codé en type de données Unsigned32, la base de temps doit être de 1 ns et doit être conforme au Tableau 266.

Tableau 266 – PTCP_PortTxDelayLocal

Valeur (hexadécimale)	Signification
0x00	Inconnu
0x01 – 0x00000FFF	Délai du port TX local
0x00001000 – 0xFFFFFFFF	Réservé

4.11.2.8 Codage du champ PTCP_PortTxDelayRemote

Ce champ doit être codé en type de données Unsigned32, la base de temps doit être de 1 ns et doit être conforme au Tableau 267.

Tableau 267 – PTCP_PortTxDelayRemote

Valeur (hexadécimale)	Signification
0x00	Inconnu
0x01 – 0x00000FFF	Délai du port TX distant
0x00001000 – 0xFFFFFFFF	Réservé

4.11.2.9 Codage du champ CableDelayLocal

Ce champ doit être codé en type de données Unsigned32, la base de temps doit être de 1 ns et doit être conforme au Tableau 268.

Tableau 268 – CableDelayLocal

Valeur (hexadécimale)	Signification
0x00	Inconnu
0x01 – 0x000FFFFF	Délai mesuré du câble
0x00100000 – 0xFFFFFFFF	Réservé

4.11.2.10 Codage du champ RTClass2_PortStatus

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 1: RTClass2_PortStatus.State

Ce champ doit être mis conformément au Tableau 269 ou au Tableau 270.

Tableau 269 – RTClass2_PortStatus.State avec ARProperties.StartupMode == Legacy

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Non utilisé ou non configuré si la valeur un est attribuée à LLDP_OrangePeriodBegin.Valid.
0x01	Réservé	—
0x02	RTCLASS2_RUN	Phase ORANGE activée pour la transmission et la réception de trames RTClass2
0x03	Réservé	—

Tableau 270 – RTClass2_PortStatus.State avec ARProperties.StartupMode == Advanced

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Non utilisé
autre	Réservé	—

Bit 2 – 15: RTClass2_PortStatus.reserved

Ce champ doit être mis conformément à 3.4.2.2.

4.11.2.11 Codage du champ RTClass3_PortStatus

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 2: RTClass3_PortStatus.State

Ce champ doit être mis conformément au Tableau 271.

Tableau 271 – RTClass3_PortStatus.State

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Non utilisé ou non configuré si la valeur un est attribuée à LLDP_RedOrangePeriodBegin.Valid. Période RED est désactivée.
0x01	Réservé	—
0x02	RTCLASS3_UP	Période ROUGE activée pour la transmission de trames RTClass3. Etat suivant prévu: RTCLASS3_RUN
0x03	Réservé	—
0x04	RTCLASS3_RUN	Période ROUGE activée pour la transmission et la réception de trames RTClass3
0x05 – 0x07	Réservé	—

Bit 3 – 11: RTClass3_PortStatus.reserved

Ce champ doit être mis conformément à 3.4.2.2.

Bit 12: RTClass3_PortStatus.Fragmentation

Ce champ doit être mis conformément au Tableau 272.

Tableau 272 – RTClass3_PortStatus.Fragmentation

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Par défaut Le mode de fragmentation pour ce port est désactivé.
0x01	ON	Le mode de fragmentation pour ce port est activé.

Les informations stipulant si la fragmentation est activée doivent être issues de PDIRGlobalData et de SendClockFactor.

Bit 13: RTClass3_PortStatus.PreambleLength

Ce champ doit être mis conformément au Tableau 273.

Tableau 273 – RTClass3_PortStatus.PreambleLength

Valeur (hexadécimale)	Signification	Utilisation
0x00	Sept octets	Par défaut Le PHY utilise un préambule de sept octets lors de la transmission
0x01	Un octet	Le PHY utilise un préambule d'un octet lors de la transmission

Il est recommandé qu'un PHY / MAC prenne en charge une longueur de préambule d'un octet lors de l'envoi avec MAUType 100BASETXFD.

Il convient qu'un PHY / MAC prenne en charge une longueur d'un octet lors de la réception avec MAUType 100BASETXFD.

Le Tableau 274 contient la table de vérité utilisée pour raccourcir le préambule.

Tableau 274 – Table de vérité pour raccourcir le préambule

Entrée		Sortie
RTC3PSM	AdjustPreambleLength. PreambleLength.Length	RTClass3_PortStatus. PreambleLength
EXPECTED_REMOTE(LLDP_ChassisID) == REMOTE(LLDP_ChassisID)	Sept octets	Sept octets
AND EXPECTED_REMOTE(LLDP_PortID) == REMOTE(LLDP_PortID)	Un octet	Un octet
AND EXPECTED_REMOTE(IRDataUUID) == REMOTE(LLDP_IRDataUUID)		
AND EXPECTED_LOCAL(MAUtype) == LOCAL(MAUtype)		
! (		
EXPECTED_REMOTE(LLDP_ChassisID) == REMOTE(LLDP_ChassisID)	—	Sept octets
AND EXPECTED_REMOTE(LLDP_PortID) == REMOTE(LLDP_PortID)		
AND EXPECTED_REMOTE(IRDataUUID) == REMOTE(LLDP_IRDataUUID)		
AND EXPECTED_LOCAL(MAUtype) == LOCAL(MAUtype)		
)		

Bit 14: RTClass3_PortStatus.reserved

Ce champ doit être mis conformément à 3.4.2.2.

Bit 15: RTClass3_PortStatus.Optimized

Ce champ doit être mis conformément au Tableau 275.

Tableau 275 – RTClass3_PortStatus.Optimized

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Par défaut
0x01	ON	Réservé

4.11.2.12 Codage du champ MRRT_PortStatus

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 1: MRRT_PortStatus.State

Ce champ doit être mis conformément au Tableau 276.

Tableau 276 – MRRT_PortStatus.State

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Non utilisé / non configuré
0x01 – 0x03	Réservé	—

Bit 2 – 15: MRRT_PortStatus.reserved

Ce champ doit être mis conformément à 3.4.2.2.

4.11.2.13 Codage du champ IRDataUUID

Ce champ doit être codé en type de données UUID conformément au Tableau 277.

Tableau 277 – IRDataUUID

Valeur (UUID)	Signification	Utilisation
00000000-0000-0000-0000-000000000000	Pas de domaine RT_CLASS_3	Valeur initiale
00000000-0000-0000-0000-000000000001 – FFFFFFFF-FFFF-FFFF-FFFF-FFFFFFFFFFFFFFE	UUID pour un domaine RT_CLASS_3	Générée par l'ingénierie et utilisée par l'appareil de commande E/S et l'appareil E/S pour identifier un domaine RT_CLASS_3

4.11.2.14 Codage du champ LLDP_RedOrangePeriodBegin

Comme pour RT_CLASS_3, l'utilisation d'un port est répartie sur deux périodes différentes. De même, l'utilisation d'un port est répartie entre la transmission et la réception. Le codage de ce champ doit être conforme à 3.4.2.5 et à la Figure 21. Les bits individuels doivent avoir la signification ci-dessous:

Bit 0 – 30: LLDP_RedOrangePeriodBegin.Offset

Ce champ doit être mis conformément au Tableau 278.

Tableau 278 – LLDP_RedOrangePeriodBegin.Offset

Valeur (hexadécimale)	Signification	Utilisation
0x00000000 – 0x003D08FF	Décalage par rapport au début du cycle, en nanosecondes	Début de la période RT_CLASS_3 du sens de transmission du port. Si RedPeriod est défini, la valeur doit être déduite de la valeur minimale de RedOrangePeriodBegin. Sinon elle doit être déduite de ReservedIntervalBegin.
0x003D0900 – 0x7FFFFFFF	Réservé	—

Bit 31: LLDP_RedOrangePeriodBegin.Valid

Ce champ doit être mis conformément au Tableau 279.

Tableau 279 – LLDP_RedOrangePeriodBegin.Valid

Valeur (hexadécimale)	Signification	Utilisation
0x00	Non valide	La valeur de LLDP_RedOrangePeriodBegin.Offset n'est pas valide. Elle doit être mise à zéro.
0x01	Valide	La valeur de LLDP_RedOrangePeriodBegin.Offset est valide.

4.11.2.15 Codage du champ LLDP_OrangePeriodBegin

NOTE Utilisé uniquement avec ARProperties.StartupMode == Legacy.

Le codage de ce champ doit être conforme à 3.4.2.5 et à la Figure 21. Les bits individuels doivent avoir la signification ci-dessous:

Bit 0 – 30: LLDP_OrangePeriodBegin.Offset

Ce champ doit être mis conformément au Tableau 280.

Tableau 280 – LLDP_OrangePeriodBegin.Offset

Valeur (hexadécimale)	Signification	Utilisation
0x00000000 – 0x003D08FF	Décalage par rapport au début du cycle, en nanosecondes	Début de la période RT_CLASS_2 du sens de transmission du port. Si RedPeriod est défini, la valeur doit être déduite de la valeur maximale d'OrangePeriodBegin. Sinon elle doit être déduite de ReservedIntervalBegin.
0x003D0900 – 0x7FFFFFFF	Réservé	—

Bit 31: LLDP_OrangePeriodBegin.Valid

Ce champ doit être mis conformément au Tableau 281 ou au Tableau 282.

Tableau 281 – LLDP_OrangePeriodBegin.Valid avec ARProperties.StartupMode == Legacy

Valeur (hexadécimale)	Signification	Utilisation
0x00	Non valide	La valeur de LLDP_OrangePeriodBegin.Offset n'est pas valide. Elle doit être mise à zéro.
0x01	Valide	La valeur de LLDP_OrangePeriodBegin.Offset est valide.

Tableau 282 – LLDP_OrangePeriodBegin.Valid avec ARProperties.StartupMode == Advanced

Valeur (hexadécimale)	Signification	Utilisation
0x00	Non valide	LLDP_OrangePeriodBegin.Offset n'est pas valide. Elle doit être mise à zéro.
autre	—	Réservé

4.11.2.16 Codage du champ LLDP_GreenPeriodBegin

Comme pour RT_CLASS_1, RT_CLASS_2, RT_CLASS_UDP et les autres protocoles, l'utilisation d'un port est répartie sur deux périodes différentes. De même, l'utilisation d'un port est répartie entre la transmission et la réception. Le codage de ce champ doit être conforme à 3.4.2.5 et à la Figure 21. Les bits individuels doivent avoir la signification ci-dessous:

Bit 0 – 30: LLDP_GreenPeriodBegin.Offset

Ce champ doit être mis conformément au Tableau 283.

Tableau 283 – LLDP_GreenPeriodBegin.Offset

Valeur (hexadécimale)	Signification	Utilisation
0x00000000 – 0x003D08FF	Décalage par rapport au début du cycle, en nanosecondes	Début de la période sans restriction du sens de transmission du port. La valeur doit être déduite de ReservedIntervalEnd.
0x003D0900 – 0x7FFFFFFF	Réservé	—

Bit 31: LLDP_GreenPeriodBegin.Valid

Ce champ facultatif doit être mis conformément au Tableau 284.

Tableau 284 – LLDP_GreenPeriodBegin.Valid

Valeur (hexadécimale)	Signification	Utilisation
0x00	Non valide	La valeur de LLDP_GreenPeriodBegin.Offset n'est pas valide. Elle doit être mise à zéro.
0x01	Valide	La valeur de LLDP_GreenPeriodBegin.Offset est valide.

4.11.2.17 Codage du champ LLDP_LengthOfPeriod

Un port est réparti en deux périodes différentes. La durée de chacune des périodes est indiquée par ce champ. Le codage de ce champ doit être conforme à 3.4.2.5 et à la Figure 21. Les bits individuels doivent avoir la signification ci-dessous:

Bit 0 – 30: LLDP_LengthOfPeriod.Length

Ce champ doit être mis conformément au Tableau 285.

Tableau 285 – LLDP_LengthOfPeriod.Length

Valeur (hexadécimale)	Signification	Utilisation
0x00007A12 – 0x003D0900	Durée d'un cycle, en nanosecondes	La valeur doit être un multiple de 31 250 ns. Voir 5.2.4.62
0x003D0900 – 0x7FFFFFFF	Réservé	—

Bit 31: LLDP_LengthOfPeriod.Valid

Ce champ facultatif doit être mis conformément au Tableau 286.

Tableau 286 – LLDP_LengthOfPeriod.Valid

Valeur (hexadécimale)	Signification	Utilisation
0x00	Non valide	LLDP_LengthOfPeriod.Length n'est pas valide. Elle doit être mise à zéro.
0x01	Valide	LLDP_LengthOfPeriod.Length est valide.

4.11.2.18 Codage du champ LLDP_POINT

Ce champ doit être codé en type de données OctetString avec un octet. La valeur doit être ".".

NOTE Le champ LLDP_POINT ne se termine pas par zéro.

4.11.2.19 Codage du champ LLDP_TimeToLive

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.20 Codage du champ LLDP_TLVHeader

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.21 Codage du champ LLDP_ChassisIDSubType

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.22 Codage du champ LLDP_PortIDSubType

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.23 Codage du champ LLDPOption

Ce champ doit être codé conformément à IEEE 802.1AB-2005, 9.4.

4.11.2.24 Codage du champ LLDP_OUI

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.25 Codage du champ LLDP_8021_SubType

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.26 Section de codage relative à LLDP_Management

4.11.2.26.1 Codage du champ LLDP_ManagementData

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.27 Section de codage relative à LLDP_8023

4.11.2.27.1 Codage du champ LLDP_8023_SubType

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.27.2 Codage du champ LLDP_8023_AUTONEG

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.27.3 Codage du champ LLDP_8023_PMDCAP

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.2.27.4 Codage du champ LLDP_8023_OPMAU

Ce champ doit être codé conformément à l'IEEE 802.1AB-2005.

4.11.3 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.11.4 Machines de Protocole de relation entre applications

Il n'existe aucune Machine de Protocole de relation entre applications (ARPM) définie pour ce Protocole.

4.11.5 Machines de Protocole de Mapping DLL

Il n'existe aucune Machine de Protocole de Mapping DLL (DMPM) définie pour ce Protocole.

4.12 Ponts MAC

4.12.1 Vue d'ensemble

Les concepts conformes à l'IEEE 802.1D doivent être appliqués.

Selon ces concepts, chaque pont MAC ne doit pas transférer de trames lorsque SourceMACAddress lui appartient et doit prendre en charge le transfert de trames à la vitesse des fils.

Outre le comportement de l'IEEE 802.1D, la spécification du protocole définit les machines de protocole suivantes exécutant des actions de transfert particulières:

- RT_CLASS_3 Machine de protocole de transfert (RED_RELAY)
- Machine de Protocole de transfert DFP (DFP_RELAY)
- Machine de protocole de transfert de synchronisation (SYNC_RELAY)

De plus, les mécanismes d'apprentissage de l'IEEE 802.1D doivent être désactivés conformément au Tableau 287, au Tableau 288 et au Tableau 289. A des fins d'optimisation, une entrée du FDB précédemment acquise peut être réactualisée.

L'utilisation du principe de "Cassure" pour le transfert de trames nécessite une gestion des erreurs tardives pour éviter les fragments de trame dans un réseau commuté. Par conséquent, lorsqu'une trame non valide (par FCS ou SFCRC16), déjà partiellement transférée, est détectée, il convient qu'elle soit raccourcie en ajoutant une erreur TX (voir Annexe S) au port de transmission.

4.12.2 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.12.3 Machines de Protocole de relation entre applications

Il n'existe aucune Machine de Protocole de relation entre applications (ARPM) définie pour ce Protocole.

4.12.4 Machines de Protocole de Mapping DLL

4.12.4.1 Forwarding Protocol Machine (Machine de protocole de transfert)

4.12.4.1.1 Base de données de filtrage

Conformément à l'IEEE 802 et à la présente norme, un commutateur contient une base de données de filtrage (FDB) visant à optimiser le pontage. Le Tableau 287, le Tableau 288 et le Tableau 289 présentent la liste des entrées.

La ligne ORG avec une entrée "OUI" indique si ce protocole peut être reçu ou transmis, même si un port est exclu de la topologie active par MRP ou RSTP.

NOTE Si la CEI 62439-2 est prise en charge, elle ajoute des entrées supplémentaires à la FDB.

Tableau 287 – Entrées FDB à diffusion unique

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Application	Pont	ORG	Learning	Signification
Vendor	n	Réception ^a	Filtre	Non	Activé	Adresse MAC de l'interface
Vendor	n+1 – n+m	Réception ^a	Filtre	Non	Activé	Adresses MAC du port
Others	Others	—	—	—	—	

^a Si applicable

Tableau 288 – Entrées FDB multidiffusion

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Application	Pont	ORG	Learning	Signification
01-0E-CF	00-00-00	Réception ^a	Transfert	Non	Activé	Avec FrameID=0xFEFE utilisé pour DCP-Identify-ReqPDU
01-0E-CF	00-00-01	Réception ^a	Transfert	Non	Activé	Avec FrameID=0xFEFC utilisé pour DCP-Hello-ReqPDU
01-0E-CF	00-01-01	Réception ^a	Filtre ^b	Non	Désactivé	Adresse multidiffusion de destination RT_CLASS_3
01-0E-CF	00-01-02	Réception ^a	Filtre ^b	Non	Désactivé	Adresse multidiffusion de trame non valide RT_CLASS_3
01-0E-CF	00-02-00 – 00-02-1F	Réception ^a	Transfert ^c	Non	Activé	Adresse de communication multidiffusion RT_CLASS_2
01-0E-CF	00-02-20 – 00-02-FF	Réception ^a	Transfert	Non	Activé	Adresse de communication multidiffusion RT_CLASS_2
01-0E-CF	00-04-00	Réception ^a	Transfert	Non	Désactivé	Conjointement avec PTCP-AnnouncePDU et FrameID (=0xFF00) utilisé pour synchroniser l'horloge de travail
01-0E-CF	00-04-20	Réception ^a	Filtre	Oui	Désactivé	Conjointement avec PTCP-RTSyncPDU avec suivi et FrameID(=0x0020) utilisé pour la synchronisation de l'horloge de travail
01-0E-CF	00-04-40	Réception ^a	Filtre	Oui	Désactivé	Conjointement avec PTCP-FollowUpPDU et FrameID (=0xFF20) utilisé pour la synchronisation de l'horloge de travail
01-0E-CF	00-04-80	Réception ^a	Filtre	Oui	Désactivé	Conjointement avec PTCP-RTSyncPDU et FrameID (=0x0080) utilisés pour la synchronisation de l'horloge
X3-XX-00	00-00-00	Réception ^a	Filtre ^b	Non	Désactivé	Adresse de multidiffusion de destination RT_CLASS_3 de transfert rapide
01-80-C2	00-00-00	Réception ^a	Filtre ^d	Oui	Désactivé	IEEE 802.1D Protocole RSTP (Rapid Spanning Tree Protocol)
01-80-C2	00-00-01 – 00-00-0D	Réception ^a	Filtre	Oui	Désactivé	IEEE 802.1D

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Application	Pont	ORG	Learning	Signification
01-80-C2	00-00-0E	Réception ^a	Filtre	Oui	Désactivé	IEEE 802.1D, IEEE 802.1AS, IEEE 802.1AB (LLDP), et PTCP Conjointement avec PTCP-DelayReqPDU et FrameID(=0xFF40), PTCP-DelayResPDU avec suivi et FrameID(=0xFF41), PTCP-DelayEuResPDU et FrameID(=0xFF42) et PTCP-DelayResPDU sans suivi et FrameID(=0xFF43) utilisés pour la mesure du délai
01-80-C2	00-00-0F	Réception ^a	Filtre	Oui	Désactivé	IEEE 802.1D
01-80-C2	00-00-10	Réception ^a	Filtre	Oui	Désactivé	IEEE 802.1D Adresse de groupe de gestion de tous les ponts LAN
01-00-5E	40-F8-00 – 40-FB-FF	Réception ^a	— ^e	Non	Activé	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP
Others	Others	—	—	—	—	—
^a Si applicable ^b Ces trames doivent être écartées par le MAC Relay si RT_CLASS_3 est prise en charge, sinon il convient qu'elles soient écartées ^c Le transfert dépend de la configuration de MulticastBoundary ^d Il est recommandé de transférer les trames du protocole RSTP (Rapid Spanning Tree Protocol) dans les commutateurs sans prise en charge de RSTP ou dans les commutateurs avec des protocoles de redondance de support alternatifs tels que MRP ^e Il est recommandé de transférer ces trames sauf si RFC 4604 (IGMPv3) est pris en charge						

Tableau 289 – Entrée FDB Multidiffusion

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Application	Pont	ORG	Learning	Signification
FF-FF-FF	FF-FF-FF	Réception ^a	Transfert	Non	Activé	Broadcast (Diffusion)
Others	Others	—	—	—	—	—
^a Si applicable						

4.12.4.1.2 Définitions de primitives

Les primitives de service, y compris leurs paramètres associés, émises par l'utilisateur de MAC_RELAY, reçues par MAC_RELAY et inversement, sont décrites dans l'ASE des ponts MAC de la définition de service.

4.12.4.1.2.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par MAC_RELAY sont décrites dans la définition de service et présentées dans le Tableau 290.

Tableau 290 – Primitives distantes émises ou reçues par MAC_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
MAC_RELAY_Data.req	LMPM	MAC_RELAY	Frame	—
MAC_RELAY_Data.cnf	LMPM	MAC_RELAY	Frame, Status	—
MAC_RELAY_Data.ind	MAC_RELAY	LMPM	Port, Frame, Timestamp	—
DEMUX_MAC_RELAY.ind	DEMUX	MAC_RELAY	Port, Frame	—
QueueHandler.req	MAC_RELAY	QueueHandler	Queue, Frame	—

4.12.4.1.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par MAC_RELAY sont décrites dans la définition de service et présentées dans le Tableau 291.

Tableau 291 – Primitives locales émises ou reçues par MAC_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.12.4.1.3 Schéma de transition d'états

Le schéma de transition d'états de MAC_RELAY doit être conforme à l'IEEE 802.1D.

4.12.4.1.4 Description du diagramme d'états

Le diagramme d'états de MAC_RELAY doit être conforme à l'IEEE 802.1D.

4.12.4.1.5 Table d'états MAC_RELAY

La table d'états doit être conforme à l'IEEE 802.1D.

4.12.4.1.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 292 contient les fonctions, macros, temporisateurs et variables utilisés par le MAC_RELAY ainsi que leurs arguments et leurs descriptions.

Tableau 292 – Fonctions, Macros, Temporisateurs et Variables utilisés par le MAC_RELAY

Nom	Type	Fonction/Signification
—	—	—

4.12.4.2 Diagramme d'états du port RT_CLASS_3

4.12.4.2.1 Généralités

Cette partie de la spécification définit l'utilisation de l'IEEE 802.1D.

Elle contient des prolongements de la phase RT_CLASS_3 relatifs à la transmission des trames.

Les mécanismes d'acheminement présentés dans l'IEEE 802.1D doivent être provisoirement désactivés dans la phase RT_CLASS_3. Dans ce cas, l'acheminement des trames RT_CLASS_3 doit être conforme aux attributs définis dans l'ASE approprié. Les valeurs des attributs définissent une table d'acheminement particulière utilisée.

4.12.4.2.2 Définitions de primitives

4.12.4.2.2.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par le Diagramme d'Etats du Port de Classe 3 en Temps Réel (RTC3PSM) sont décrites dans la définition de service et présentées dans le Tableau 293.

Tableau 293 – Primitives distantes émises ou reçues par RTC3PSM

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.12.4.2.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par RTC3PSM sont décrites dans la définition de service et présentées dans le Tableau 294.

Tableau 294 – Primitives locales émises ou reçues par RTC3PSM

Primitive	Source	Destination	Paramètres associés	Fonctions
Remote_Systems_Data_Change_ind	LLDP	RTC3PSM	Local Port ID, List of Neighbors	—
MAU_Type_Change_ind	IEEE 802.3	RTC3PSM	Local Port ID, Local MAU Type, Local Link Status	—
Port_State_Change_ind	IEEE 802.1D	RTC3PSM	Local PortState	—
Sync_State_Change_ind	PTCP	RTC3PSM	Sync State	—
SetPortState	RTC3PSM	IEEE 802.1D	Local PortState	—

4.12.4.2.3 Schéma de transition d'états

Le schéma de transition d'états du RTC3PSM est présenté à la Figure 67.

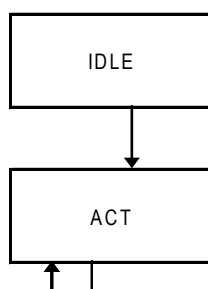


Figure 67 – Schéma de transition d'états de RTC3PSM

Etats du RT3PSM

IDLE	Etat inactif
ACT	Etat actif

4.12.4.2.4 Description du diagramme d'états

Le Diagramme d'Etats du Port de Classe 3 en Temps Réel (RTC3PSM) organise les états du port de classe 3 en temps réel.

4.12.4.2.5 Table d'états RTC3PSM

La table d'états RTC3PSM apparaît au Tableau 295 et doit exister pour chacun des ports. L'allocation de la largeur de bande à l'aide de TXBeginEndAssignment et de RXBeginEndAssignment doit être couplée aux états du port RTC3PSM.

Tableau 295 – Table d'états RTC3PSM

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	IDLE	=> EvaluateTruthTable() SetPortState ()	ACT
2	ACT	Remote_Systems_Data_Change_ind () => REMOTEPORTSTATE NEIGHBOR ENGINEERING EvaluateTruthTable() SetPortState ()	ACT
3	ACT	MAU_Type_Change_ind () => PORT EvaluateTruthTable() SetPortState ()	ACT
4	ACT	Sync_State_Change_ind () => SYNC EvaluateTruthTable() SetPortState ()	ACT
5	ACT	Port_State_Change_ind () => PORT EvaluateTruthTable() SetPortState ()	ACT

4.12.4.2.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 296 contient les fonctions, macros, temporisateurs et variables utilisés par le RTC3PSM ainsi que leurs arguments et leurs descriptions.

Tableau 296 – Fonctions, Macros, Temporisateurs et Variables utilisés par le RTC3PSM

Nom	Type	Fonction/Signification
EvaluateTruthTable	Fonction	Cette fonction est utilisée comme déclencheur pour évaluer la table de vérité correspondante.
If activated	Fonction	Cette vérification est activée par un sous-bloc de l'enregistrement PDPortDataCheck.
CHECK_DATA	Macro	Cette fonction donne les informations stipulant si les données stockées en local sont valides.
DIFFERENCE_FAULT	Macro	Cette fonction vérifie si la différence entre les CableDelay mesurés local et distant est inférieure à 50 ns.

Nom	Type	Fonction/Signification
EXPECTED_LOCAL	Macro	Cette fonction délivre les informations stockées localement qui sont prévues pour être atteintes localement.
EXPECTED_REMOTE	Macro	Cette fonction délivre les informations stockées localement qui sont prévues d'être délivrées à distance.
GET_DOMAIN_BOUNDARY_VALUE	Macro	Cette fonction délivre les informations locales, stockées dans PDIRGlobalData et PDIRBeginEndData, que ce port soit ou non utilisé comme RT_CLASS_3.
LOCAL	Macro	Cette fonction délivre les informations locales.
NEIGHBOR	Macro	<pre> ! (// physical if activated: EXPECTED_REMOTE(LLDP_ChassisID) != REMOTE(LLDP_ChassisID) OR if activated: EXPECTED_REMOTE(LLDP_PortID) != REMOTE(LLDP_PortID) OR REMOTE(MAUType) != VALID^a OR if activated: (EXPECTED_REMOTE(MAUType) != LOCAL(MAUType)) OR if activated: (LOCAL(LineDelay) > EXPECTED_LOCAL(LineDelay)) OR if activated: (EXPECTED_REMOTE(PreambleLength) != REMOTE(PreambleLength)) OR if activated: (EXPECTED_REMOTE(Fragmentation) != REMOTE(Fragmentation))) </pre> ^a Cette fonction retourne la valeur VALID pour le paramètre MAUType, si la vitesse est supérieure ou égale à 100 Mbit/s et si la duplicité est totalement conforme à 6.2.12.7.
PORT	Macro	<pre> ! (// physical LOCAL(MAUType) != VALID^a OR if activated: (EXPECTED_LOCAL(MAUType) != LOCAL(MAUType)) OR LOCAL(PortState) == DISCARDING OR LOCAL(LinkStatus) != UP OR // logical LOCAL(GET_DOMAIN_BOUNDARY_VALUE()) == TRUE) </pre> ^a Cette fonction retourne la valeur VALID pour le paramètre MAUType, si la vitesse est supérieure ou égale à 100 Mbit/s et si la duplicité est totalement conforme au Tableau 551.
REMOTE	Macro	Cette fonction délivre les informations distantes stockées localement.
REMOTEPORTSTATE	Macro	<pre> (// physical Does the remote RT_CLASS_3_Port_State exist and contain a value out of the following list {OFF, RTCLASS3_UP, RTCLASS3_RUN}) </pre>
SYNC	Macro	<pre> ! (// physical LOCAL(SYNC) == FALSE OR // logical if activated: (DIFFERENCE_FAULT(LOCAL(CableDelayLocal), REMOTE(CableDelayRemote)))) </pre>

4.12.4.2.7 Table de vérité

Le Tableau 297 et le Tableau 298 contiennent la table de vérité utilisée par le RTC3PSM.

Tableau 297 – Table de vérité pour le RTC3PSM

Entrée				Sortie		
Condition du port d'émission				Etat du port distant	Etat du port local	Etat du prochain port local
PORT	NEIGHBOR	SYNC	ENGINEERING ^b	—	—	—
FALSE	—	—	—	—	—	OFF
TRUE	FALSE	—	—	—	—	OFF
TRUE	TRUE	FALSE	—	—	—	OFF
TRUE	TRUE	TRUE	FALSE	—	—	OFF
TRUE	TRUE	TRUE	TRUE	OFF	—	UP
TRUE	TRUE	TRUE	TRUE	UP	—	RUN
TRUE	TRUE	TRUE	TRUE	RUN	— ^a	RUN

^a L'état local "OFF" en combinaison avec l'état du port distant "RUN" n'est pas conforme mais est possible en tant qu'état transitoire. Même dans ce cas, il convient que l'état local suivant soit "RUN".

^b PDIRData et le cas échéant PDIRSubframeData existent et sont valides.

Tableau 298 – RXBeginEndAssignment et TXBeginEndAssignment

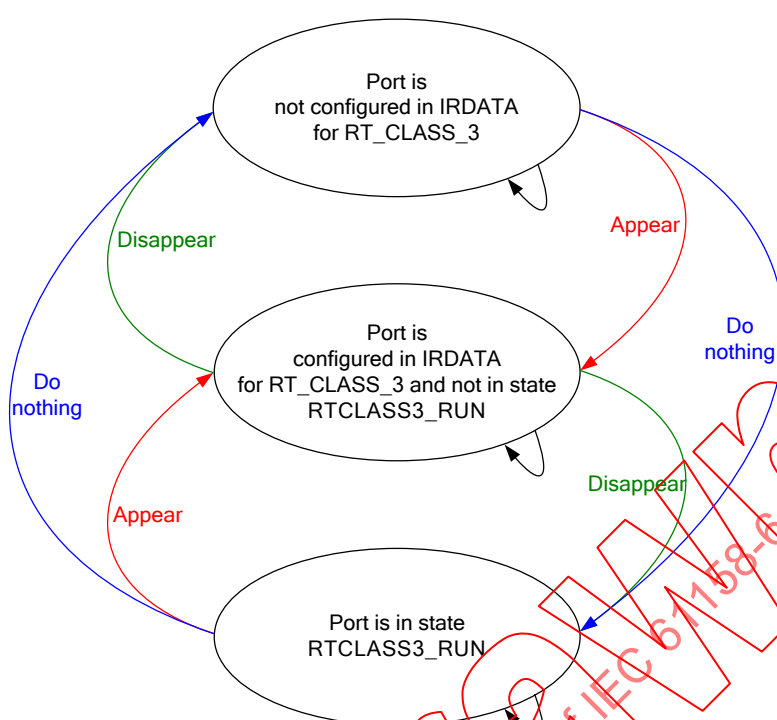
Entrée		Sortie	
Etat du port local	Etat du port distant	Local RXBeginEndAssignment	Local TXBeginEndAssignment
OFF	—	OFF	OFF
UP	—	OFF	ON ^a
RUN	UP	ON	ON ^a
RUN	RUN	ON	ON ^b

^a Pour protéger l'installation d'un RXBeginEndAssignment distant, il convient que le TXBeginEndAssignment local commence par le RedOrangePeriodBegin:= 0, même si le PDIRBeginEndData indique le contraire.

^b Après l'installation d'un RXBeginEndAssignment distant, il convient que le TXBeginEndAssignment local commence par le RedOrangePeriodBegin du PDIRBeginEndData.

4.12.4.2.8 Génération d'événements

Le RTC3PSM présente les modifications d'état nécessaires à l'établissement d'une communication entre pairs RT_CLASS_3. De plus, la Figure 68 illustre les événements de diagnostic générés en fonction des modifications d'état de RTC3PSM. Le Tableau 299 définit le codage des événements associés.

**Légende**

Anglais	Français
Port is not configured in IRDATA for RT_CLASS_3	Le port n'est pas configuré dans IRDATA pour RT_CLASS_3
Disappear	Disappear (Disparaître)
Appear	Appear (Apparaître)
Port is configured in IRDATA for RT_CLASS_3 and not in state RTCLASS3_RUN	Le port est configuré dans IRDATA pour RT_CLASS_3 et n'est pas à l'état RTCLASS3_RUN
Do nothing	Do nothing (Ne rien faire)
Port is in state RTCLASS3_RUN	Le port est à l'état RTCLASS3_RUN

Figure 68 – Schéma de transition d'états pour générer des événements

Tableau 299 – Table de fonction d'événement

Nom de la fonction	Opérations
Appear (Apparaître)	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:=Port Data Change Notification Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelErrorType := "Remote Mismatch" ExtChannelErrorType := "RT Class3 mismatch" Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Disappear (Disparaître)	Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:=Port Data Change Notification Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Disappears ChannelErrorType := "Remote Mismatch" ExtChannelErrorType := "RT Class3 mismatch" Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, Alarm Item)
Do nothing (Ne rien faire)	Ce changement d'état ne doit indiquer aucun événement

4.12.4.3 RT_CLASS_3 Forwarding Protocol Machine (Machine de protocole de transfert de RT_CLASS_3)

4.12.4.3.1 Définitions de primitives

4.12.4.3.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Transfert de Classe 3 en Temps Réel (RED_RELAY) sont décrites dans la définition de service et présentées dans le Tableau 300.

Tableau 300 – Primitives distantes émises ou reçues par RED_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_RED_RELAY.ind	RED_RELAY, DEMUX	RED_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—
RED_RELAY_Data.ind	RED_RELAY	LMPM	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—
QueueHandler.req	RED_RELAY	QueueHandler	ROUGE	Cette fonction met la trame dans la file d'attente RED de QueueHandler. Le MUX retire la trame de la file d'attente et l'achemine au MAC.

4.12.4.3.1.2 Primitives échangées entre les machines locales

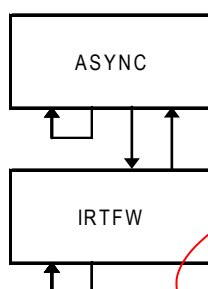
Les primitives de service locales, y compris leurs paramètres associés émises ou reçues par RED_RELAY sont décrites dans la définition de service et présentées dans le Tableau 301.

Tableau 301 – Primitives locales émises ou reçues par RED_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
RED_RELAY_Error_Ind	RED_RELAY	Local matter	CREP, ErrorCode	—
SyncStateChange_ind	PTCP	RED_RELAY	SYNC	—

4.12.4.3.2 Schéma de transition d'états

Le schéma de transition d'états du RED_RELAY est présenté à la Figure 69.

**Figure 69 – Schéma de transition d'états de RED_RELAY**

Etats du RED_RELAY

ASYNC

Le nœud doit être synchronisé, soit en local en tant que maître de synchronisation ou à distance en tant qu'esclave de synchronisation pour le transfert RED.

IRTFW

Si le nœud est synchronisé, le transfert et la réception locale des trames RED à l'aide de PDIRFrameData ou de PDIRGlobalData et PDIRBeginEndData doivent être effectués.

4.12.4.3.3 Description du diagramme d'états

La Machine de Protocole de Classe 3 en Temps Réel (RED_RELAY) organise le transfert et la réception locale des trames RED.

4.12.4.3.4 Table d'états RED_RELAY

Le RED_RELAY doit garantir que le transfert des trames RT_CLASS_3 est effectué correctement dans les délais. Cela signifie que la trame doit être transférée conformément au FrameSendOffset donné (voir Figure 29 pour le point de référence FrameSendOffset), pour fonctionner avec tous les types de RedGuards.

Les appareils prenant en charge le RedGuard avec une vérification complète peuvent générer des trames substitués (voir Tableau 169 pour les valeurs utilisées de l'APDU_Status.TransferStatus), si une RED_RELAY_Error_Ind est détectée ou qu'il manque une trame.

Les données nécessaires pour RED_RELAY sont données par PDIRFrameData ou générées localement à partir de PDIRGlobalData.

La table d'états RED_RELAY apparaît dans le Tableau 302.

Tableau 302 – Table d'états RED_RELAY

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	ASYN	SyncStateChange_ind () /SYNC => ignore	IRTFW
2	ASYN	SyncStateChange_ind () /!SYNC => ignore	ASYN
3	ASYN	TickEvent () => ignore	ASYN
4	ASYN	DMUX_RED_RELAY.ind () => ignore	ASYN
5	IRTFW	SyncStateChange_ind () /SYNC => ignore	IRTFW
6	IRTFW	SyncStateChange_ind () /!SYNC => ignore	ASYN
7	IRTFW	TickEvent (CycleCounter, CycleOffset) => Calculate Local_Phase as input for the RedGuard Calculate Local_FrameSendOffset as input for the RedGuard	IRTFW
8	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == WrongTxPortState => ErrorCode := WRONG_TX_STATE RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
9	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == Data length => ErrorCode := WRONG_LENGTH RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
10	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == FrameID OR Reason == Receive Port OR Reason == Phase => ErrorCode := WRONG_FRAMEID RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
11	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == DISCARD, Reason == FrameSendOffset => ErrorCode := WRONG_FrameSendOffset RED_RELAY_Error_Ind (CREP, ErrorCode)	IRTFW
12	IRTFW	DMUX_RED_RELAY.ind () /RedGuard () == FORWARD, List of TX Ports => StartTimer (IRT, Expected FrameSendOffset) For (TimerExpired (IRT), List of TX ports) if TX Port == 0 RED_RELAY_Data.ind () else QueueHandler.req ()	IRTFW

4.12.4.3.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 303 contient les fonctions, macros, temporisateurs et variables utilisés par le RED RELAY ainsi que leurs arguments et leurs descriptions.

Tableau 303 – Fonctions, Macros, Temporisateurs et Variables utilisés par le RED RELAY

Nom	Type	Fonction/Signification
RedGuard	Fonction	Cette fonction vérifie la trame reçue par rapport aux informations locales.
StartTimer	Fonction	Lance le temporisateur IRT qui contrôle la transmission d'une trame effectuée correctement dans les délais
TickEvent	Fonction	CycleCounter, CycleOffset
TimerExpired	Fonction	Contrôle la transmission d'une trame correctement effectuée dans les délais
Calculate Local_FrameSendOffset as input for the RedGuard	Macro	Calcule le paramètre d'entrée Local_FrameSendOffset pour le RedGuard
Calculate Local_Phase as input for the RedGuard	Macro	Calcule le paramètre d'entrée Local_Phase pour le RedGuard
IRT_Timer	Temporisateur	Temporisateur utilisé pour contrôler la transmission d'une trame effectuée correctement dans les délais par la génération de l'événement FrameSendOffset
Expected FrameSendOffset	Variable	FrameSendOffset prévu ou calculé de la trame
Local_FrameSendOffset	Variable	Cette variable locale contient la valeur réelle de FrameSendOffset induite par TickEvent
Local_Phase	Variable	Cette variable locale contient la valeur réelle de Phase induite par TickEvent

4.12.4.3.6 RedGuard

Le RedGuard vérifie la trame reçue (qui est "Reçue en RED" en raison du DEMUX) par rapport au Tableau 304, au Tableau 305 ou au Tableau 306. Le tableau utilisé dépend de la fonction prise en charge en local.

Tableau 304 – Table de vérité pour le RedGuard avec vérification complète

Entrée					Sortie		
Conditions du port de réception					Conditions du port d'émission	Local	
FrameID ^a	Port de réception	Phase	Longueur des données	FrameSend Offset ^c	Liste des ports TX	RTC3PSM distant	Relais RED
FALSE	—	—	—	—	—	—	Ecart ^b
TRUE	FALSE	—	—	—	—	—	Ecart ^b
TRUE	TRUE	FALSE	—	—	—	—	Ecart ^b
TRUE	TRUE	TRUE	FALSE	—	—	—	Ecart ^b
TRUE	TRUE	TRUE	TRUE	FALSE	—	—	Ecart ^b
TRUE	TRUE	TRUE	TRUE	TRUE	—	! RUN	Ecart ^b
TRUE	TRUE	TRUE	TRUE	TRUE	Oui	RUN	Transfert

^a Le FrameID doit être vérifié par rapport au PDIRFrameData et PDIRBeginEndData.

^b Il convient que le pont écarte la trame.

^c Le transfert de la trame pourrait avoir lieu en conservant (TRUE, vrai) ou non (FALSE, faux) son FrameSendOffset.

Tableau 305 – Table de vérité pour le RedGuard avec vérification réduite

Entrée		Sortie		
Conditions du port de réception		Conditions du port d'émission		Local
FrameID ^a	FrameSend Offset ^c	Liste des ports TX	RTC3PSM distant	Relais RED
FALSE	—	—	—	Ecart ^b
TRUE	FALSE	—	—	Ecart ^b
TRUE	TRUE	—	! RUN	Ecart ^b
TRUE	TRUE	Oui	RUN	Transfert
^a Le FrameID doit être vérifié par rapport au PDIRFrameData et au PDIRBeginEndData. ^b Il convient que le pont écarte la trame. ^c Le transfert de la trame pourrait avoir lieu en conservant (TRUE vrai) ou non (FALSE, faux) son FrameSendOffset.				

Tableau 306 – Table de vérité pour le RedGuard avec vérification minimale

Entrée		Sortie		
Conditions du port de réception		Conditions du port d'émission		Local
FrameID ^a		Liste des ports TX	RTC3PSM distant	Relais RED
FALSE	—	—	—	Ecart ^b
TRUE	—	—	! RUN	Ecart ^b
TRUE	Oui	Oui	RUN	Transfert
^a Le FrameID doit être vérifié par rapport au PDIRFrameData (pour les trames reçues localement et non transférées) et au PDIRBeginEndData. ^b Il convient que le pont écarte la trame.				

4.12.4.4 Machine de Protocole de Transfert de DFP

4.12.4.4.1 Généralités

Le DFP_RELAY organise les trames avec l'attribut DFP. Une trame DFP est une forme particulière d'une trame RT_CLASS_3.

DFP offre deux principes différents, entrant et sortant, pour la prise en charge de ce type de trames. La Figure 70, la Figure 71 et la Figure 72 montrent l'intégration de DFP.

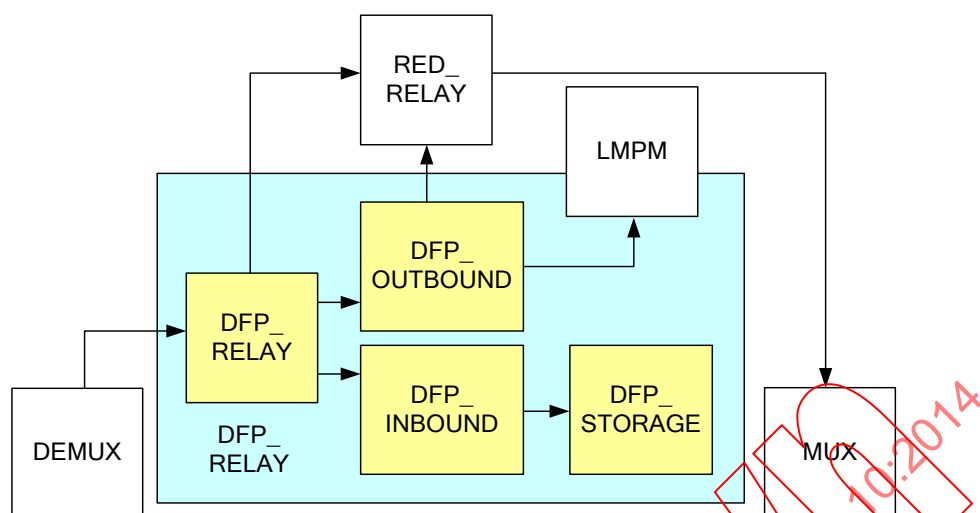
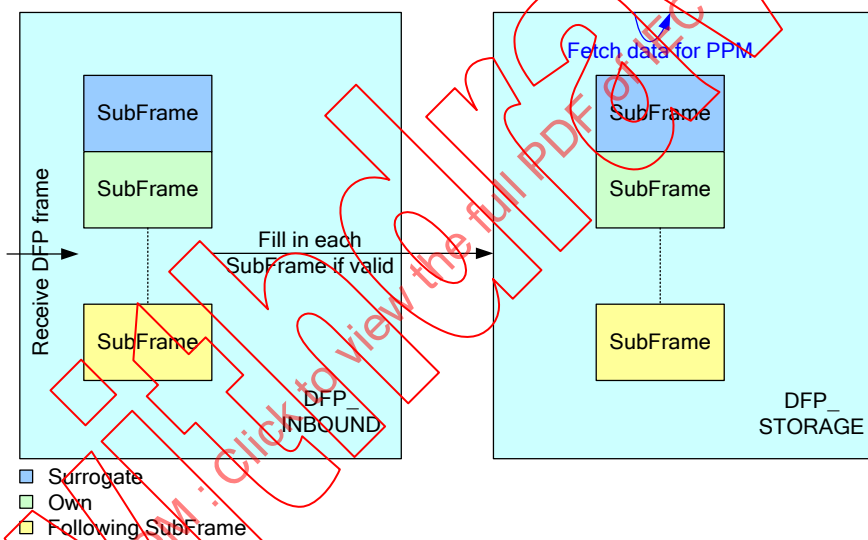


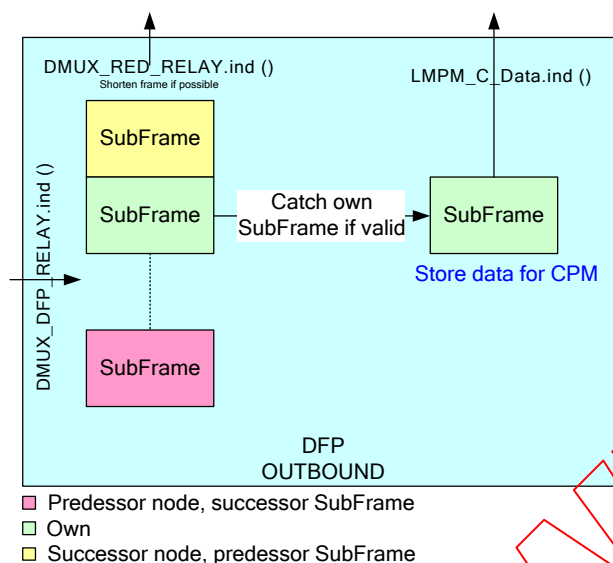
Figure 70 – Diagramme de DFP_RELAY



Légende

Anglais	Français
Fetch data for PPM	Extraction des données pour PPM
SubFrame	Sous-Trame
Receive DFP frame	Réception de la trame DFP
Fill in each SubFrame if valid	Remplir chaque Sous-Trame si valide
Surrogate	Substitut
Own	Propre
Following SubFrame	Sous-Trame suivante
SubFrame	Sous-Trame

Figure 71 – Diagramme de DFP_INBOUND et de DFP_STORAGE



Légende

Anglais	Français
SubFrame	Sous-Trame
Catch own SubFrame if Valid	Attraper la propre Sous-Trame si valide
Store data for CPM	Stockage des données pour CPM
Shorten frame if possible	Raccourcir la trame si possible
Predessor node, successor SubFrame	Nœud du prédecesseur, Sous-Trame du successeur
Own	Propre
Successor node, predessor SubFrame	Nœud du successeur, Sous-Trame du prédecesseur

Figure 72 – Diagramme de DFP_OUTBOUND

4.12.4.4.2 Machine de Protocole DFP

4.12.4.4.2.1 Définitions de primitives

4.12.4.4.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole DFP (DFP_RELAY) sont décrites dans la définition de service et présentées dans le Tableau 307.

Tableau 307 – Primitives distantes émises ou reçues par DFP_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_DFP_RELAY.ind	DEMUX	DFP_RELAY	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
DFP_INBOUND.ind	DFP_RELAY	DFP_INBOUND	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—

Primitive	Source	Destination	Paramètres associés	Fonctions
DFP_OUTBOUND.ind	DFP_RELAY	DFP_OUTBOUND	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
DMUX_RED_RELAY.ind	DFP_RELAY	RED_RELAY	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—

4.12.4.4.2.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DFP_RELAY sont décrites dans la définition de service et présentées dans le Tableau 308.

Tableau 308 – Primitives locales émises ou reçues par DFP_RELAY

Primitive	Source	Destination	Paramètres associés	Fonctions
SyncStateChange_ind	PTCP	DFP_RELAY	state {SYNC, ASYNC}	—

4.12.4.4.2.2 Schéma de transition d'états

Le schéma de transition d'états du DFP_RELAY est présenté à la Figure 73.

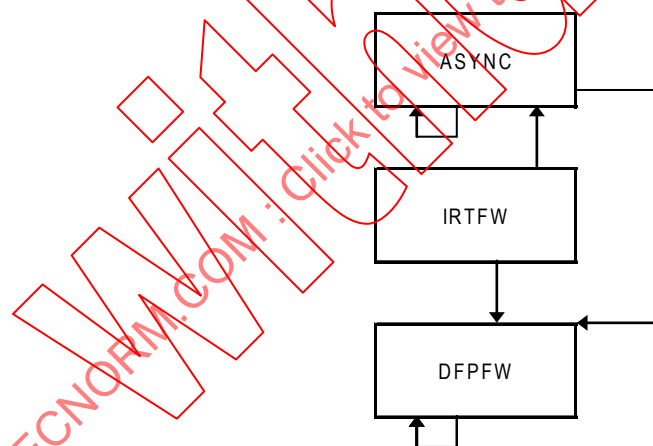


Figure 73 – Schéma de transition d'états de DFP_RELAY

Etats du DFP_RELAY

ASYNC

Le nœud doit être synchronisé, soit en local en tant que maître de synchronisation ou à distance en tant qu'esclave de synchronisation pour le transfert DFP basé sur le transfert RED

DFPFW

Transfert DFP actif

4.12.4.4.2.3 Description du diagramme d'états

Ce diagramme d'états reçoit les trames DFP possibles du DEMUX et les transfère au diagramme d'états attribué.

Les trames RT_CLASS_3 inconnues sont transférées au RED_RELAY.

4.12.4.4.2.4 Table d'états DFP_RELAY

Le DFP_RELAY vérifie la trame reçue (qui est "Reçue en RED" en raison du DEMUX) par rapport à la table d'états présentée au Tableau 309.

Tableau 309 – Table d'états DFP_RELAY

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	ASYNCR	SyncStateChange_ind () /SYNC => ignore	DFPFW
2	ASYNCR	SyncStateChange_ind () /!SYNC => ignore	ASYNCR
3	DFPFW	SyncStateChange_ind () /SYNC => ignore	DFPFW
4	DFPFW	SyncStateChange_ind () /!SYNC => ignore	ASYNCR
5	DFPFW	DMUX_DFP_RELAY.ind () /DFPGuard () == OUTBOUND => DFP_OUTBOUND.ind ()	DFPFW
6	DFPFW	DMUX_DFP_RELAY.ind () /DFPGuard () == INBOUND => DFP_INBOUND.ind ()	DFPFW
7	DFPFW	DMUX_DFP_RELAY.ind () /DFPGuard () == RED_RELAY => DMUX_RED_RELAY.ind ()	DFPFW

4.12.4.4.2.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 310 contient les fonctions, macros, temporisateurs et variables utilisés par le DFP_RELAY ainsi que leurs arguments et leurs descriptions.

Tableau 310 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY

Nom	Type	Fonction/Signification
DFPGuard	Fonction	Cette fonction vérifie la trame reçue par rapport aux informations locales.

4.12.4.4.2.6 DFPGuard

Le DFPGuard vérifie la trame reçue par rapport au Tableau 311.

Tableau 311 – Table de vérité pour DFPGuard

Entrée			Sortie		
Local			Local		
FrameID ^a	Sortant	Entrant	RED_RELAY	SORTANT	ENTRANT
FALSE	—	—	TRUE	—	—
TRUE	—	—	TRUE	—	—
TRUE	TRUE	FALSE	FALSE	TRUE	FALSE
TRUE	FALSE	TRUE	FALSE	FALSE	TRUE
TRUE	FALSE	FALSE	TRUE	FALSE	FALSE
TRUE	TRUE ^b	TRUE ^b	—	—	—

^a Vérifié par rapport à PDIRSubframeData.
^b Evité par des vérifications locales.

4.12.4.4.3 Machine de Protocole d'Entrée de DFP

4.12.4.4.3.1 Définitions de primitives

4.12.4.4.3.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de DFP (DFP_RELAY_INBOUND) sont décrites dans la définition de service et présentées dans le Tableau 312.

Tableau 312 – Primitives distantes émises ou reçues par DFP_RELAY_INBOUND

Primitive	Source	Destination	Paramètres associés	Fonctions
DFP_INBOUND.ind	DFP_RELAY	DFP_RELAY_INBOUND	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—

4.12.4.4.3.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DFP_RELAY_INBOUND sont décrites dans la définition de service et présentées dans le Tableau 313.

Tableau 313 – Primitives locales émises ou reçues par DFP_RELAY_INBOUND

Primitive	Source	Destination	Paramètres associés	Fonctions
DFP_Activate_req	CMSU	DFP_RELAY_INBOUND	CREP	—
DFP_Close_req	CMSU	DFP_RELAY_INBOUND	CREP	—
DFP_Activate_cnf (+)	DFP_RELAY_INBOUND	CMSU	CREP	—
DFP_Close_cnf (+)	DFP_RELAY_INBOUND	CMSU	CREP	—
StoreSubframe_ind	DFP_RELAY_INBOUND	DFP_RELAY_IN_STORAGE	CREP, Subframe	Déclenche DFP STORAGE pour une prise en charge ultérieure de la sous-trame.

4.12.4.4.3.2 Schéma de transition d'états

Le schéma de transition d'états du DFP_RELAY_INBOUND est présenté à la Figure 74.

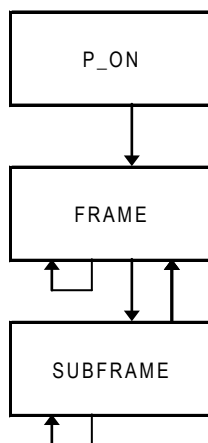


Figure 74 – Schéma de transition d'états de DFP_RELAY_INBOUND

Etats du DFP_RELAY_INBOUND

P_ON	Initialisation des données
FRAME	Vérification de la trame DFP
SUBFRAME	Vérification de la sous-trame DFP

4.12.4.4.3.3 Description du diagramme d'états

Ce diagramme d'états reçoit les trames DFP D'ENTREE, identifiées par leur FrameID, les compare à la définition de protocole prenant en charge les deux modes DistributedWatchDog actif et passif.

Pour MRPD, il existe deux présentations de DFP, identifiées par le FrameID.

4.12.4.4.3.4 Table d'états DFP_RELAY_INBOUND

Le DFP_RELAY_INBOUND doit garantir que la réception des trames d'un DFP est effectuée de façon correcte et la table d'états est présentée au Tableau 314.

Tableau 314 – Table d'états DFP_RELAY_INBOUND

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P_ON	DFP_Activate_req () => DFP_Activate_cnf (+)	FRAME
2	FRAME	DFP_INBOUND.ind () /InboundGuard () == FrameStructureValid => DFP_Check_SubFrame ()	SUBFRAME
3	FRAME	DFP_INBOUND.ind () /InboundGuard () == INVALID => ignore	FRAME

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
4	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == SubFrameStructureValid, SubFrameValid => StoreSubFrame_ind () DFP_Check_SubFrame ()	SUBFRAME
5	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == SubFrameStructureValid, SubFrameValid, EndOfSubFrame => ignore	FRAME
6	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == SubFrameStructureValid, SubFrameDataInvalid => DFP_Check_SubFrame ()	FRAME
7	SUBFRAME	DFP_Check_SubFrame() /InboundGuard () == INVALID => ignore	FRAME
8	ANY	DFP_Close_req () => DFP_Close_cnf (+)	P_ON

4.12.4.4.3.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 315 contient les fonctions, macros, temporisateurs et variables utilisés par le DFP_RELAY_INBOUND ainsi que leurs arguments et leurs descriptions.

Tableau 315 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY_INBOUND

Nom	Type	Fonction/Signification
InboundGuard	Fonction	Cette fonction vérifie la trame reçue par rapport aux informations locales.
DFP_Check_SubFrame	Fonction	Déclencheur pour l'analyse ultérieure des trames. Les sous-trames sont vérifiées selon leur séquence dans la trame; chaque appel de la fonction vérifie une sous-trame particulière.

4.12.4.4.3.6 InboundGuard

L'InboundGuard vérifie la trame reçue par rapport au Tableau 316, au Tableau 317, au Tableau 318 et au Tableau 319.

Tableau 316 – Table de vérité pour InboundGuard – vérification de trame

Entrée	Sortie
En-tête Subframe. SFCRC16	Vérification de trame
Non valide	Non valide
Valide	Valide

Tableau 317 – Table de vérité pour InboundGuard – vérification de sous-trame

Entrée			Sortie
Subframe. Position ^a	Subframe. Length ^b	Subframe. SFCRC16 ^c	Vérification de sous- trame
Non trouvé	—	—	Non valide
Trouvé	FALSE	—	Non valide
Trouvé	TRUE	Non valide	Non valide
Trouvé	TRUE	Valide	Valide

^a Vérifie si la prochaine sous-trame prévue est reçue
^b Vérifie si la sous-trame possède la longueur prévue
^c Vérifie si SFCRC16 est valide

Tableau 318 – Table de vérité pour InboundGuard – vérification des données de la sous-trame

Entrée				Sortie
RestartFor- DistributedWdTime ^a	Distributed- WatchDog- Factor ^b	Subframe. DataStatus. Valid ^c	Subframe. SFCycleCounter ^d	Vérification des données de sous- trame
OFF	OFF	—	—	Valide
En cours d'exécution	—	—	—	Non valide
Expiré / OFF	ON	Non valide	—	Non valide
Expiré / OFF	ON	Valide	Ancien	Non valide
Expiré / OFF	ON	Valide	Nouveau	Valide

^a Un RestartForDistributedWdTime en cours d'exécution arrête le stockage de la sous-trame.
^b La vérification des données de la sous-trame est effectuée par l'extrémité si le chien de garde distribué est désactivé.
^c Les données sont uniquement stockées si elles sont valides.
^d Seules les sous-trames les plus récentes sont stockées.

Tableau 319 – Table de vérité pour InboundGuard – vérification complète

Entrée			Sortie		
Vérification de trame	Vérification de sous- trame	Vérification des données de sous- trame	Frame- Structure- Valid	Subframe- Structure- Valid	Subframe- DataValid
Non valide	—	—	Non valide	—	—
Valide	Non valide	—	Valide	Non valide	—
Valide	Valide	Non valide	Valide	Valide	Non valide
Valide	Valide	Valide	Valide	Valide	Valide

4.12.4.4.4 Machine de Protocole de Stockage à l'Entrée de DFP

4.12.4.4.4.1 Généralités

Du point de vue de la cohérence, il n'existe aucune différence entre les règles pour une trame et celles pour une sous-trame. Le système d'acheminement doit garantir qu'une trame est

transportée avec la cohérence de trame et qu'une sous-trame est transportée avec la cohérence de sous-trame.

4.12.4.4.2 Définitions de primitives

4.12.4.4.2.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Stockage à l'Entrée de DFP (DFP_RELAY_IN_STORAGE) sont décrites dans la définition de service et présentées dans le Tableau 320.

Tableau 320 – Primitives distantes émises ou reçues par DFP_RELAY_IN_STORAGE

Primitive	Source	Destination	Paramètres associés	Fonctions
AlarmNotification.ind	DFP_RELAY_IN_STORAGE	FAL	AREP, API, AlarmPriority, AlarmType, SlotNumber, SubslotNumber, AlarmSpecifier, ModuleIdentifier, SubmoduleIdentifier, AlarmItem	Informe l'application qu'un défaut de concaténation a lieu.

4.12.4.4.2.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DFP_RELAY_IN_STORAGE sont décrites dans la définition de service et présentées dans le Tableau 321.

Tableau 321 – Primitives locales émises ou reçues par DFP_RELAY_IN_STORAGE

Primitive	Source	Destination	Paramètres associés	Fonctions
DFP_Activate_req	CMSU	DFP_RELAY_IN_STORAGE	CREP	—
DFP_Close_req	CMSU	DFP_RELAY_IN_STORAGE	CREP	—
DFP_Activate_cnf(+)	DFP_RELAY_IN_STORAGE	CMSU	CREP	—
DFP_Close_cnf	DFP_RELAY_IN_STORAGE	CMSU	CREP	—
DFP_Error_ind	DFP_RELAY_IN_STORAGE	CMSU CTLSU	CREP, ERRCLS, ERRCODE	—
StoreSubFrame_ind	DFP_RELAY_INBOUND	DFP_RELAY_IN_STORAGE	CREP, Subframe	Stocke une sous-trame reçue sur son stockage

4.12.4.4.3 Schéma de transition d'états

Le schéma de transition d'états du DFP_RELAY_IN_STORAGE est présenté à la Figure 75.

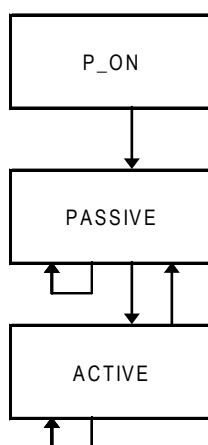


Figure 75 – Schéma de transition d'états de DFP_RELAY_IN_STORAGE

Etats du DFP_RELAY_IN_STORAGE

P_ON	Initialisation des données
PASSIVE	Attend la première réception ou la temporisation de RestartForDistributedWDTime.
ACTIVE	Vérifie la réception d'une sous-trame DFP avec un chien de garde distribué s'il est activé. De plus, vérifie la concaténation des sous-trames par rapport aux contraintes de temps de PPM associées et génération d'indications si nécessaire.

4.12.4.4.4 Description du diagramme d'états de DFP_RELAY_IN_STORAGE

Ce diagramme d'états reçoit les sous-trames et les stocke pour une utilisation ultérieure. La concaténation des sous-trames à la frame générée en local est effectuée parallèlement au stockage des données.

Si un défaut est détecté, alors la transmission de la trame doit être arrêtée prématurément avec une erreur TX correspondante. Dans ce cas, les dernières données de la sous-trame stockées sont encore valides et utilisables.

4.12.4.4.5 Table d'états DFP_RELAY_IN_STORAGE

Le DFP_RELAY_IN_STORAGE doit contrôler la réception des trames DFP. La table d'état correspondante est présentée au Tableau 322.

Tableau 322 – Table d'états DFP_RELAY_IN_STORAGE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P_ON	DFP_Activate_req () => Create list of sub frames in surrogate state List of successors If activated List of distributed watchdogs DFP_Activate_cnf(+)	PASSIVE
2	PASSIVE	StoreSubFrame_ind () /RestartForDistributedWDTime (Position) != Running => Update content of the stored sub frame (Position) If activated StartTimer (Distributed WDT (Position))	ACTIVE

#	Etat en cours	Événement /Condition =>Action	Etat suivant
3	PASSIVE	StoreSubFrame_ind () /RestartForDistributedWDTTime (Position) == Running =>	PASSIVE
4	PASSIVE	TimerExpired (RestartForDistributedWDTTime (Position)) => ignore	PASSIVE
5	PASSIVE	TimerExpired (Distributed WDT (Position)) => ignore	PASSIVE
6	ACTIVE	StoreSubFrame_ind () => Update content of the stored sub frame (Position) If activated StartTimer (Distributed WDT (Position))	ACTIVE
7	ACTIVE	TimerExpired (Distributed WDT (Position)) => Set stored sub frame in surrogate state (Position) StartTimer (RestartForDistributedWDTTime (Position))	PASSIVE
8	ACTIVE	TimerExpired (RestartForDistributedWDTTime (Position)) => ignore	ACTIVE
9	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && Sub frames updated in this Phase && !Appear sent => ignore	ACTIVE
10	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && !Sub frames updated in this Phase && Appear sent => ignore	ACTIVE
11	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && Sub frames updated in this Phase && Appear sent => Appear sent := FALSE Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= Dynamic Frame Packing problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Disappears ChannelErrorType := "Dynamic frame packing function mismatch" ExtChannelErrorType := "Frame late error" AlarmNotification.ind ()	ACTIVE
12	ACTIVE	TickEvent () /Phase and FrameSendOffset of the associated PPM && !Sub frames updated in this Phase && !Appear sent => Appear sent := TRUE Alarm Priority := ALARM_HIGH or ALARM_LOW Alarm Type:= Dynamic Frame Packing problem notification User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelErrorType := "Dynamic frame packing function mismatch" ExtChannelErrorType := "Frame late error" AlarmNotification.ind ()	ACTIVE

#	Etat en cours	Événement /Condition =>Action	Etat suivant
13	ANY	DFP_Close_req () => Remove list of sub frames Remove stored diagnosis from Diagnosis ASE DFP_Close_cnf ()	P_ON

4.12.4.4.4.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 323 contient les fonctions, macros, temporisateurs et variables utilisés par le DFP_RELAY_IN_STORAGE ainsi que leurs arguments et leurs descriptions.

Tableau 323 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY_IN_STORAGE

Nom	Type	Fonction/Signification
Set stored sub frame in surrogate state	Fonction	Cette fonction locale doit mettre le C_SDU de la sous-trame à zéro et SFDataStatus.Ignore doit être réglé sur TRUE: il convient que les autres fanions de DataStatus et SFCycleCounter soient zéro.
StartTimer	Fonction	Cette fonction locale est utilisée pour lancer ou relancer un temporisateur.
TickEvent	Fonction	Événement généré en local à partir d'un temporisateur. Le temporisateur doit être contrôlé par le maître ou l'esclave PTCP en cas de synchronisation ou d'exécution libre sans synchronisation. Paramètres: CycleCounter, CycleOffset
TimerExpired	Fonction	Cette fonction locale est émise si un temporisateur expire.
Update content of the stored sub frame	Fonction	Cette fonction locale a mis à jour le C_SDU, DataStatus et le SFCycleCounter de la sous-trame
Phase and FrameSendOffset of the associated PPM	Macro	Cette macro locale utilise les informations temporelles du PPM associé pour une vérification par rapport aux informations de TickEvent.
Sub frames updated in this Phase	Macro	Cette macro locale vérifie si la sous-trame a été reçue en temps afin d'être concaténée à la trame injectée en local.
Distributed Watchdog Time	Temporisateur	Ce temporisateur est utilisé pour contrôler la réception correcte de données valides s'il est activé.
RestartForDistributedWDTime	Temporisateur	Ce temporisateur est utilisé pour protéger une sous-trame subrogée pour la détection d'un défaut par l'appareil de commande E/S. Il n'est actif que si le chien de garde distribué est requis par l'ingénierie.
Appear sent	Variable	Cette variable locale est utilisée pour stocker les informations stipulant si une demande d'alarme a ou non déjà été donnée.

4.12.4.4.5 Machine de Protocole à la Sortie de DFP

4.12.4.4.5.1 Définitions de primitives

4.12.4.4.5.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole à la Sortie de DFP (DFP_RELAY_OUTBOUND) sont décrites dans la définition de service et présentées dans le Tableau 324.

Tableau 324 – Primitives distantes émises ou reçues par DFP_RELAY_OUTBOUND

Primitive	Source	Destination	Paramètres associés	Fonctions
DFP_OUTBOUND.ind	DFP_RELAY_OUTBOUND	DFP_RELAY_OUTBOUND	Port, Tstamp, DA, SA, C_SDU, ReceivedInRED	—
LMPM_C_Data.ind	DFP_RELAY_OUTBOUND	—	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status, ReceivedInRED	Acheminer la propre sous-trame à l'interface locale pour une prise en charge ultérieure dans le CPM associé.
DMUX_RED_RELAY.ind	DFP_RELAY_OUTBOUND	RED_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—

4.12.4.4.5.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DFP_RELAY_OUTBOUND sont décrites dans la définition de service et présentées dans le Tableau 325.

Tableau 325 – Primitives locales émises ou reçues par DFP_RELAY_OUTBOUND

Primitive	Source	Destination	Paramètres associés	Fonctions
DFP_Activate_cnf (+)	DFP_RELAY_OUTBOUND	CMSU	CREP	—
DFP_Close_cnf (+)	DFP_RELAY_OUTBOUND	CMSU	CREP	—
DFP_Activate_req	CMSU	DFP_RELAY_OUTBOUND	CREP	—
DFP_Close_req	CMSU	DFP_RELAY_OUTBOUND	CREP	—

4.12.4.4.5.2 Schéma de transition d'états

Le schéma de transition d'états du DFP_RELAY_OUTBOUND est présenté à la Figure 76.

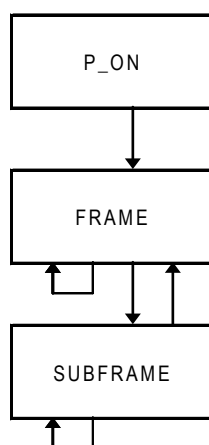


Figure 76 – Schéma de transition d'états de DFP_RELAY_OUTBOUND

Etats du DFP_RELAY_OUTBOUND

P_ON	Initialisation des données
FRAME	Vérifie si la trame a une structure valide et doit être raccourcie.
SUBFRAME	Recherche la propre sous-trame et raccourcit la trame si la structure est valide.

4.12.4.4.5.3 Description du diagramme d'états de DFP_RELAY_OUTBOUND

Ce diagramme d'états reçoit une trame identifiée par le FrameID, vérifie si le raccourcissement est possible. Si cela s'avère possible, la trame sera raccourcie avant la propre sous-trame et sera transférée. La propre sous-trame sera acheminée au CPM. Si cela n'est pas possible, la trame est transférée au RED_RELAY.

Le DFP_RELAY_OUTBOUND doit garantir que la partie locale des trames DFP est donnée à la LMPM et que, si cela est possible, la trame raccourcie est donnée au RED_RELAY.

Il convient que la trame soit raccourcie si SFIOCRProperties.SFCRC16 est mis à un et si la sous-trame attribuée est reçue localement après vérification de sa SFCRC16.

Le raccourcissement doit être effectué en remplaçant la sous-trame reçue localement par un SFEndDelimiter, si nécessaire Padding et un APDU_Status (CycleCounter:=valeur locale réelle, TransferStatus:=0, DataStatus.Ignore:=1, DataStatus.State:=1, DataStatus.DataValid:=1, DataStatus.StationProblemIndicator:=1, DataStatus.ProviderState:=1).

Une sous-trame attribuée localement doit être reçue après vérification du FCS de la trame d'acheminement si SFIOCRProperties.SFCRC16 est mis à zéro.

4.12.4.4.5.4 Table d'états DFP_RELAY_OUTBOUND

La table d'états DFP_OUTBOUND est présentée au Tableau 326.

Tableau 326 – Table d'états DFP_RELAY_OUTBOUND

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P_ON	DFP_Activate_req () => DFP_Activate_cnf (+)	FRAME
2	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == FrameStructureValid, FrameShortenImpossible => DMUX_RED_RELAY.ind ()	FRAME
3	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == FrameStructureValid, FrameShortenPossible && FirstSubframeInFrame == OWN => DMUX_RED_RELAY.ind ()	FRAME
4	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == FrameStructureValid, FrameShortenPossible && ! FirstSubframeInFrame == OWN => DFP_Check_SubFrame ()	SUBFRAME
5	FRAME	DFP_OUTBOUND.ind () /OutboundGuard () == INVALID => ignore NOTE La trame doit être raccourcie après la dernière sous-trame valide par le MUX avec l'erreur TX appropriée	FRAME
6	SUBFRAME	DFP_Check_SubFrame () /OutboundGuard () == SubFrameStructureValid && SFPosition.Position == OWN => N_SDU := ShortenFrame (N_SDU) LMPM_C_Data.ind (), DMUX_RED_RELAY.ind ()	SUBFRAME
7	SUBFRAME	DFP_Check_SubFrame () /OutboundGuard () == SubFrameStructureValid && SFPosition.Position != OWN => DFP_Check_SubFrame ()	SUBFRAME
8	SUBFRAME	DFP_Check_SubFrame () /OutboundGuard () == INVALID => ignore //NOTE La trame doit être raccourcie après la dernière sous-trame valide par le MUX avec l'erreur TX appropriée	FRAME
9	ANY	DFP_Close_req () => DFP_Close_cnf (+)	P_ON

4.12.4.4.5.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 327 contient les fonctions, macros, temporisateurs et variables utilisés par le DFP_RELAY_OUTBOUND ainsi que leurs arguments et leurs descriptions.

Tableau 327 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DFP_RELAY_OUTBOUND

Nom	Type	Fonction/Signification
DFP_Check_SubFrame	Fonction	Déclencheur pour l'analyse ultérieure des trames. Les sous-trames sont vérifiées selon leur séquence dans la trame; chaque appel de la fonction vérifie une sous-trame particulière.
OutboundGuard	Fonction	Cette fonction vérifie la trame reçue par rapport aux informations locales.
ShortenFrame	Fonction	Cette fonction locale raccourcit la N_SDU si la séquence de SFCRC16 valide n'est pas rompue.
FirstSubframeInFrame	Macro	Cette macro locale retourne les informations de la première sous-trame dans la trame. Ces informations sont requises pour éviter le raccourcissement d'une trame ne contenant qu'une seule sous-trame.
OWN	Variable	Cette variable locale contient les informations relatives à la propre sous-trame.

4.12.4.4.5.6 OutboundGuard

L'OutboundGuard vérifie la trame reçue par rapport au Tableau 328 et au Tableau 329.

Tableau 328 – Table de vérité pour l'OutboundGuard – vérification de la trame

Entrée			Sortie	
SFIOCRProperties. SFCRC16 ^a	En-tête Subframe. SFCRC16 ^b	Structure avant la propre sous- trame ^c	Structure de la trame	Raccourcissement de la trame
FALSE	—	—	Valide	Impossible
TRUE	Non valide	—	Non valide	—
TRUE	Valide	Non valide	Non valide	Impossible
TRUE	Valide	Valide	Valide	Possible

^a Le raccourcissement d'une trame n'est possible que si le SFCRC16 est valide. Par conséquent, les sous-trames ne sont vérifiées que si SFCRC16 doit exister.

^b Le SFCRC16 d'en-tête est vérifié avant que la vérification de sous-trame ne commence.

^c Le raccourcissement n'est possible que si une structure cohérente est trouvée avant la propre sous-trame.

Tableau 329 – Table de vérité pour l'OutboundGuard – vérification de la sous-trame

Entrée				Sortie
Structure de la trame	Subframe. Position ^a	Subframe. Length ^b	Subframe. SFCRC16	Structure de la sous-trame
Non valide	—	—	—	Non valide
Valide	Non valide	—	—	Non valide
Valide	Valide	Non valide	—	Non valide
Valide	Valide	Valide	Non valide	Non valide
Valide	Valide	Valide	Valide	Valide

Entrée				Sortie
Structure de la trame	Subframe. Position ^a	Subframe. Length ^b	Subframe. SFCRC16	Structure de la sous-trame
^a Vérification par rapport à la définition du protocole, facultatif par rapport à la structure de PDIRSubframeData.				
^b Vérification par rapport à la définition du protocole, facultative par rapport à la structure de PDIRSubframeData.				

4.13 Ponts virtuels

4.13.1 Vue d'ensemble

Les concepts de l'IEEE 802.1Q doivent être appliqués. Cette partie de la spécification définit l'utilisation de l'IEEE 802.1Q. Elle contient des prolongements de la phase RT_CLASS_3 relatifs à la transmission des trames. Les mécanismes d'acheminement présentés dans l'IEEE 802.1Q doivent être provisoirement désactivés dans la phase RT_CLASS_3. Dans ce cas, l'acheminement des trames RT_CLASS_3 doit être conforme aux attributs définis dans l'ASE approprié. Les valeurs des attributs définissent une table d'acheminement particulière utilisée.

Outre le comportement de l'IEEE 802.1Q, la spécification du protocole définit les machines de protocole suivantes exécutant des actions MAC particulières.

- Machine de protocole de gestionnaire de file d'attente
- Machine de protocole Multiplex MAC (MUX)
- Machine de protocole Demultiplex MAC (DEMUX)

4.13.2 QueueHandler

Le QueueHandler gère les files d'attente de priorité d'un port. Il offre des primitives de service pour mettre en file d'attente et retirer de la file d'attente les trames. Des informations complémentaires relatives au contenu d'une file d'attente sont disponibles.

4.13.3 Machines de Protocole de Service FAL

Il n'existe aucune Machine de Protocole de Service FAL (FSPM) définie pour ce Protocole.

4.13.4 Machines de Protocole de relation entre applications

Il n'existe aucune Machine de Protocole de relation entre applications (ARPM) définie pour ce Protocole.

4.13.5 Machines de Protocole de Mapping DLL

4.13.5.1 Machine de Protocole de l'Emetteur d'Accès au Réseau

4.13.5.1.1 Définitions de primitives

4.13.5.1.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par l'Emetteur d'Accès au Réseau (MUX) sont décrites dans la définition de service et présentées dans le Tableau 330.

Tableau 330 – Primitives distantes émises ou reçues par MUX

Primitive	Source	Destination	Paramètres associés	Fonctions
M_UNITDATA.req	MUX	MAC	frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, rif_information (optional), include_tag	—
M_UNITDATA.ind	MAC	MUX	time_stamp	—
LinkStateChange.ind	MAC	MUX	Port, Link_State	—
PortStateChange.ind	Virtual Bridge	MUX	Port, Port_State	—

4.13.5.1.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par MUX sont décrites dans la définition de service et présentées dans le Tableau 331.

Tableau 331 – Primitives locales émises ou reçues par MUX

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.13.5.1.2 Schéma de transition d'états

Le schéma de transition d'états du MUX est présenté à la Figure 77.

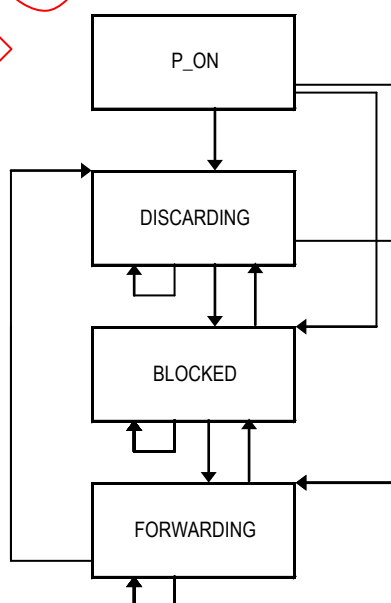


Figure 77 – Schéma de transition d'états de MUX

Etats du MUX

P-ON	Attend l'état de la Liaison et du Port du démarrage et passe à l'état approprié
DISCARDING	Toutes les trames sont rejetées dans cet état
BLOCKED	Seules les trames de commande du réseau ou RT_CLASS_3 sont transférées dans cet état
FORWARDING	Toutes les trames sont transférées

4.13.5.1.3 Description du diagramme d'états

Le MUX est chargé de mapper les services de LMPM, RED_RELAY et MAC_RELAY(IEEE 802.1D) à MAC conformément à l'IEEE 802.3.

De plus, le Gestionnaire de File d'Attente présenté à la Figure 79 est utilisé pour gérer les files d'attente de priorité.

4.13.5.1.4 Table d'états MUX

La table d'états MUX est présentée au Tableau 332.

Tableau 332 – Table d'états MUX

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P_ON	/StateChecker () == FORWARDING => Period := GREEN	FORWARDING
2	P_ON	/StateChecker () == BLOCKED => Period := GREEN	BLOCKED
3	P_ON	/StateChecker () == DISCARDING => Period := GREEN	DISCARDING
4	P_ON	M_UNITDATA.cnf () => ignore	P_ON
5	DISCARDING	MUX_Schedule_req (CycleCounter, Tx_Period) => ignore	DISCARDING
6	DISCARDING	QueuesNotEmpty () => GET_FRAME_FROM_QUEUE(NwC_Hi, NwC_Lo, PRI07, RED, PRI06, ..., PRIOR2-1) DISCARD_FRAME()	DISCARDING
7	DISCARDING	M_UNITDATA.cnf () => ignore	DISCARDING
8	BLOCKED	QueuesNotEmpty () /Period == GREEN && FrameSizeFits (NwC_Hi, NwC_Lo) => GET_FRAME_FROM_QUEUE(NwC_Hi, NwC_Lo) M_UNITDATA.req ()	BLOCKED
9	BLOCKED	QueuesNotEmpty () /Period == GREEN && !FrameSizeFits (NwC_Hi, NwC_Lo) =>	BLOCKED

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
		ignore	
10	BLOCKED	M_UNITDATA.cnf () => ignore	BLOCKED
11	BLOCKED / FORWARDING	MUX_Schedule_req (CycleCounter, Tx_Period) => Period := Tx_Period	BLOCKED / FORWARDING
12	BLOCKED / FORWARDING	QueuesNotEmpty () /Period == RED => GET_FRAME_FROM_QUEUE(RED) M_UNITDATA.req ()	BLOCKED / FORWARDING
13	FORWARDING	QueuesNotEmpty () /Period == GREEN && FrameSizeFits (NwC_Hi, NwC_Lo, PRIOR7, PRIOR6, ..., PRIOR2-1) => GET_FRAME_FROM_QUEUE(NwC_Hi, NwC_Lo, PRIOR7, PRIOR6, ..., PRIOR2-1) M_UNITDATA.req ()	FORWARDING
14	FORWARDING	QueuesNotEmpty () /Period == GREEN && !FrameSizeFits (NwC_Hi, NwC_Lo, PRIOR7, PRIOR6, ..., PRIOR2-1) => ignore	FORWARDING
15	FORWARDING	M_UNITDATA.cnf () => ignore	FORWARDING
16	BLOCKED / FORWARDING	QueuesNotEmpty () /Period != RED && FrameEgressFilter () => GET_FRAME_FROM_QUEUE() DISCARD_FRAME()	BLOCKED / FORWARDING
17	ANY	LinkStateChange () /StateChecker () == DISCARDING => Period := GREEN	DISCARDING
18	ANY	PortStateChange () /StateChecker () == DISCARDING => Period := GREEN	DISCARDING
19	ANY	LinkStateChange () /StateChecker () == BLOCKED => ignore	BLOCKED
20	ANY	PortStateChange () /StateChecker () == BLOCKED => ignore	BLOCKED
21	ANY	LinkStateChange () /StateChecker () == FORWARDING => ignore	FORWARDING
22	ANY	PortStateChange () /StateChecker () == FORWARDING => ignore	FORWARDING

4.13.5.1.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 333 contient les fonctions, macros, temporisateurs et variables utilisés par le MUX ainsi que leurs arguments et leurs descriptions.

Tableau 333 – Fonctions, Macros, Temporisateurs et Variables utilisés par le MUX

Nom	Type	Fonction/Signification
FrameEgressFilter	Fonction	Cette fonction locale vérifie s'il convient que la trame soit transmise ou supprimée conformément au filtrage à la sortie de l'IEEE 802.1D
FrameSizeFits	Fonction	Cette fonction locale vérifie si la trame trouvée est adaptée à la période restante jusqu'au StartOfRED suivant. Elle n'a été recherchée que dans les files d'attente (...) de la priorité la plus haute à la priorité la plus basse. La première trame non adaptée arrête la recherche et FALSE est retourné. La vérification comprend les exigences de la table de vérité correspondante.
QueuesNotEmpty	Fonction	Cet événement local est émis si une entrée existe dans une file d'attente. Une entrée existe après qu'une réception de trame a été détectée et la file d'attente de transmission de la commutation CT ou S&F a été attribuée
StateChecker	Fonction	Cette fonction locale vérifie si Link_State et Port_State correspondent à l'état suivant possible et elle retourne le résultat
DISCARD_FRAME	Macro	Rejette la trame sélectionnée
GET_FRAME_FROM_QUEUE	Macro	Délivre une trame des files d'attente (...). Ce retrait des files d'attente fonctionne toujours de la priorité la plus haute à la priorité la plus basse des files d'attente.
Period	Variable	Cette variable locale stocke la période réelle pour une utilisation ultérieure

4.13.5.1.6 Tables de vérité

4.13.5.1.6.1 Table de vérité des règles de FrameSizeFits

Le Tableau 334 contient la table de vérité utilisée par le MUX.

Tableau 334 – Table de vérité pour FrameSizeFits

Entrée						Sortie
Period	Distance supérieure à MaxFrameSize	Distance supérieure à YellowTime	Protocoles parfaitement connus	Autres protocoles	Dimension d'image	FrameSizeFits
VERTE	True	True	CT / S&F	CT / S&F	—	True
VERTE	False	True	CT / S&F	—	—	True
VERTE	False	True	—	S&F	Fits	True
VERTE	False	True	—	S&F	Trop gros	False
VERTE	False	False	S&F	S&F	Fits	True
VERTE	False	False	S&F	S&F	Trop gros	False
VERTE	False	False	CT	CT	—	False
ROUGE	—	—	—	—	—	—

4.13.5.1.6.2 Table de vérité pour StateChecker

Le Tableau 335 contient la table de vérité utilisée par le MUX.

Tableau 335 – Table de vérité pour StateChecker

Entrée		Sortie
Link_State	Port_State	StateChecker
DOWN	DISCARDING	DISCARDING
DOWN	BLOCKED	DISCARDING
DOWN	FORWARDING	DISCARDING
UP	DISCARDING	DISCARDING
UP	BLOCKED	BLOCKED
UP	FORWARDING	FORWARDING

4.13.5.2 Machine de Protocole du Destinataire de l'Accès au Réseau

4.13.5.2.1 Définitions de primitives

4.13.5.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par le Destinataire de l'Accès au Réseau (DEMUX) sont décrites dans la définition de service et présentées dans le Tableau 336.

Tableau 336 – Primitives distantes émises ou reçues par DEMUX

Primitive	Source	Destination	Paramètres associés	Fonctions
M_UNITDATA.ind	MAC	DEMUX	Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier	—
DMUX_DEFRAG.ind	DEMUX	DEFRAG	Port, Tstamp, DA, SA, VLAN, N_SDU	—
DMUX_DFP_RELAY.ind	DEMUX	DFP_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—
DMUX_MAC_RELAY.ind	DEMUX		Port, Tstamp, DA, SA, N_SDU	—
DMUX_P_Data.ind	DEMUX	LMPM	Port, Tstamp, DA, SA, N_SDU	—

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_RED_RELAY.ind	DEMUX	RED_RELAY	Port, Tstamp, DA, SA, N_SDU, ReceivedInRED	—

4.13.5.2.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par DEMUX sont décrites dans la définition de service et présentées dans le Tableau 337.

Tableau 337 – Primitives locales émises ou reçues par DEMUX

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_Schedule_req	SCHEDULER	DEMUX	Rx_Period	—
DMUX_Schedule_cnf	DEMUX	SCHEDULER	—	—

4.13.5.2.2 Schéma de transition d'états

Le schéma de transition d'états du DEMUX est présenté à la Figure 78.

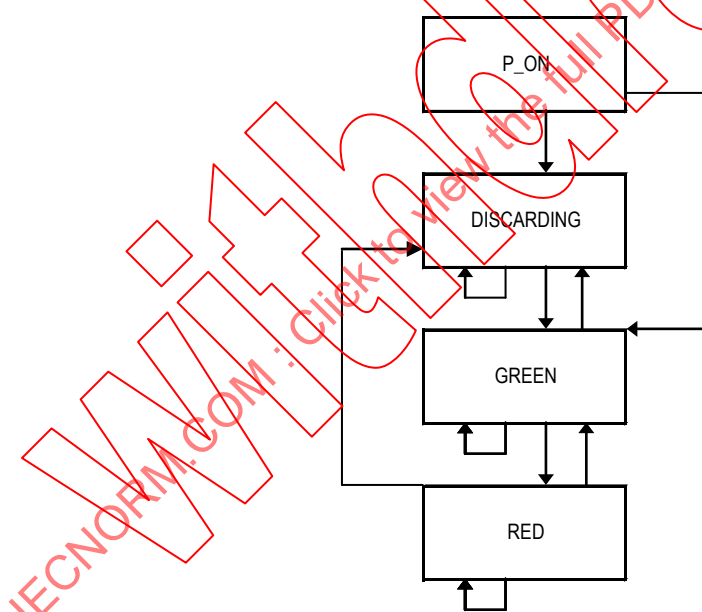


Figure 78 – Schéma de transition d'états de DEMUX

Etats du DEMUX

P_ON	Mise sous tension
DISCARDING	Rejet de l'état du port
GREEN	période verte
RED	période red

4.13.5.2.3 Description du diagramme d'états

Le DEMUX est chargé du mapping des services MAC conformément à l'IEEE 802.3 avec LMPM, RED_RELAY et MAC_RELAY (IEEE 802.1D).

La génération du fanion ReceivedInRED doit être effectuée en vérifiant le TimeStamp de réception effectué par le SHIM. Si la valeur du ReceiveTimeStamp est supérieure ou égale à RedOrangePeriodBegin et inférieure ou égale au point dans le temps GreenPeriodBegin, alors le ReceivedInRED doit être mis sur TRUE, sinon sur FALSE.

4.13.5.2.4 Table d'états DEMUX

La table d'états DEMUX apparaît au Tableau 338.

Tableau 338 – Table d'états DEMUX

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
1	P_ON	/Port_State != Discarding => ignore	GREEN
2	P_ON	/Port_State == Discarding => ignore	DISCARDING
3	DISCARDING	DMUX_Schedule_req () => DMUX_Schedule_cnf ()	DISCARDING
4	DISCARDING	M_UNITDATA.ind () => ignore	DISCARDING
5	DISCARDING	PortStateChange () /Port_State == Discarding => ignore	DISCARDING
6	DISCARDING	PortStateChange () /Port_State != Discarding => ignore	GREEN
7	GREEN	PortStateChange () /Port_State != Discarding => ignore	GREEN
8	GREEN	PortStateChange () /Port_State == Discarding => ignore	DISCARDING
9	GREEN	DMUX_Schedule_req () /Rx_Period == GREEN => DMUX_Schedule_cnf ()	GREEN
10	GREEN	DMUX_Schedule_req () /Rx_Period == RED => DMUX_Schedule_cnf ()	RED
11	GREEN	M_UNITDATA.ind () /FrameIngressFilter // Checking the ingress filter is always the first check! => ignore	GREEN

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
12	GREEN	M_UNITDATA.ind () /FRAGMENT => DMUX_DEFRAG.ind ()	GREEN
13	GREEN	M_UNITDATA.ind () /OTHER OR RED OR DFP => DMUX_MAC_RELAY.ind ()	GREEN
14	GREEN	M_UNITDATA.ind () /CTRL => Get_Receive_TimeStamp () DMUX_P_Data.ind ()	GREEN
15	RED	PortStateChange () /Port_State != Discarding => ignore	RED
16	RED	PortStateChange () /Port_State == Discarding => ignore	DISCARDING
17	RED	DMUX_Schedule_req () /Rx_Period == GREEN => DMUX_Schedule_cnf ()	GREEN
18	RED	DMUX_Schedule_req () /Rx_Period == RED => DMUX_Schedule_cnf ()	RED
19	RED	M_UNITDATA.ind () /RED => ReceivedInRED := TRUE DMUX_RED_RELAY.ind ()	RED
20	RED	M_UNITDATA.ind () /DFP => ReceivedInRED := TRUE DMUX_DFP_RELAY.ind ()	RED
21	RED	M_UNITDATA.ind () /FRAGMENT => DMUX_DEFRAG.ind ()	RED
22	RED	M_UNITDATA.ind () /OTHER => DMUX_MAC_RELAY.ind ()	RED
23	RED	M_UNITDATA.ind () /CTRL => Get_Receive_TimeStamp () DMUX_P_Data.ind ()	RED

4.13.5.2.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 339 contient les fonctions, macros, temporisateurs et variables utilisés par le DEMUX ainsi que leurs arguments et leurs descriptions.

Tableau 339 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DEMUX

Nom	Type	Fonction/Signification
Get_Receive_TimeStamp	Fonction	Récupère la trame d'horodatage de réception, le Timestamp Shim de cette trame
CTRL	Macro	Contrôle/gestion du réseau par exemple, trames de synchronisation PTCP sans "attribut RT_CLASS_3", LLDP, mesure de retard de ligne PTCP, annonce PTCP, redondance de support, ...
DFP	Macro	Trames RT_CLASS_3 frames avec "attribut DFP" EtherType == 0x8892 FrameID == 0x0100...0x0FFF
FRAGMENT	Macro	Trame utilisant le protocole de fragmentation. EtherType == 0x8892 FrameID == Fragmentation
FrameIngressFilter	Macro	Vérifie si cette trame est filtrée par une règle d'entrée.
OTHER	Macro	Toute autre trame.
RED	Macro	RT_CLASS_3 frames EtherType == 0x8892 FrameID == 0x0100 ...0x0FFF Cas particulier: trames de synchronisation PTCP avec "RT_CLASS_3 attribute" EtherType == 0x8892 FrameID == 0x0080
PortStateChange	Fonction	Achemine les informations relatives à l'état du port

4.14 Suite IP

4.14.1 Vue d'ensemble

L'utilisation des normes RFC 768 (UDP), RFC 791 (IP), RFC 792 (ICMP), RFC 826 (ARP), RFC 2236 (IP Multicasting) et RFC 2474 (IP Extensions) est définie pour la présente spécification. Elle contient les définitions des ports UDP et des adresses de multidiffusion IP, ainsi que l'utilisation des champs d'en-tête IP de RT_CLASS_UDP.

4.14.2 Description de la syntaxe IP/UDP

4.14.2.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

4.14.2.2 Syntaxe abstraite IP/UDP APDU

Le Tableau 340 définit la syntaxe abstraite des PDU de couche application appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 340 doivent représenter le contenu de la DLSDU dans le Tableau 5.

Tableau 340 – Syntaxe IP/UDP APDU

Nom de l'APDU	Structure de l'APDU
ICMP-PDU	IPHeader ^a , ICMPHeader, Data*
^a La valeur 1 doit être attribuée à IPHeader.IP_Protocol pour indiquer ICMP.	

Le Tableau 341 définit les structures pour la substitution des éléments des structures APDU présentés dans le Tableau 340.

Tableau 341 – Substitutions IP/UDP

Nom de substitution	Structure
IPHeader	IP_VersionIHL(0x45), IP_TypeOfService, IP_TotalLength, IP_Identification, IP_Flags_FragOffset ^a , IP_TTL, IP_Protocol, IP_HeaderChecksum, IP_SrcIPAddress, IP_DstIPAddress, [IP_Options] ^b Le codage des champs doit être conforme à RFC 791.
ICMPHeader	ICMP_Type, ICMP_Code, ICMP_HeaderChecksum
UDPHeader	UDP_SrcPort, UDP_DstPort, UDP_DataLength, UDP_Checksum ^c
^a La fragmentation IP ne doit pas être utilisée pour UDP-RTC-PDU et UDP-RTA-PDU. ^b Le champ IP_Options doit être ignoré pour UDP-RTC-PDU et UDP-RTA-PDU. ^c Il convient d'attribuer une valeur nulle à UDP_Checksum pour RT_CLASS_UDP et RTA_CLASS_UDP afin d'améliorer les performances. Il n'est pas indispensable que le destinataire vérifie UDP_Checksum pour UDP-RTC-PDU et UDP-RTA-PDU.	

4.14.3 Syntaxe de transfert IP/UDP**4.14.3.1 Section de codage relative à IP, ICMP et UDP****4.14.3.1.1 Section de codage relative à ICMP****4.14.3.1.1.1 Codage du champ ICMP_Type**

Le codage des champs doit être conforme à RFC 792.

4.14.3.1.1.2 Codage du champ ICMP_Code

Le codage des champs doit être conforme à RFC 792.

4.14.3.1.1.3 Codage du champ ICMP_HeaderChecksum

Le codage des champs doit être conforme à RFC 792.

4.14.3.1.2 Section de codage relative à UDP**4.14.3.1.2.1 Codage du champ UDP_DataLength**

Le codage des champs doit être conforme à RFC 768.

4.14.3.1.2.2 Codage du champ UDP_Checksum

Le codage des champs doit être conforme à RFC 768.

4.14.3.1.2.3 Codage du champ UDP_SrcPort

Le codage des champs doit être conforme à RFC 768. Les valeurs définies sont données dans le Tableau 342.

Tableau 342 – UDP_SrcPort

Valeur (hexadécimale)	Signification	Utilisation
0x8892	IANA_PNIO_UDP_UNICAST_PORT	UDP-RTC-PDU and UDP-RTA-PDU
0x8893	IANA_PNIO_UDP_MULTICAST_PORT	Réservé
0x8894	IANA_PNIO_EPM_PORT	NDREMapLookupReq or NDREMapLookupFreeReq

0xC000 – 0xFFFF	IANA_FREE_PORT fait office de port dynamique et/ou privé	PROFINETIOServiceReqPDU and PROFINETIOServiceResPDU

4.14.3.1.2.4 Codage du champ UDP_DstPort

Le codage des champs doit être conforme à RFC 768. Les valeurs définies sont données dans le Tableau 343.

Tableau 343 – UDP_DstPort

Valeur (hexadécimale)	Signification	Utilisation
0x8892	IANA_PNIO_UDP_UNICAST_PORT	UDP-RTC-PDU and UDP-RTA-PDU
0x8893	IANA_PNIO_UDP_MULTICAST_PORT	Réservé
0x8894	IANA_PNIO_EPM_PORT	NDREMapLookupReq or NDREMapLookupFreeReq, also possible for each RPC call
0xC000 – 0xFFFF	IANA_FREE_PORT fait office de port dynamique et/ou privé	PROFINETIOServiceReqPDU and PROFINETIOServiceResPDU

4.14.3.1.3 Section de codage relative à IP

4.14.3.1.3.1 Codage du champ IP_DstIPAddress

Le codage des champs doit être conforme à RFC 791, RFC 2236 et à RFC 2365. Les valeurs définies sont données dans le Tableau 344 et le Tableau 345.

Tableau 344 – IP_DstIPAddress

Adresse IP	Plage	Signification
224.0.0.34	sous-réseau	SUBNET_MULTICAST_IP_ADDRESS_FOR_DISCOVERY
—	site	SITE_MULTICAST_IP_ADDRESS_FOR_DISCOVERY

Tableau 345 – IP Multicast DstIPAddress conformément à RFC 2365

Adresse IP de multidiffusion	Adresse MAC de multidiffusion associée	FrameID associé	Utilisation
239.0.0.0 – 239.192.247.255	—	—	Pas utilisé
239.192.248.0	01-00-5E-40-F8-00	0xF800	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP
239.192.248.1 – 239.192.251.254	01-00-5E-40-F8-01 – 01-00-5E-40-FB-FE	0xF801 – 0xFBFE	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP
239.192.251.255	01-00-5E-40-FB-FF	0xFBFF	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP
239.193.252.0 – 239.255.255.255	—	—	Pas utilisé

4.14.3.1.3.2 Codage du champ IP_VersionIHL

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.3 Codage du champ IP_TypeOfService

Le codage des champs doit être conforme à RFC 791 et à RFC 2474.

4.14.3.1.3.4 Codage du champ IP_TotalLength

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.5 Codage du champ IP_Identification

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.6 Codage du champ IP_Flags_FragOffset

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.7 Codage du champ IP_TTL

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.8 Codage du champ IP_Protocol

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.9 Codage du champ IP_HeaderChecksum

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.10 Codage du champ IP_SrcIPAddress

Le codage des champs doit être conforme à RFC 791.

4.14.3.1.3.11 Codage du champ IP_Options

Le codage des champs doit être conforme à RFC 791.

4.14.4 Cache ARP

L'utilisation du RFC 826 (ARP) et la commande du cache ARP peuvent être optimisées pour réduire le temps de basculement entre deux nœuds en utilisant l'adresse IP à différents moments. Ceci peut être effectué en supprimant l'entrée de cache ARP d'une adresse IP à chaque fois qu'une AR est libérée ou arrêtée prématurément.

4.15 Système de noms de domaine**4.15.1 Généralités**

L'utilisation de la norme RFC 1034 (DNS) est définie pour cette spécification. Elle inclut l'utilisation et la syntaxe de contenu du nom de domaine.

4.15.2 Définitions de primitives**4.15.2.1 Primitives échangées entre les machines distantes**

Les primitives de service distantes, y compris leurs paramètres associés, émises ou reçues par DNS sont décrites dans la définition de service et présentées dans le Tableau 346.

Tableau 346 – Primitives distantes émises ou reçues par DNS

Nom de primitive	Source	Destination	Paramètres associés	Fonctions
DNS_QueryName.req	DNS	DNS client	QNAME := StationName.RealStationName or StationName.AliasStationName, QCLASS, QTYPE	Résoudre le nom avec l'adresse IP
DNS_QueryName.cnf	DNS server	DNS	QNAME := StationName.RealStationName or StationName.AliasStationName, QCLASS, QTYPE, Answer := IPAddress	Résoudre le nom avec l'adresse IP

NOTE Des interfaces de haut niveau peuvent offrir un service GetHostByName à la place du service QNAME.

4.15.2.2 Primitives échangées entre les machines locales

Les primitives locales, y compris leurs paramètres associés, émises ou reçues par DNS sont décrites dans la définition de service et présentées dans le Tableau 347.

Tableau 347 – Primitives locales émises ou reçues par DNS

Nom de primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.15.3 Schéma de transition d'états de DNS

Voir RFC 1034 (DNS).

4.15.4 Description du diagramme d'états

Voir RFC 1034 (DNS).

4.15.5 Table d'états DNS

Voir RFC 1034 (DNS).

4.15.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 348 contient les fonctions, macros, temporisateurs et variables utilisés par le DNS ainsi que leurs arguments et leurs descriptions.

Tableau 348 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DNS

Nom	Type	Fonction/Signification
—	—	—

4.16 Configuration d'hôte dynamique

4.16.1 Généralités

L'utilisation de la norme RFC 2131 (DHCP) est définie pour la présente spécification. Elle inclut l'utilisation et la syntaxe du nom de l'ID client.

NOTE Un couplage strict entre DNS et le serveur DHCP empêche la discordance Nom contre Adresse IP.

4.16.2 Définitions de primitives

4.16.2.1 Primitives échangées entre les machines distantes

Les primitives de service distantes, y compris leurs paramètres associés, émises ou reçues par DHCP sont décrites dans la définition de service et présentées dans le Tableau 349.

Tableau 349 – Primitives distantes émises ou reçues par DHCP

Nom de primitive	Source	Destination	Paramètres associés	Fonctions
DHCP offer	DHCP	DHCP server	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 1, 3, 51, 54, 6, ...)	Second service dans une séquence à offrir au client DHCP un ensemble de paramètres.
DHCP acknowledgement	DHCP	DHCP server	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 1, 3, 51, 54, 6, ...)	Quatrième service dans une séquence à délivrer l'ensemble des paramètres agréés.
DHCP discovery	DHCP client	DHCP	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 50, 55, ...)	Premier service dans une séquence à trouver un serveur DHCP.
DHCP request	DHCP client	DHCP	OP, HTYPE, HLEN, HOPS, XID, SECS, FLAGS, CIADDR, YIADDR, SIADDR, GIADDR, CHADDR, DHCPOptions (53, 50, 54, ...)	Troisième service dans une séquence qui est utilisé par le client pour choisir un serveur.

4.16.2.2 Primitives échangées entre les machines locales

Les primitives locales, y compris leurs paramètres associés, émises ou reçues par DHCP sont décrites dans la définition de service et présentées dans le Tableau 350.

Tableau 350 – Primitives locales émises ou reçues par machines

Nom de primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.16.3 Schéma de transition d'états de DHCP

Voir RFC 2131 (DHCP).

4.16.4 Description du diagramme d'états

Voir RFC 2131 (DHCP).

4.16.5 Table d'états DHCP

Voir RFC 2131 (DHCP).

4.16.6 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 351 contient les fonctions, macros, temporisateurs et variables utilisés par le DHCP ainsi que leurs arguments et leurs descriptions.

Tableau 351 – Fonctions, Macros, Temporisateurs et Variables utilisés par le DHCP

Nom	Type	Fonction/Signification
—	—	—

4.17 Gestion de réseau simple

4.17.1 Vue d'ensemble

L'utilisation des normes RFC 1213 (MIB 2), RFC 2674 (Bridges with traffic classes), RFC 2863 (IF MIB), RFC 3418 (SNMP), RFC 3621 (Power over Ethernet MIB) et RFC 4836 (MAU MIB) et IEEE 802.1AB (LLDP MIB) est définie pour la présente spécification. De plus, le PNIO MIB est spécifié en 4.17.2 et la LLDP EXT MIB est spécifiée en 4.17.4.

4.17.2 Numéro d'entreprise pour PNIO MIB

La codage du champ enterprise number doit être conforme au Tableau 352.

Tableau 352 – Numéro d'entreprise

Valeur (décimale)	Signification
24 686	Numéro d'entreprise pour PNIO MIB

4.17.3 Référence croisée MIB

Il est recommandé que chaque (LLDP-MIB)."lldpLocPortDesc" soit défini séparément pour rendre possible une référence croisée à la MIB2.

Le lien entre le MIB2 et le LLDP MIB peut être effectué par la règle (LLDP-MIB)."lldpLocPortDesc" == (MIB2)."ifDescr".

4.17.4 LLDP EXT MIB

Voir Annexe W

4.18 Machines de Protocole de Mapping DLL communes

4.18.1 Vue d'ensemble

Les Machines de Protocole de Mapping DLL (DMPM) sont composées de plusieurs machines de protocole:

- Gestion des services appelés par d'autres entités de la Couche application (LMPM)
- Transfert de RT_CLASS_3 (RED_Relay)
- Transfert de DFP (DFP_Relay)

NOTE 1 Le transfert de trames normalisées est conforme à l'IEEE 802.1D et n'est donc pas abordé ici.

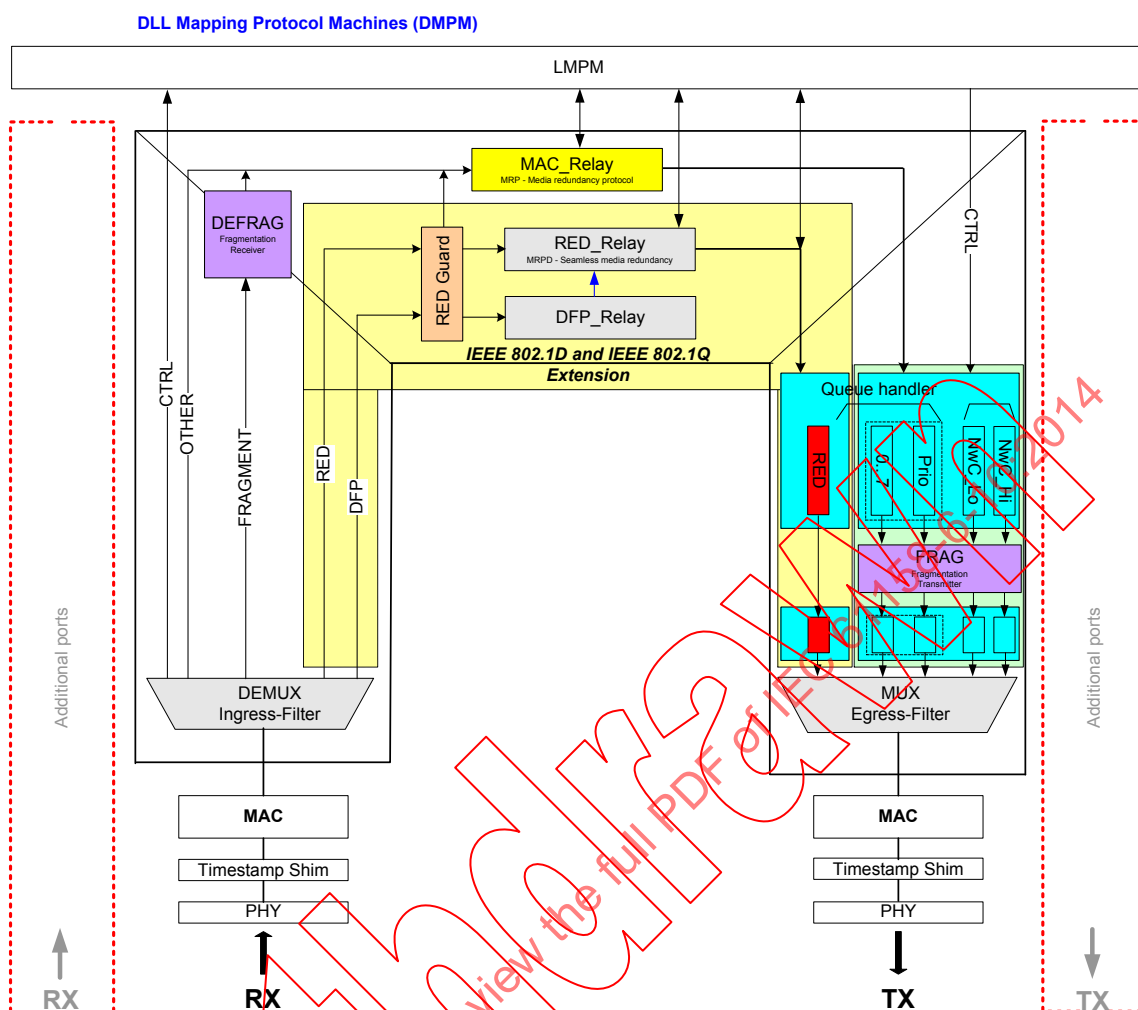
- Mapping avec l'Interface de Contrôle d'Accès au Support (MAC) et fonctions d'envoi/de réception pour les messages de synchronisation

NOTE 2 L'appel de MAC est conforme à l'IEEE 802.3 et n'est donc pas abordé ici.

- Fragmentation et réassemblage (FRAG et DEFRAG)
- SYNC_SHIMP pour ajouter des horodatages précis pour la synchronisation

L'Interface entre le gestionnaire de service (LMPM), le transfert des machines et le Mapping avec le Contrôle d'Accès au Support (MAC) est composée d'un ensemble de files d'attente et de données globales.

La Figure 79 illustre la structure des machines de protocole pour le DMPM d'un appareil et l'Annexe H décrit les couches de protocole supérieures.

**Légende**

Anglais	Français
DLL Mapping Protocol Machines (DMPM)	Machines de Protocole de Mapping DLL (DMPM)
MRP – Media Redundancy protocol	MRP – Protocole de Redondance de Support
DEFRAG Fragmentation Receiver	Destinataire de Fragmentation DEGRAG
RED Guard	RED Guard
MRPD – Seamless media redundancy	MRPD - Redondance de support sans coupure
IEEE 802.1D and IEEE 802.1Q Extension	IEEE 802.1D et Extension IEEE 802.1Q
OTHER	AUTRE
FRAGMENT	FRAGMENT
Queue handler	Gestionnaire de File d'attente
ORANGE	ORANGE
Prio	Prio
FRAG Fragmentation Transmitter	Émetteur de Fragmentation FRAG
MUX Ingress-Filter	Filtre de sortie MUX
TimeStamp Shim	TimeStamp Shim
Additional ports	Ports supplémentaires
CTRL	CTRL
OTHER	AUTRE

Anglais	Français
RED	ROUGE
DFP	DFP
MUX Egress-Filter	Filtre de Sortie MUX

Figure 79 – Structure des machines de protocole dans DMPM (pont)

NOTE 3 Cette image est souvent appelée modèle pantalon flottant (baggy pants).

4.18.2 Data Link Layer Mapping Protocol Machine (Machine de protocole de mapping de la couche de liaison de données)

4.18.2.1 Définitions de primitives

4.18.2.1.1 Primitives échangées entre les machines distantes

Les primitives de service, y compris leurs paramètres associés, émises ou reçues par la Machine de Protocole de Mapping de la Couche de Liaison de Données (LMPM) sont décrites dans la définition de service et présentées dans le Tableau 353.

Tableau 353 – Primitives distantes émises ou reçues par LMPM

Primitive	Source	Destination	Paramètres associés	Fonctions
DMUX_P_Data.ind	DEMUX	LMPM	CREP, S_Port, Tstamp, DA, SA, N_SDU	Les protocoles de gestion du réseau sont transmis directement au LMPM.
LMPM_A_Data.cnf	LMPM	APMS APMR DCPUCS DCPUCR DCPMCS DCPMCR DCPHMCS DCPHMCR	CREP, LMPM_status	—
LMPM_A_Data.ind	LMPM	APMS APMR DCPUCS DCPUCR DCPMCS DCPMCR DCPHMCS DCPHMCR	CREP, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_A_Data.req	APMS APMR DCPUCS DCPUCR DCPMCS DCPMCR DCPHMCS DCPHMCR	LMPM	CREP, DA, SA, VLANPrio, VLANID, A_SDU	—
LMPM_C_Data.cnf	LMPM	SCHEDULER (PROGRAMMATEUR)	CREP, LMPM_status	—
LMPM_C_Data.ind	LMPM	CPM	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status	—
LMPM_C_Data.req	SCHEDULER	LMPM	CREP, DA,	—

Primitive	Source	Destination	Paramètres associés	Fonctions
	PPM		SA, VLANPrio, VLANID, C_SDU, APDU_Status	
LMPM_N_Data.cnf	LMPM	APMS APMR IP	CREP, LMPM_status	L'IP du paramètre fictif est utilisé pour d'autres machines de protocole.
LMPM_N_Data.ind	LMPM	APMS APMR IP	CREP, DA, SA, VLANPrio, VLANID, N_SDU	L'IP du paramètre fictif est utilisé pour d'autres machines de protocole.
LMPM_N_Data.req	APMS APMR IP	LMPM	CREP, DA, SA, VLANPrio, VLANID, N_SDU	L'IP du paramètre fictif est utilisé pour d'autres machines de protocole.
LMPM_P_Data.cnf	LMPM	PTCP	CREP, D_Port, Tstamp, LMPM_status	
LMPM_P_Data.ind	LMPM	PTCP	CREP, S_Port, Tstamp, DA, SA, N_SDU	—
LMPM_P_Data.req	DelayReq DelayRsp MPSM	LMPM	CREP, D_Port, Tstamp, DA, SA, N_SDU	—
MAC_RelayData.cnf	MAC_RELAY	LMPM	CREP, Status	—
MAC_RelayData.ind	MAC_RELAY	LMPM	CREP, DA, SA, LT, VLANPrio, VLANID, C_SDU, APDU_status	—
MAC_RelayData.req	LMPM	MAC_RELAY	CREP, DA, SA, VLANPrio, VLANID, C_SDU, APDU_Status	Cette fonction est fournie par l'IEEE 802.1D. Elle recherche la base de données de filtrage et exécute pour chaque port de destination trouvé la M_UNITDATA.req.
MUX_Data.ind	MUX	LMPM	CREP, D_Port, Tstamp, Status	En raison de la définition d'une M_UNITDATA.req une indication est requise pour délivrer l'horodatage de transmission si nécessaire.
QueueHandler.req	LMPM	QueueHandler	CREP, D_Port, Tstamp, N_SDU	Pour les protocoles de gestion du réseau ou les protocoles qui n'utilisent pas la base de données de filtrage, la LMPM accède au MUX par le biais de la QueueHandler.req.

4.18.2.1.2 Primitives échangées entre les machines locales

Les primitives de service locales, y compris leurs paramètres associés, émises ou reçues par LMPM sont décrites dans la définition de service et présentées dans le Tableau 354.

Tableau 354 – Primitives locales émises ou reçues par LMPM

Primitive	Source	Destination	Paramètres associés	Fonctions
—	—	—	—	—

4.18.2.2 Schéma de transition d'états

Le schéma de transition d'états du LMPM est présenté à la Figure 80.

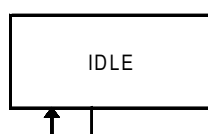


Figure 80 – Schéma de transition d'états du LMPM

Etats du LMPM

IDLE

Le LMPM constitue l'interface entre les ponts ou le MAC et les diagrammes d'états locaux.

4.18.2.3 Description du diagramme d'états

Le LMPM forme l'interface entre MAC et l'utilisateur LMPM pour différentes données et différents services de gestion. Les services sont tous gérés à l'état IDLE.

La demande de service est validée en premier. Une validation négative donne lieu à une confirmation négative. Sinon, le service est placé dans la file d'attente de service à la priorité appropriée. Les services exécutés par MAC sont déplacés des files d'attente de demandes vers les files d'attente de confirmation. Tous les services entrants valides sont placés dans la file d'attente des indications.

4.18.2.4 Table d'états LMPM

La table d'états LMPM est présentée dans le Tableau 355.

Tableau 355 – Table d'états LMPM

#	Etat en cours	Événement /Condition =>Action	Etat suivant
1	IDLE	LMPM_P_Data.req () /CheckPDUType () == TimeStamped => QueueHandler.req () //NOTE Après le retrait de la file d'attente et la transmission, un MUX_Data.ind () est généré à partir du MUX et est émis au LMPM pour acheminer le Tstamp de transmission pour la trame	IDLE

#	Etat en cours	Evénement /Condition =>Action	Etat suivant
2	IDLE	MUX_Data.ind () => ignore LMPM_P_Data.cnf ()	IDLE
3	IDLE	LMPM_N_Data.req () /CheckPDUType () == other //NOTE "Autre" inclut RT_CLASS_UDP and RTA_CLASS_UDP => LMPM_N_Data.cnf () MAC_RELAY_Data.req ()	IDLE
4	IDLE	MAC_RELAY_Data.cnf () => ignore	IDLE
5	IDLE	LMPM_A_Data.req () /CheckPDUType () == RTA_CLASS_1 => LMPM_A_Data.cnf () MAC_RELAY_Data.req ()	IDLE
6	IDLE	LMPM_C_Data.req () /CheckPDUType () == RT_CLASS_1 => LMPM_C_Data.cnf () MAC_RELAY_Data.req ()	IDLE
7	IDLE	LMPM_C_Data.req () /CheckPDUType () == RT_CLASS_2 => LMPM_C_Data.cnf () MAC_RELAY_Data.req ()	IDLE
8	IDLE	LMPM_C_Data.req () /CheckPDUType () == RT_CLASS_3 => LMPM_C_Data.cnf () QueueHandler.req ()	IDLE
9	IDLE	DMUX_P_Data.ind () /CheckPDUType () == TimeStamped => LMPM_P_Data.ind ()	IDLE
10	IDLE	RED_RELAY_Data.ind () /CheckPDUType () == RT_CLASS_3 => LMPM_C_Data.ind ()	IDLE
11	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == RT_CLASS_2 => LMPM_C_Data.ind ()	IDLE
12	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == RT_CLASS_1 => LMPM_C_Data.ind ()	IDLE
13	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == RTA_CLASS_1 => LMPM_A_Data.ind ()	IDLE
14	IDLE	MAC_RELAY_Data.ind () /CheckPDUType () == other => LMPM_N_Data.ind ()	IDLE

4.18.2.5 Fonctions, Macros, Temporisateurs et Variables

Le Tableau 356 contient les fonctions, macros, temporisateurs et variables utilisés par le LMPM ainsi que leurs arguments et leurs descriptions.

Tableau 356 – Fonctions, Macros, Temporisateurs et Variables utilisés par le LMPM

Nom	Type	Fonction/Signification
CheckPDUType	Fonction	Cette fonction locale vérifie le type de PDU et retourne les valeurs - RT_CLASS_1 - RT_CLASS_2 - RT_CLASS_3 - RTA_CLASS_1 - Other - TimeStamped

5 Spécification du protocole de couche application pour la périphérie décentralisée

5.1 Description de la syntaxe FAL

5.1.1 Référence de syntaxe abstraite DLPDU

La syntaxe abstraite de DLPDU selon 4.1.1 doit être appliquée.

5.1.2 Syntaxe abstraite APDU

Le Tableau 357 définit la syntaxe abstraite des PDU de couche application appelées APDU. L'ordre défini des octets doit être utilisé pour acheminer les APDU. Les APDU dans le Tableau 357 doivent représenter le contenu de la DLSDU dans le Tableau 5.

Tableau 357 – Substitutions d'IO APDU

Nom de substitution	Structure
RTA-SDU	AlarmNotification-PDU ^ AlarmAck-PDU
BlockHeader	BlockType, BlockLength, BlockVersionHigh, BlockVersionLow
AlarmNotification-PDU	BlockHeader, AlarmType, API ^a , SlotNumber ^a , SubslotNumber ^a , ModuleIdentNumber, SubmoduleIdentNumber, AlarmSpecifier, AlarmPayload ^a Ces champs adressent le sous-module qui peut être la source ou le puits d'une alarme. Ils contiennent la source si l'appareil E/S émet une alarme et le puits si l'appareil de commande E/S émet et une alarme. Le champ BlockType au sein de BlockHeader doit être mis à AlarmNotification high ou low conformément au Tableau 358.

Nom de substitution	Structure
AlarmAck-PDU	BlockHeader, AlarmType, API, SlotNumber, SubslotNumber, AlarmSpecifier, PNIOStatus Le champ BlockType au sein de BlockHeader doit être mis à AlarmAck high ou low conformément au Tableau 358. Cas particulier "No Error": PNIOStatus(=0x00, 0x00, 0x00, 0x00) Cas particulier "Alarm Type Not Supported" or if a reserved Alarm Type is used: PNIOStatus(=0xDA, 0x81, 0x3C, 0x00) Cas particulier "Wrong Submodule State": PNIOStatus(=0xDA, 0x81, 0x3C, 0x01) Cas particulier "IOCARSR: Sauvegarde – Alarme non exécutée": PNIOStatus(=0xDA, 0x81, 0x3C, 0x02)
AlarmPayload	[MaintenanceItem], [DiagnosisItem] ^ [AlarmItem] ^ [Upload&RetrievalItem] ^ [iParameterItem] Voir Tableau 361
MaintenanceItem	UserStructureIdentifier, BlockHeader, Padding, Padding, MaintenanceStatus
Upload&RetrievalItem	UserStructureIdentifier, BlockHeader, Padding, Padding, (URRecordIndex, URRecordLength)* Cas particulier "Retrieve all stored records": UserStructureIdentifier, BlockHeader(=0x0F04), Padding, Padding, URRecordIndex(=0), URRecordLength(=0)
iParameterItem	UserStructureIdentifier, BlockHeader, Padding, Padding, iPar_Req_Header, Max_Segm_Size, Transfer_Index, Total_iPar_Size
DiagnosisItem	UserStructureIdentifier, ChannelDiagnosisData* ^ ExtChannelDiagnosisData* ^ QualifiedChannelDiagnosisData* ^ DiagnosisData* Cas particulier ChannelDiagnosisData: UserStructureIdentifier(=0x8000), ChannelDiagnosisData* Cas particulier ExtChannelDiagnosisData: UserStructureIdentifier(=0x8002), ExtChannelDiagnosisData* Cas particulier QualifiedChannelDiagnosisData: UserStructureIdentifier(=0x8003), QualifiedChannelDiagnosisData* Cas particulier DiagnosisData avec ManufacturerSpecificDiagnosis: UserStructureIdentifier(=0x0000 – 0x7FFF), (BlockHeader, API, ManufacturerSpecific-Diagnosis)*
AlarmItem	UserStructureIdentifier, Data*
PROFINETIO-ServiceReqPDU	IODConnectReq ^ IODWriteReq ^ IODWriteMultipleReq ^ IODReadReq ^ IODControlReq ^ IODReleaseReq ^ IOXControlReq
PROFINETIO-ServiceResPDU	IODConnectRes ^ IODWriteRes ^ IODWriteMultipleRes ^ IODReadRes ^ IODControlRes ^ IODReleaseRes ^ IOXControlRes

Nom de substitution	Structure
IODConnectReq with StartupMode:=1	ARBlockReq, {[IOCRBlockReq*], [AlarmCRBlockReq], [ExpectedSubmoduleBlockReq*]^b, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq], [IRInfoBlock], [SRInfoBlock], [ARVendorBlockReq*], [ARFSUBlock] } Cas particulier: "IOCARSingle" ou "IOSAR": ARBlockReq, {[IOCRBlockReq*, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq], [IRInfoBlock], [SRInfoBlock], [ARVendorBlockReq*], [ARFSUBlock] } Cas particulier: "IOSAR avec ARProperties.DeviceAccess:=1": ARBlockReq^c Cas particulier: "IOCARSR": ARBlockReq, {[IOCRBlockReq*, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq], [IRInfoBlock], SRInfoBlock, [ARVendorBlockReq*], [ARFSUBlock] } ^a Le champ MCRBlockReq doit uniquement être présent si au moins un IOCRBlockReq contient la valeur MULTICAST_CONSUMER_CR comme IO CR Type. ^b Le champ ExpectedSubmoduleBlockReq doit uniquement contenir les sous-modules auxquels il est fait référence dans au moins un IOCRBlockReq. ^c Le champ CMLInitiatorActivityTimeoutFactor indique le temps de fonctionnement du chien de garde.
IODConnectRes with StartupMode:=1	ARBlockRes, {[IOCRBlockRes*], [AlarmCRBlockRes], [ModuleDiffBlock], [ARRPCBlockRes]^a, ARServerBlockRes, [ARVendorBlockRes*]^b } Cas particulier: "IOCARSingle" ou "IOSAR": ARBlockRes, {[IOCRBlockRes*, AlarmCRBlockRes, [ModuleDiffBlock], [ARRPCBlockRes]^a, ARServerBlockRes, [ARVendorBlockRes*]^b } Cas particulier: "IOSAR avec ARProperties.DeviceAccess=1": ARBlockRes, ARServerBlockRes ^a Le champ ARRPCBlockRes doit uniquement être présent si IODConnectReq contient un ARRPCBlockReq. ^b Le nombre d'entrées ARVendorBlockRes doit être égal au nombre d'entrées de ARVendorBlockReq dans IODConnectReq.

Nom de substitution	Structure
IODConnectReq with StartupMode:=0	ARBlockReq, {[IOCRBlockReq*], [AlarmCRBlockReq], [ExpectedSubmoduleBlockReq*]^b, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq] } Cas particulier: "IOCARSingle" ou "IOSAR": ARBlockReq, {[IOCRBlockReq*, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq] } Cas particulier: "IOSAR avec ARProperties.DeviceAccess:=1": ARBlockReq^c Cas particulier: "IOCARSingle utilisant RT_CLASS_3": ARBlockReq, {[IOCRBlockReq*^d, AlarmCRBlockReq, ExpectedSubmoduleBlockReq*, [PrmServerBlock], [MCRBlockReq*]^a, [ARRPCBlockReq] } ^a Le champ MCRBlockReq doit uniquement être présent si au moins un IOCRBlockReq contient la valeur MULTICAST_CONSUMER_CR comme IO CR Type. ^b Le champ ExpectedSubmoduleBlockReq doit uniquement contenir les sous-modules auxquels il est fait référence dans au moins un IOCRBlockReq. ^c Le champ CMInitiatorActivityTimeoutFactor indique le temps de fonctionnement du chien de garde. ^d Pour établir un "IOCARSingle using RT_CLASS_3" avec les relations de communication d'entrée et de sortie RT_CLASS_3, une relation de communication de ce type doit exister.
IODConnectRes with StartupMode:=0	ARBlockRes, {[IOCRBlockRes*], [AlarmCRBlockRes], [ModuleDiffBlock], [ARRPCBlockRes]^a } Cas particulier: "IOCARSingle" ou "IOSAR": ARBlockRes, {[IOCRBlockRes*, AlarmCRBlockRes, [ModuleDiffBlock], [ARRPCBlockRes]^a } Cas particulier: "IOSAR avec ARProperties.DeviceAccess=1": ARBlockRes ^a Le champ ARRPCBlockRes doit uniquement être présent si IODConnectReq contient un ARRPCBlockReq.
IODWriteMultipleReq	IODWriteReqHeader^a, (IODWriteReqHeader, RecordDataWrite, [Padding*]^b)* ^a La valeur de l'index de paramètre doit être 0xE040. ^b Le nombre d'octets de remplissage (value=0) doit être 0,1,2,3 pour obtenir un alignement 32 bit au IODWriteReqHeader suivant.
IODWriteReq	IODWriteReqHeader, RecordDataWrite
IODWriteReqHeader	BlockHeader, SeqNumber, ARUID, API, SlotNumber, SubslotNumber, Padding*^a, Index, RecordDataLength, RWPadding*^b ^a La demande d'écriture contient 2 octets de remplissage (value=0). ^b La demande d'écriture contient 24 octets de remplissage (value=0).
IODWriteRes	IODWriteResHeader^a ^a La valeur du paramètre PNIOStatus doit être (0,0,0,0) ou un double de PNIOStatus de NDRDataResponse.

Nom de substitution	Structure
IODWriteMultipleRes	IODWriteResHeader ^{a b}, (IODWriteResHeader)* ^c ^a La valeur de l'index de paramètre doit être 0xE040. ^b La valeur du paramètre PNIOStatus doit être (0,0,0,0) ou un double de PNIOStatus de NDRDataResponse. ^c La valeur du paramètre PNIOStatus contient le résultat d'une écriture particulière.
IODWriteResHeader	BlockHeader, SeqNumber, ARUUID, API, SlotNumber, SubslotNumber, Padding* ^a, Index, RecordDataLength, AdditionalValue1, AdditionalValue2, PNIOStatus, RWPadding* ^b ^a La demande d'écriture contient 2 octets de remplissage (value=0).. ^b La demande d'écriture contient 16 octets de remplissage (value=0)
IODReadReq	IODReadReqHeader, RecordDataReadQuery
IODReadReqHeader	BlockHeader, SeqNumber, ARUUID, API, SlotNumber, SubslotNumber, Padding* ^a, Index, RecordDataLength, [TargetARUUID] ^b, RWPadding* ^c ^a La demande de lecture contient 2 octets de bourrage (value=0) . ^b Le champ facultatif TargetARUUID ne doit être utilisé que conjointement avec la valeur 0 d'ARUUID (AR Implicite). ^c La demande de lecture contient 24 octets ou 8 octets (value=0) de bourrage .
RecordDataReadQuery	UploadBLOBQuery ^a (BlockHeader, Data* ^a) ^a NULL ^a L'utilisation, la structure et les valeurs doivent être définies a) par le fabricant pour la plage d'index spécifique au fabricant b) par le profil pour les plages d'index spécifiques au profil c) par la présente norme pour toutes les autres plages d'index
IODReadRes	IODReadResHeader, RecordDataRead
IODReadResHeader	BlockHeader, SeqNumber, ARUUID, API, SlotNumber, SubslotNumber, Padding* ^a, Index, RecordDataLength, AdditionalValue1, AdditionalValue2, RWPadding* ^b ^a La réponse de lecture contient 2 octets de bourrage (value=0). ^b La réponse de lecture contient 20 octets de bourrage (value=0).
IODControlReq	ControlBlockConnect ^a ^a (ControlBlockConnect, SubmoduleListBlock) ^b ^a ControlBlockPlug ^c ^a Doit être utilisé dans le contexte d'une connexion ou d'une séquence PrmBegin/PrmEnd pour ControlCommand.PrmEnd ^b Doit être utilisé dans le contexte d'une séquence PrmBegin/PrmEnd pour ControlCommand.PrmBegin avec SubmoduleListBlock ^c Doit être utilisé dans le contexte d'une connexion/libération pour ControlCommand.PrmEnd
IODControlRes	ControlBlockConnect ^a ^a (ControlBlockConnect, [ModuleDiffBlock]) ^b ^a ControlBlockPlug ^c ^a NULL ^d ^a Doit être utilisé dans le contexte d'une connexion ou d'une séquence PrmBegin/PrmEnd pour ControlCommand.PrmEnd ^b Doit être utilisé dans le contexte d'une séquence PrmBegin/PrmEnd pour ControlCommand.PrmBegin si la vérification du SubmoduleListBlock conduit à un ModuleDiffBlock ^c Doit être utilisé dans le contexte d'une connexion/libération pour ControlCommand.PrmEnd ^d En cas de réponse négative, NULL doit être transmis.

Nom de substitution	Structure
IODReleaseReq	ReleaseBlock
IODReleaseRes	ReleaseBlock ^ NULL ^a ^a En cas de réponse négative, NULL doit être transmis.
IOXControlReq with StartupMode:=1	(ControlBlockConnect, [ModuleDiffBlock]) ^a ^ (ControlBlockPlug, [ModuleDiffBlock])) ^b ^a Doit être utilisé dans le contexte d'une séquence PrmBegin/PrmEnd pour ControlCommand.ApplicationReady avec ModuleDiffBlock si nécessaire. ^b Doit être utilisé dans le contexte d'une connexion/libération pour ControlCommand.ApplicationReady avec ModuleDiffBlock si nécessaire. Le champ ModuleDiffBlock doit uniquement être présent s'il contient des entrées. Un ModuleDiffBlock vide doit être ignoré.
IOXControlRes with StartupMode:=1	ControlBlockConnect ^a ^ ControlBlockPlug ^b ^ NULL ^c ^a Doit être utilisé dans le contexte d'une séquence PrmBegin/PrmEnd pour ControlCommand.ApplicationReady ^b Doit être utilisé dans le contexte d'une connexion/libération pour ControlCommand.ApplicationReady ^c En cas de réponse négative, NULL doit être transmis.
IOXControlReq with StartupMode:=0	(ControlBlockConnect, [ModuleDiffBlock]) ^a ^ (ControlBlockPlug, [ModuleDiffBlock])) ^b ^ ControlBlockRFC ^c ^ ControlBlockRTC ^d ^a Doit être utilisé dans le contexte d'une connexion pour ControlCommand.ApplicationReady avec ModuleDiffBlock si nécessaire. ^b Doit être utilisé dans le contexte d'une connexion/libération pour ControlCommand.ApplicationReady avec ModuleDiffBlock si nécessaire. ^c Doit être utilisé pour ControlCommand.ReadyForCompanion ^d Doit être utilisé pour ControlCommand.ReadyForRT_CLASS_3 Le champ ModuleDiffBlock doit uniquement être présent s'il contient des entrées. Un ModuleDiffBlock vide doit être ignoré.
IOXControlRes avec StartupMode:=0	ControlBlockConnect ^a ^ ControlBlockPlug ^b ^ ControlBlockRFC ^c ^ ControlBlockRTC ^d ^ NULL ^e ^a Doit être utilisé dans le contexte d'une connexion pour ControlCommand.ApplicationReady ^b Doit être utilisé dans le contexte d'une connexion/libération pour ControlCommand.ApplicationReady ^c Doit être utilisé pour ControlCommand.ReadyForCompanion ^d Doit être utilisé pour ControlCommand.ReadyForRT_CLASS_3 ^e En cas de réponse négative, NULL doit être transmis.

Nom de substitution	Structure
ARBlockReq	BlockHeader, ARType, ARUUID, SessionKey, CMInitiatorMacAdd, CMInitiatorObjectUUID, ARProperties, CMInitiatorActivityTimeoutFactor, InitiatorUDPRTPort^a, StationNameLength, CMInitiatorStationName^b ^a Si aucune RT_CLASS_UDP CR n'est utilisée, la valeur d'InitiatorUDPRTPort doit être mise à 0x8892. ^b Ce codage conduit à un désalignement si la longueur de CMInitiatorStationName est inégale. Il convient qu'un appareil E/S prenne en charge un [Padding*] derrière pour garantir un alignement Unsigned32. Le champ BlockType au sein de BlockHeader doit être mis à ARBlockReq conformément au Tableau 358.
ARBlockRes	BlockHeader, ARType, ARUUID, SessionKey, CMResponderMacAdd, ResponderUDPRTPort ^a Si aucune RT_CLASS_UDP CR n'est utilisée, la valeur pour ResponderUDPRTPort doit être mise à 0x8892. Le champ BlockType au sein de BlockHeader doit être mis à ARBlockRes conformément au Tableau 358.
IOCRBlockReq	BlockHeader, IOCRTType, IOCReference, LK, IOCProperties, DataLength, FrameID, SendClockFactor, ReductionRatio, Phase, Sequence, FrameSendOffset, DataHoldFactor^a, DataHoldFactor, IOCRTagHeader, IOCRTMulticastMACAdd, NumberOfAPIs, (API, NumberOfIODataObjects, (SlotNumber, SubslotNumber, IODataObjectFrameOffset)*, NumberOfIOCS, (SlotNumber, SubslotNumber, IOCSFrameOffset)*)* ^a Ce champ existe pour des raisons d'héritage, il a été formellement appelé WatchDogFactor. Le champ BlockType au sein du BlockHeader doit être mis à IOCRBlockReq conformément au Tableau 358. Cas particulier IOCRTType == OUTPUT_CR sans utiliser RT_CLASS_3 Le champ FrameID ne doit pas être évalué. Cas particulier IOCRTType == OUTPUT_CR en utilisant RT_CLASS_3 Le champ FrameID doit contenir le FrameID utilisé dans le PDIRData correspondant. Cas particulier IOCRTType == MULTICAST_CONSUMER_CR Le champ FrameID doit contenir un FrameID issu de la plage multidiffusion définie. L'outil de configuration doit garantir le caractère unique de ce FrameID dans le contexte du projet.
MCRBlockReq	BlockHeader, IOCReference, AddressResolutionProperties, MCITimeoutFactor, StationNameLength, ProviderStationName, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
IOCRBlockRes	BlockHeader, IOCRTType, IOCReference, FrameID Le champ BlockType au sein du BlockHeader doit être mis à IOCRBlockRes conformément au Tableau 358.

Nom de substitution	Structure
ExpectedSubmodule-BlockReq	BlockHeader, NumberOfAPIs, (API, SlotNumber^a, ModuleIdentNumber, ModuleProperties, NumberOfSubmodules, (SubslotNumber, SubmoduleIdentNumber, SubmoduleProperties^b, (DataDescription, SubmoduleDataLength, LengthIOPS, LengthIOCS)*))* ^a Ce paramètre est constant pour un ExpectedSubmoduleBlockReq, car chacun contient les données d'un seul emplacement. ^b Le champ SubmoduleProperties.Type détermine le nombre de blocs de description de données consécutifs.. Le champ BlockType au sein du BlockHeader doit être mis à ExpectedSubmoduleBlockReq conformément au Tableau 358.
ModuleDiffBlock	BlockHeader, NumberOfAPIs, (API, NumberOfModules, (SlotNumber, ModuleIdentNumber^a, ModuleState^b, NumberOfSubmodules, [(SubslotNumber, SubmoduleIdentNumber^a, SubmoduleState)*]))* ^a Données d'identification réelles ^b Si ModuleState=NO_MODUL, NumberOfSubmodules doit être nul. La partie suivante doit être ignorée. Pour tous les autres ModuleState, seul SubmoduleState doit être utilisé pour décider si le sous-module doit être paramétré. Le champ BlockType au sein du BlockHeader doit être mis à ModuleDiffBlock conformément au Tableau 358.
AlarmCRBlockReq	BlockHeader, AlarmCRType, LT, AlarmCRProperties, RTATimeoutFactor, RTARetries, LocalAlarmReference, MaxAlarmDataLength, AlarmCRTagHeaderHigh, AlarmCRTagHeaderLow Le champ BlockType au sein du BlockHeader doit être mis à AlarmCRBlockReq conformément au Tableau 358.
AlarmCRBlockRes	BlockHeader, AlarmCRType, LocalAlarmReference, MaxAlarmDataLength Le champ BlockType au sein du BlockHeader doit être mis à AlarmCRBlockRes conformément au Tableau 358.
PrmServerBlock	BlockHeader, ParameterServerObjectUUID, ParameterServerProperties, CMInitiatorActivityTimeoutFactor, StationNameLength, ParameterServerStationName^a ^a Il convient qu'un appareil E/S prenne en charge un [Padding*] derrière pour garantir un alignement avec Unsigned32. Le champ BlockType au sein du BlockHeader doit être mis à PrmServerBlock conformément au Tableau 358.
ARRPCBlockReq	BlockHeader, InitiatorRPCServerPort Le champ BlockType au sein du BlockHeader doit être mis à ARRPCBlockReq conformément au Tableau 358.
ARRPCBlockRes	BlockHeader, ResponderRPCServerPort Le champ BlockType au sein du BlockHeader doit être mis à ARRPCBlockRes conformément au Tableau 358.

Nom de substitution	Structure
ARServerBlockRes	BlockHeader, StationNameLength, CMResponderStationName, [Padding* ^a] ^a Le nombre d'octets de remplissage doit garantir l'alignement Unsigned32. Le champ BlockType au sein du BlockHeader doit être mis à ARServerBlockRes conformément au Tableau 358.
IRInfoBlock	BlockHeader, Padding* ^a, IRDataUUID, Padding*^a, NumberOfIOCRs ^b, [(IOCRReference, SubframeOffset, SubframeData)*] ^a Le nombre d'octets de bourrage doit être 2. ^b Ce champ doit être codé en zéro si aucun DFP n'est utilisé. Le champ BlockType au sein du BlockHeader doit être mis à IRInfoBlock conformément au Tableau 358.
SRInfoBlock	BlockHeader, RedundancyDataHoldFactor, SRProperties Le champ BlockType au sein du BlockHeader doit être mis à SRInfoBlock conformément au Tableau 358.
ARFSUBlock	ARFSUDDataAdjust Le champ BlockType au sein du BlockHeader doit être mis à ARFSUBlock conformément au Tableau 358.
ARVendorBlockReq	BlockHeader, APStructureIdentifier, API, [Data*], [Padding* ^a] ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. Le champ BlockType au sein du BlockHeader doit être mis à ARVendorBlockReq conformément au Tableau 358. Cas particulier "Règles d'identification étendues" BlockHeader, APStructureIdentifier (:=0x8000), API (:=0), ExpectedExtendedIdentification-Info, [Padding* ^a]
ARVendorBlockRes	BlockHeader, APStructureIdentifier, API, [Data*], [Padding* ^a] ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. Le champ BlockType au sein du BlockHeader doit être mis à ARVendorBlockRes conformément au Tableau 358. Cas particulier "Règles d'identification étendues" BlockHeader, APStructureIdentifier (:=0x8000), API (:=0), RealExtendedIdentificationInfo, [Padding* ^a]
ExpectedExtended-IdentificationInfo	ExtendedIdentificationInfo
RealExtended-IdentificationInfo	ExtendedIdentificationInfo
ExtendedIdentification-Info	ExtendedIdentificationUUID, ExtendedIdentificationVersionHigh, ExtendedIdentificationVersionLow

Nom de substitution	Structure
SubframeBlock	BlockHeader, FrameID^a, SFIOCRProperties, SubframeData*^b ^a Pour MRPD seul le SubframeBlock pour le FrameID avec le bit de poids faible mis à zéro doit être créé. ^b L'ordre des octets de SubframeData doit être utilisé pour le chemin non-redondant. Le champ BlockType au sein du BlockHeader doit être mis à SubframeBlock conformément au Tableau 358.
PDIRSubframeData	BlockHeader, NumberOfSubframeBlocks, SubframeBlock* Le champ BlockType au sein du BlockHeader doit être mis à PDIRSubframeBlock conformément au Tableau 358.
ControlBlockConnect	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a Le nombre d'octets de bourrage doit être 2. Les valeurs IOBlockReq, IOBlockRes, IOXBlockReq et IOXBlockRes doivent être attribuées au champ BlockType de BlockHeader conformément au Tableau 358.
ControlBlockPlug	BlockHeader, Padding*^a, ARUID, SessionKey, AlarmSequenceNumber, ControlCommand, ControlBlockProperties ^a Le nombre d'octets de bourrage doit être 2. Les valeurs IOXBlockReq, IOXBlockRes doivent être attribuées au champ BlockType de BlockHeader conformément au Tableau 358. Le champ AlarmSequenceNumber doit contenir la valeur du sous-champ AlarmSpecifier.SequenceNumber du AlarmNotification-PDU correspondant.
ControlBlockRFC	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a Le nombre d'octets de bourrage doit être 2. Le champ BlockType au sein du BlockHeader doit être mis à ReadyForCompanionBlock conformément au Tableau 358.
ControlBlockRTC	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a Le nombre d'octets de bourrage doit être 2. Le champ BlockType au sein du BlockHeader doit être mis à ReadyForRTCLASS3Block conformément au Tableau 358.
ReleaseBlock	BlockHeader, Padding*^a, ARUID, SessionKey, Padding*^a, ControlCommand, ControlBlockProperties ^a Le nombre d'octets de bourrage doit être 2. Le champ BlockType au sein du BlockHeader doit être mis à ReleaseBlock conformément au Tableau 358.

Nom de substitution	Structure
SubmoduleListBlock	BlockHeader, NumberOfEntries, (API, SlotNumber, SubslotNumber)* Le champ BlockType au sein du BlockHeader doit être mis à ReleaseBlock conformément au Tableau 358.
RecordDataRead	DiagnosisData* ^ ExpectedIdentificationData* ^ RealIdentificationData* ^ SubstituteValue ^ RecordInputDataObjectElement ^ RecordOutputDataObjectElement ^ Data* ^ ARData ^ IMDData ^ LogBookData ^ ModuleDiffBlock ^ APIDData ^ PDPortDataAdjust* ^ PDPortDataCheck* ^ PDPortDataReal* ^ PDIRData ^ PDSyncData* ^ PdevData ^ IsochronousModeData* ^ PDInterfaceAdjust ^ PDInterfaceMrpDataAdjust* ^ PDInterfaceMrpDataCheck* ^ PDInterfaceMrpDataReal* ^ PDPortMrpDataAdjust ^ PDPortMrpDataReal ^ PDPortFODDataReal ^ PDPortFODDataAdjust ^ PDPortFODDataCheck ^ PDRealData* ^ PDExpectedData* ^ PDNCDataCheck ^ PDPortStatistic ^ I&M0FilterData ^ ARFSUDDataAdjust ^ PDInterfaceFSUDDataAdjust ^ PDInterfaceDataReal ^ AutoConfiguration ^ PDIRSubframeData ^ NestedDiagnosisInfo ^ UploadBLOB ^ NULL ^a ^a NULL doit être utilisé si un enregistrement de données connu obligatoire est vide (Diagnosis, ModuleDiffBlock, ARData, par exemple).
RecordDataWrite	SubstituteValue ^ Data* ^ IMDDataWrite ^ PDPortDataAdjust ^ PDPortDataCheck ^ PDIRData ^ PDSyncData ^ IsochronousModeData ^ PDInterfaceMrpDataAdjust ^ PDInterfaceMrpDataCheck ^ PDPortMrpDataAdjust ^ PDPortFODDataAdjust ^ PDPortFODDataCheck ^ PDNCDataCheck ^ ARFSUDDataAdjust ^ PDInterfaceFSUDDataAdjust ^ PDIRSubframeData ^ PDInterfaceAdjust ^ CombinedObjectContainer
DiagnosisData avec BlockVersionLow=0	BlockHeader, ChannelDiagnosis ^ ManufacturerSpecificDiagnosis ^ ExtChannelDiagnosis BlockVersionLow=0 doit être pris en charge par appareil de commande E/S et superviseur d'E/S. Il ne doit pas être produit par l'appareil d'E/S.
DiagnosisData avec BlockVersionLow=1	BlockHeader, API, ChannelDiagnosis ^ ManufacturerSpecificDiagnosis ^ ExtChannelDiagnosis ^ QualifiedChannelDiagnosis BlockVersionLow=1 doit être généré par l'appareil E/S et pris en charge par l'appareil de commande E/S et le superviseur E/S.
ChannelDiagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^a , UserStructureIdentifier(0x8000), ChannelDiagnosisData* ^a Le champ ChannelProperties.Type, le champ ChannelProperties.Direction, le champ ChannelProperties.Maintenance doivent être mis à zéro. Le champ ChannelPro_perties.Specifier doit être réglé pour apparaître si au moins un ChannelProperties.Specifier dans ChannelDiagnosisData est réglé pour apparaître conjointement à ChannelProperties.Maintenance(=diagnosis). Sinon, le champ ChannelProperties.Specifier doit être mis pour disparaître.
ChannelDiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType
Manufacturer-SpecificDiagnosis	SlotNumber, SubslotNumber, ChannelNumber, ChannelProperties, UserStructureIdentifier, Data*
ExtChannelDiagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^a , UserStructureIdentifier(0x8002), ExtChannelDiagnosisData* ^a Le champ ChannelProperties.Type, le champ ChannelProperties.Direction, le champ ChannelProperties.Maintenance doivent être mis à zéro. Le champ ChannelPro_perties.Specifier doit être réglé pour apparaître si au moins un ChannelProperties.Specifier dans ExtChannelDiagnosisData est réglé pour apparaître conjointement à ChannelProperties.Maintenance(=diagnosis). Sinon, le champ ChannelProperties.Specifier doit être mis pour disparaître.
ExtChannel-DiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType, ExtChannelAddValue

Nom de substitution	Structure
QualifiedChannel-Diagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^a , UserStructureIdentifier(0x8003), QualifiedChannelDiagnosisData*
QualifiedChannel-DiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType, ExtChannelAddValue, QualifiedChannelQualifier
ExpectedIdentification-Data avec BlockVersionLow = 0 Legacy	BlockHeader, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber))*
ExpectedIdentification-Data avec BlockVersionLow = 1	BlockHeader, NumberOfAPIs, (API, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber))*)*
RealIdentificationData avec BlockVersionLow = 0	BlockHeader, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber))*
RealIdentificationData avec BlockVersionLow = 1	BlockHeader, NumberOfAPIs, (API, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber))*)*
PDIRData avec BlockVersionLow = 1	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, PDIRGlobalData, PDIRFrameData ^a ^b , PDIRBeginEndData ^a Ce bloc peut contenir des informations relatives à la réception, la transmission et au transfert ou uniquement à la réception et à la transmission. ^b Si ce bloc existe, les trames DFP frames doivent être intégrées.
PDSyncData avec BlockVersionLow = 2	BlockHeader, Padding, Padding, PTCP_SubdomainUUID, ReservedIntervalBegin, ReservedIntervalEnd, PLLWindow, SyncSendFactor, SendClockFactor, RTCPTimeoutFactor, PTCPTakeoverTimeoutFactor, PTCPMasterStartupTime, SyncProperties, PTCP_MasterPriority1, PTCP_MasterPriority2, PTCPLengthSubdomainName, PTCPSubdomainName, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PDTimeData	BlockHeader, TimeMasterPriority1, TimeMasterPriority2, TimePLLWindow, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PdevData	BlockHeader, Padding, Padding, [PDIRData], [PDSyncData], [PDIRSubframeData], [PDTimeData]
PDRealData	MultipleBlockHeader, { [PDPortDataReal] ^b , [PDInterfaceMrpDataReal], [PDPortMrpDataReal], [PDPortFODDataReal] ^a , [PDInterfaceDataReal], [PDPortStatistic] } ^c ^a Il ne doit y avoir aucune information FiberOpticManufacturerSpecific ^b Les champs SlotNumber et SubslotNumber doivent être ignorés ^c Un MultipleBlockHeader est requis pour les données de chaque sous-module (interface ou port, par exemple)

Nom de substitution	Structure
PDExpectedData	MultipleBlockHeader, { [PDPortDataCheck]^a, [PDPortDataAdjust]^a, [PDInterfaceMrpDataAdjust], [PDInterfaceMrpDataCheck], [PDPortMrpDataAdjust], [PDPortFODataAdjust], [PDPortFODataCheck], [PDNCDataCheck], [PDInterfaceFSUDataAdjust], [PDInterfaceAdjust] }^b ^a Les champs SlotNumber et SubslotNumber doivent être ignorés ^b Un MultipleBlockHeader est requis pour les données de chaque sous-module (interface ou port, par exemple)
PDPortDataReal	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, LengthOwnPortName, OwnPortName, NumberOfPeers, [Padding*]^a, [LengthPeerPortName, PeerPortName, LengthPeerStationName, PeerStationName, [Padding*]^a, LineDelay, PeerMACAddress^b, [Padding*]^a]*, MAUType^c, [Padding*]^a, Reserved^d, RTClass3, PortStatus, MulticastBoundary, LinkState, [Padding*]^a, MediaType ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. ^b Ce champ contient l'adresse MAC de l'interface de l'homologue ^c Voir Tableau 552 ^d Le nombre d'octets réservés doit être 2.
PDInterfaceDataReal	BlockHeader, LengthOwnStationName, OwnStationName, [Padding*]^a, MACAddressValue^b, [Padding*]^a, IPParameterValue, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. ^b Ce champ contient l'adresse MAC de l'interface.
TxPortGroup	NumberOfTxPortGroups, TxPortGroupArray
SubstituteValue	BlockHeader, SubstitutionMode, SubstituteDataItem
RecordInputData-ObjectElement	BlockHeader, LengthIOCS, IOCS, LengthIOPS, IOPS, LengthData, Data Cas particulier: réponse pour un sous-module de sortie PNIOStatus(=0xDE, 0x80, 0xB0, 0x00)
RecordOutputData-ObjectElement	BlockHeader^c, SubstituteActiveFlag, LengthIOCS, LengthIOPS, LengthData, DataItem^a, SubstituteValue^b Cas particulier: réponse pour un sous-module de sortie PNIOStatus(=0xDE, 0x80, 0xB0, 0x00) ^a Pour cet enregistrement, DataItem doit être codé en IOCS, Data, IOPS ^b L'IOPS du SubstituteDataItem héritée par SubstituteValue contient les informations valides ^c BlockLength doit comprendre SubstituteValue

Nom de substitution	Structure
ARData avec BlockVersionLow = 0 Legacy Ne peut être utilisé qu'avec StartupMode:=0	BlockHeader, NumberOfARs, (ARUUID, ARType, ARProperties, CMInitiatorObjectUUID, StationNameLength, CMInitiatorStationName, NumberOfIOCRs, (IOCRType, IOCRProperties, FrameID, APDU_Status ^a , InitiatorUDPRTPort, ResponderUDPRTPort)*, AlarmCRTType, LocalAlarmReference, RemoteAlarmReference, ParameterServerObjectUUID ^b , StationNameLength ^c , [ParameterServerStationName], NumberOfAPIs, API*)* Cas particulier: "IOSAR avec ARProperties.DeviceAccess=1": NumberOfIOCRs := 0 AlarmCRTType := 0 NumberOfAPIs := 0 ^a APDU_Status.CycleCounter et APDU_Status.TransferStatus peuvent être nuls ^b La valeur NIL doit être attribuée au paramètre ParameterServerObjectUUID s'il n'est pas utilisé ^c La valeur 0 doit être attribuée à StationNameLength s'il n'est pas utilisé. Dans ce cas, le champ ParameterServerStationName doit être ignoré
ARData avec BlockVersionLow = 1	BlockHeader, NumberOfARs, (ARUUID, CMInitiatorObjectUUID ^b , ARProperties, ARType, AlarmCRTType, LocalAlarmReference, RemoteAlarmReference, InitiatorUDPRTPort, ResponderUDPRTPort, StationNameLength, CMInitiatorStationName, [Padding*] ^d , StationNameLength ^c , [ParameterServerStationName], [Padding*] ^d , NumberOfIOCRs, [Padding*] ^d , (IOCRProperties, IOCRType, FrameID, APDU_Status ^a)*, NumberOfAPIs, [Padding*] ^d , API*, ARDataInfo)* Cas particulier: "IOSAR avec ARProperties.DeviceAccess=1": NumberOfIOCRs := 0 AlarmCRTType := 0 NumberOfAPIs := 0 ^a APDU_Status.CycleCounter et APDU_Status.TransferStatus peuvent être nuls ^b La valeur NIL doit être attribuée au paramètre ParameterServerObjectUUID s'il n'est pas utilisé ^c La valeur 0 doit être attribuée à StationNameLength s'il n'est pas utilisé. Dans ce cas, le champ ParameterServerStationName doit être omis. ^d Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
ARDataInfo	NumberOfEntries, [Padding*] ^a , { [IRInfoBlock], [SRInfoBlock], [ARVendorBlockReq *], [ARVendorBlockRes *], [ARFSUBBlock], [SRLData] ^b } ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. ^b Le bloc SRLData doit être utilisé si un point d'accès au réseau redondant existe.
SRLData	BlockHeader, RedundancyInfo, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
APIData	BlockHeader, NumberOfAPIs, API*
LogBookData	BlockHeader, ActualLocalTimeStamp, NumberOfLogEntries, (LocalTimeStamp, ARUUID, PNIOStatus, EntryDetail)*
CMInitiatorObjectUUID	PROFINETIOConstantValue, InstanceHigh, InstanceLow, DeviceIdentNumber
ParameterServer-ObjectUUID	PROFINETIOConstantValue, InstanceHigh, InstanceLow, DeviceIdentNumber

Nom de substitution	Structure
DeviceIDentNumber	DeviceIDHigh, DeviceIDLow, VendorIDHigh, VendorIDLow
IMDataWrite	[I&M1] ^ [I&M2] ^ [I&M3] ^ [I&M4]
IMData	I&M0 ^ [I&M1] ^ [I&M2] ^ [I&M3] ^ [I&M4]
I&M0	BlockHeader, VendorIDHigh, VendorIDLow, OrderID, IM_Serial_Number, IM_Hardware_Revision, IM_Software_Revision, IM_Revision_Counter, IM_Profile_ID, IM_Profile_Specific_Type, IM_Version, IM_Supported
I&M1	BlockHeader, IM_Tag_Function, IM_Tag_Location
I&M2	BlockHeader, IM_Date
I&M3	BlockHeader, IM_Descriptor
I&M4	BlockHeader, IM_Signature
I&M0FilterData	{I&M0FilterDataSubmodule ^a , [I&M0FilterDataModule] ^b , I&M0FilterDataDevice ^c }
	^a Doit contenir tous les sous-modules avec IMData discret
	^b Si elle existe, doit contenir uniquement la référence du module
	^c Doit contenir la référence de l'appareil (par exemple le sous-module d'interface) qui doit prendre en charge l'écriture d'I&M1, d'I&M2, d'I&M3 et d'I&M4
I&M0FilterDataInfo	NumberOfAPIs, (API, NumberOfModules, (SlotNumber, ModuleIdentNumber, NumberOfSubmodules, (SubslotNumber, SubmoduleIdentNumber)*))*
I&M0FilterData-Submodule	BlockHeader, I&M0FilterDataInfo
I&M0FilterDataModule	BlockHeader, I&M0FilterDataInfo
I&M0FilterDataDevice	BlockHeader, I&M0FilterDataInfo
IM_Version	IM_Version_Major, IM_Version_Minor
IM_Software_Revision	SWRevisionPrefix, IM_SWRevision_Functional_Enhancement, IM_SWRevision_Bug_Fix, IM_SWRevision_Internal_Change
MultipleBlockHeader	BlockHeader, Padding, Padding, API, SlotNumber, SubslotNumber
PDIRGlobalData avec BlockVersionLow = 1	BlockHeader, Padding, Padding, IRDataUUID, MaxBridgeDelay, NumberOfPorts, (MaxPortTxDelay, MaxPortRxDelay)*
PDIRGlobalData avec BlockVersionLow = 2	BlockHeader, Padding, Padding, IRDataUUID, MaxBridgeDelay, NumberOfPorts, (MaxPortTxDelay, MaxPortRxDelay, MaxLineRxDelay, YellowTime)*
PDIRBeginEndData	BlockHeader, Padding, Padding, RedGuard, NumberOfPorts, (NumberOfAssignments, (TXBeginEndAssignment, RXBeginEndAssignment)*, NumberOfPhases, (TXPhaseAssignment, RXPhaseAssignment)*)*
RedGuard	StartOfRedFrameID, EndOfRedFrameID
TXBeginEndAssignment	RedOrangePeriodBegin, OrangePeriodBegin ^a , GreenPeriodBegin
	^a Doit être mis à GreenPeriodBegin
RXBeginEndAssignment	RedOrangePeriodBegin, OrangePeriodBegin ^a , GreenPeriodBegin ^b
	^a Doit être mis à GreenPeriodBegin
	^b GreenPeriodBegin doit être réglé à la fin de la période rouge.
TXPhaseAssignment	PhaseAssignment
RXPhaseAssignment	PhaseAssignment
PDIRFrameData avec BlockVersionLow = 0	BlockHeader, Padding, Padding, (FrameSendOffset, DataLength, ReductionRatio, Phase, FrameID, EtherType, RxPort, FrameDetails, TxPortGroup, [Padding*] ^a)*
	^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.

Nom de substitution	Structure
PDIRFrameData avec BlockVersionLow = 1	BlockHeader, Padding, Padding, FrameDataProperties, [(FrameSendOffset, DataLength, ReductionRatio, Phase, FrameID^{a b c}, EtherType, RxPort, FrameDetails, TxPortGroup, [Padding*]^d)*] ^a Ce bloc doit contenir tous les FrameIDs qui sont localement reçus (le nœud est la destination) ou injecté localement (le nœud est la source) dans le réseau. Il peut contenir les FrameID qui ne sont transférées que par le nœud. ^b Avec DFP: Ce bloc doit contenir deux entrées pour le même FrameID si ce nœud contient un diagramme d'états DFP_INBOUND ou DFP_OUTBOUND qui est un récepteur de cette trame. Une entrée contient le chemin de réception et l'autre le chemin de transmission. Si le chemin de transmission n'existe pas (le nœud consomme entièrement la trame sortante), l'entrée correspondante doit être sautée. ^c Avec MRPD: Un consommateur doit exister deux fois (FrameID et FrameID+1) et un fournisseur doit exister deux fois (FrameID et FrameID+1) dans le cas d'IOCR et peut exister plus de deux fois dans le cas de MCR. ^d Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
IsochronousModeData	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, ControllerApplicationCycleFactor, TimeDataCycle, TimeIOinput, TimeIOOutput, TimeIOInputValid, TimeIOOutputValid
PDPorDataAdjust	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, { [AdjustDomainBoundary], [AdjustMulticastBoundary], [AdjustMAUType], [AdjustLinkState], [AdjustPeerToPeerBoundary], [AdjustDCPBoundary], [AdjustPreambleLength] }
PDPorDataCheck	BlockHeader, Padding, Padding, SlotNumber, SubslotNumber, { [CheckPeers], [CheckLineDelay], [CheckMAUType], [CheckLinkState], [CheckSyncDifference], [CheckMAUTypeDifference] }
AdjustDomainBoundary with BlockVersionLow = 1	BlockHeader, Padding, Padding, DomainBoundaryIngress, DomainBoundaryEgress, AdjustProperties, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
AdjustMulticast-Boundary	BlockHeader, Padding, Padding, MulticastBoundary, AdjustProperties, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
AdjustMAUType	BlockHeader, Padding, Padding, MAUType, AdjustProperties
AdjustLinkState	BlockHeader, Padding, Padding, LinkState, AdjustProperties
AdjustPeerToPeer-Boundary	BlockHeader, Padding, Padding, PeerToPeerBoundary, AdjustProperties, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
AdjustDCPBoundary	BlockHeader, Padding, Padding, DCPBoundary, AdjustProperties, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
AdjustPreambleLength	BlockHeader, Padding, Padding, PreambleLength, AdjustProperties, [Padding*]^a ^a The number of padding octets shall be adapted to make the block Unsigned32 aligned.
CheckPeers	BlockHeader, NumberOfPeers, (LengthPeerPortName, PeerPortName, LengthPeerStationName, PeerStationName)*, [Padding*]^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.

Nom de substitution	Structure
CheckLineDelay	BlockHeader, Padding, Padding, LineDelay
CheckMAUType	BlockHeader, MAUType
CheckLinkState	BlockHeader, LinkState
CheckSyncDifference	BlockHeader, CheckSyncMode
CheckMAUType-Difference	BlockHeader, MAUTypeMode
PDInterfaceAdjust	BlockHeader, Padding, Padding, MultipleInterfaceMode, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PDInterface-MrpDataAdjust	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Role, [Padding*] ^a , MRP_LengthDomainName, MRP_DomainName, [Padding*] ^a , { [(MrpManagerParams) ^ (MrpClientParams)] } ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PDInterface-MrpDataReal with BlockVersionLow = 0	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Role, MRP_LengthDomainName, MRP_DomainName, [Padding*] ^a , MRP_Version, { [(MrpManagerParams) ^ (MrpClientParams)], [MrpRingStateData] }, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PDInterface-MrpDataReal with BlockVersionLow = 1	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Role, MRP_Version, MRP_LengthDomainName, MRP_DomainName, [Padding*] ^a , { [(MrpManagerParams) ^ (MrpClientParams)], [MrpRingStateData] }, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PDInterface-MrpDataCheck	BlockHeader, Padding, Padding, MRP_DomainUUID, MRP_Check
PDPortMrpDataAdjust	BlockHeader, Padding, Padding, MRP_DomainUUID
PDPortMrpDataReal	BlockHeader, Padding, Padding, MRP_DomainUUID
MrpManagerParams	BlockHeader, MRP_Prio, MRP_TOPchgT, MRP_TOPNRmax, MRP_TSTshortT, MRP_TSTdefaultT, MRP_TSTNRmax, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
MrpClientParams	BlockHeader, MRP_LNKdownT, MRP_LNKupT, MRP_LNKNRmax
MrpRingStateData	BlockHeader, MRP_RingState
PDPortFODataReal	BlockHeader, Padding, Padding, FiberOpticType, FiberOpticCableType, [FiberOpticManufacturerSpecific*] ^a , [FiberOpticDiagnosisInfo] ^a Uniquement utilisé par des nœuds d'une version originelle.
FiberOpticManufacturer Specific	BlockHeader, VendorIDHigh, VendorIDLow, VendorBlockType, Data*, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
FiberOpticDiagnosisInfo	BlockHeader, Padding, Padding, FiberOpticPowerBudgetReal, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.

Nom de substitution	Structure
FiberOpticPowerBudgetReal	FiberOpticPowerBudgetType ^a ^a FiberOpticPowerBudgetType.CheckEnable doit être mis à zéro. Il n'a pas de signification dans ce cas.
PDPortFODataAdjust	BlockHeader, Padding, Padding, FiberOpticType, FiberOpticCableType
PDPortFODataCheck	BlockHeader, Padding, Padding, MaintenanceRequiredPowerBudget, MaintenanceDemandedPowerBudget, ErrorPowerBudget
MaintenanceRequired-PowerBudget	FiberOpticPowerBudgetType
MaintenanceDemanded-PowerBudget	FiberOpticPowerBudgetType
ErrorPowerBudget	FiberOpticPowerBudgetType
PDNCDataCheck	BlockHeader, Padding, Padding, MaintenanceRequiredDropBudget, MaintenanceDemandedDropBudget, ErrorDropBudget
MaintenanceRequired-DropBudget	NCDropBudgetType
MaintenanceDemanded-DropBudget	NCDropBudgetType
ErrorDropBudget	NCDropBudgetType
PDPortStatistic	BlockHeader, Padding, Padding, ifInOctets, ifOutOctets, ifInDiscards, ifOutDiscards, ifInErrors, ifOutErrors, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
PDInterface-FSUDataAdjust	BlockHeader, [Padding*] ^a , { [FSHelloBlock], [FastStartUpBlock] } ^b ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. ^b Il doit exister au moins un bloc facultatif.
ARFSUDataAdjust	BlockHeader, [Padding*] ^a , { [FSPParameterBlock], [FastStartUpBlock] } ^b ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. ^b Il doit exister au moins un bloc facultatif.
FSHelloBlock	BlockHeader, [Padding*] ^a , FSHelloMode, FSHelloInterval, FSHelloRetry, FSHelloDelay, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
FSPParameterBlock	BlockHeader, [Padding*] ^a , FSPParameterMode, FSPParameterUUID, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
FastStartUpBlock	BlockHeader, [Padding*] ^a , Data*, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
AutoConfiguration	BlockHeader, [Padding*] ^a , ACCommunication, ACConfiguration, [ACIsochronous], [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.

Nom de substitution	Structure
ACCommunication	BlockHeader, [Padding*] ^a , ACMinDeviceInterval, ACCommunicationProperties, SendClockFactorArray, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
SubmoduleInputDataLength	SubmoduleDataLength
SubmoduleOutputDataLength	SubmoduleDataLength
ACConfiguration	BlockHeader, [Padding*] ^a , (API, SlotNumber, SubslotNumber, ModuleIdentNumber, SubmoduleIdentNumber, SubmoduleInputDataLength, SubmoduleOutputDataLength)* ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
ACIsochronous	BlockHeader, [Padding*] ^a , (API, SlotNumber, SubslotNumber, TimeIOBase ^b , TimeIOInput ^c , TimeIOOutput ^d)* ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32. ^b Valeur minimale conforme au mot-clés de GSDML T_IO_Base ^c Valeur minimale conforme au mot-clés de GSDML T_IO_InputMin ^d Valeur minimale conforme au mot-clés de GSDML T_IO_OutputMin
CombinedObject-Container	{ COContainerContent* }
COContainerContent	BlockHeader, Padding, Padding, API, Slot, Subslot, Padding, Padding, Index, COContainerBlock, [Padding*] ^a ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.
COContainerBlock	Data* Cas particulier "PhysicalDevice" (le point d'accès de l'objet combiné est le sous-module de l'interface): [PDPortDataAdjust] ^ [PDPortDataCheck] ^ [PDIRData] ^ [PDSyncData] ^ [PDInterfaceMrpDataAdjust] ^ [PDInterfaceMrpDataCheck] ^ [PDPortMrpDataAdjust] ^ [PDPortFODDataAdjust] ^ [PDPortFODDataCheck] ^ [PDNCDataCheck] ^ [PDInterfaceFSUDDataAdjust] ^ [PDIRSubframeData] ^ [PDInterfaceAdjust] ^ [PDTimeData]
UploadBLOBQuery	BlockHeader, Padding, Padding, FromOffsetData
UploadBLOB	BlockHeader, Padding, Padding, FromOffsetData, NextOffsetData, TotalSize, Data*
NestedDiagnosisInfo	BlockHeader, NumberOfEntries, (IPAddress, StationNameLength, NameOfStationValue, [Padding*] ^a)* ^a Le nombre d'octets de bourrage doit être adapté de manière à aligner le bloc Unsigned32.

5.2 Syntaxe de transfert

5.2.1 Section de codage relative aux champs spécifiques du BlockHeader

5.2.1.1 Codage du champ BlockType

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 358. Ce champ identifie le PDU ou le type de bloc de données.

Tableau 358 – BlockType

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0000	Réservé	—
0x0001	AlarmNotification High	RTA-SDU.AlarmNotification-PDU
0x0002	AlarmNotification Low	RTA-SDU.AlarmNotification-PDU
0x0003 – 0x0007	Réservé	—
0x0008	IODWriteReqHeader	PROFINETIO-ServiceReqPDU.IODWriteReq
0x0009	IODReadReqHeader	PROFINETIO-ServiceReqPDU.IODReadReq
0x000A – 0x000F	Réservé	—
0x0010	DiagnosisData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. DiagnosisData RTA-SDU.AlarmNotification-PDU.AlarmPayload. DiagnosisItem.DiagnosisData
0x0011	Réservé	—
0x0012	ExpectedIdentificationData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. ExpectedIdentificationData
0x0013	RealIdentificationData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. RealIdentificationData
0x0014	SubstituteValue	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. SubstituteValue PROFINETIO-ServiceReqPDU. IODWriteReq.RecordDataWrite. SubstituteValue
0x0015	RecordInputDataObjectElement	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. RecordInputDataObjectElement
0x0016	RecordOutputDataObjectElement	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. RecordOutputDataObjectElement
0x0017	Réservé	—
0x0018	ARData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. ARData
0x0019	LogBookData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. LogBookData
0x001A	APIData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. APIData
0x001B	SRLData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. ARDataInfo.SRLData
0x001C – 0x001F	Réservé	—
0x0020	I&M0	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IMData.I&M0

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0021	I&M1	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IMData.I&M1 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. IMDataWrite.I&M1
0x0022	I&M2	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IMData.I&M2 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. IMDataWrite.I&M2
0x0023	I&M3	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IMData.I&M3 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. IMDataWrite.I&M3
0x0024	I&M4	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IMData.I&M4 PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. IMDataWrite.I&M4
0x0025 – 0x002F	I&M5 – 15 ^a ^a Réserve aux données supplémentaires d'identification et de maintenance	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite
0x0030	I&M0FilterDataSubmodule	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. I&M0FilterData.I&M0FilterDataSubmodule
0x0031	I&M0FilterDataModule	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. I&M0FilterData.I&M0FilterDataModule
0x0032	I&M0FilterDataDevice	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. I&M0FilterData.I&M0FilterDataDevice
0x0033 – 0x0100	Réserve	—
0x0101	ARBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.ARBlockReq
0x0102	IOCRBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.IOCRBlockReq
0x0103	AlarmCRBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.AlarmCRBlockReq
0x0104	ExpectedSubmoduleBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq. ExpectedSubmoduleBlockReq
0x0105	PrmServerBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.PrmServerBlockReq
0x0106	MCRBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.MCRBlockReq
0x0107	ARRPCBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.ARRPCBlockReq
0x0108	ARVendorBlockReq	PROFINETIO-ServiceReqPDU.IODConnectReq.ARVendorBlockReq

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0109	IRInfoBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.IRInfoBlock
0x010A	SRInfoBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.SRInfoBlock
0x010B	ARFSUBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.ARFSUBlock
0x010C – 0x010F	Réservé	—
0x0110	IODBlockReq, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceReqPDU.IODControlReq.ControlBlockConnect (PrmEnd.req)
0x0111	IODBlockReq, doit uniquement être utilisé conjointement avec un événement d'alarme de connexion	PROFINETIO-ServiceReqPDU.IODControlReq.ControlBlockPlug (PrmEnd.req)
0x0112	IOXBlockReq, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion ou une séquence PrmBegin/PrmEnd	PROFINETIO-ServiceReqPDU.IOXControlReq.ControlBlockConnect (ApplicationReady.req)
0x0113	IOXBlockReq, doit uniquement être utilisé conjointement avec un événement d'alarme de connexion	PROFINETIO-ServiceReqPDU.IOXControlReq.ControlBlockPlug (ApplicationReady.req)
0x0114	ReleaseBlockReq	PROFINETIO-ServiceReqPDU.IODReleaseReq.ReleaseBlock
0x0115	Réservé	—
0x0116	IOXBlockReq, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceReqPDU.IOXControlReq.ControlBlockConnect (ReadyForCompanion.req)
0x0117	IOXBlockReq, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceReqPDU.IOXControlReq.ControlBlockConnect (ReadyForRT_CLASS_3.req)
0x0118	IODBlockReq	PROFINETIO-ServiceReqPDU.IODControlReq.ControlBlockConnect (PrmBegin.req)
0x0119	SubmoduleListBlock	PROFINETIO-ServiceReqPDU.IODControlReq.SubmoduleListBlock (PrmBegin.req)
0x011A – 0x011F	Réservé	—
0x0200	PDPorDataCheck	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataCheck PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataCheck
0x0201	PdevData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PdevData
0x0202	PDPorDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataAdjust

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0203	PDSyncData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDSyncData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDSyncData
0x0204	IsochronousModeData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. IsochronousModeData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. IsochronousModeData
0x0205	PDIRData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRData
0x0206	PDIRGlobalData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRGlobalData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRGlobalData
0x0207	PDIRFrameData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRFrameData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRFrameData
0x0208	PDIRBeginEndData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDIRBeginEndData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDIRBeginEndData
0x0209	AdjustDomainBoundary Sous-bloc d'ajustement de DomainBoundary	PROFINETIO-ServiceReqPDU.IODReadReq. RecordDataRead. PDPortDataAdjust.AdjustDomainBoundary PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataAdjust. AdjustDomainBoundary
0x020A	Sous-bloc de vérification de Peers	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckPeers PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataAdjust.CheckPeers
0x020B	Sous-bloc de vérification de LineDelay	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckLineDelay PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataCheck.CheckLineDelay

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x020C	Sous-bloc de vérification de MAUType	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataCheck.CheckMAUType PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataCheck.CheckMAUType
0x020E	AdjustMAUType Sous-bloc d'ajustement de MAUType	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustMAUType PROFINETIO-ServiceReqPDU.IODWriteReq. RecordDataWrite. PDPortDataAdjust.AdjustMAUType
0x020F	PDPortDataReal	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataReal PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortDataReal
0x0210	AdjustMulticastBoundary Sous-bloc d'ajustement de MulticastBoundary	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustMulticastBoundary PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.AdjustMulticastBoundary
0x0211	PDInterfaceMrpDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataAdjust
0x0212	PDInterfaceMrpDataReal	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDInterfaceMrpDataReal
0x0213	PDInterfaceMrpDataCheck	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataCheck PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataCheck PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDInterfaceMrpDataCheck
0x0214	PDPortMrpDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortMrpDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortMrpDataAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDPortMrpDataAdjust

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0215	PDPorMrpDataReal	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPorMrpDataReal PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPorMrpDataReal
0x0216	MrpManagerParams Sous-bloc des paramètres du gestionnaire de redondance de support	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataAdjust.MrpManagerParams PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataAdjust.MrpManagerParams PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal.MrpManagerParams
0x0217	MrpClientParams Sous-bloc des paramètres client de redondance de support	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataAdjust.MrpClientParams PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceMrpDataAdjust.MrpClientParams PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal.MrpClientParams
0x0219	MrpRingStateData Sous bloc des données d'état de l'anneau de redondance de support	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceMrpDataReal.MrpRingStateData
0x021B	AdjustLinkState Sous-bloc d'ajustement de LinkState	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataAdjust.AdjustLinkState PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataAdjust.AdjustLinkState
0x021C	CheckLinkState Sous-bloc de vérification de LinkState	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataCheck.CheckLinkState PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataAdjust.CheckLinkState
0x021E	CheckSyncDifference Sous-bloc du CableDelay local et distant détecté par LLDP pour reconnaître une différence de synchronisation	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataCheck.CheckSyncDifference PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataCheck.CheckSyncDifference
0x021F	CheckMAUTypeDifference Sous-bloc de vérification de MAUTypes local et distant détecté par LLDP	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPorDataCheck.CheckMAUTypeDifference PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPorDataCheck.CheckMAUTypeDifference
0x0220	PDPorFODDataReal	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDPorFODDataReal PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPorFODDataReal

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0221	FiberOpticManufacturerSpecific Sous-bloc de lecture des données à fibre optique réelles spécifiques au fabricant	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataReal.FiberOpticManufacturerSpecific PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortFODataReal.FiberOpticManufacturerSpecific
0x0222	PDPortFODataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortFODataAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDPortFODataAdjust
0x0223	PDPortFODataCheck	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataCheck PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortFODataCheck PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDPortFODataCheck
0x0224	AdjustPeerToPeerBoundary Sous-bloc d'ajustement de la frontière entre homologues	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustPeerToPeerBoundary PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.AdjustPeerToPeerBoundary
0x0225	AdjustDCPBoundary Sous-bloc d'ajustement de la frontière DCP	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustDCPBoundary PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.AdjustDCPBoundary
0x0226	AdjustPreambleLength Sous-bloc d'ajustement de la longueur de préambule utilisée	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortDataAdjust.AdjustPreambleLength PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDPortDataAdjust.AdjustPreambleLength
0x0227	Réservé	—
0x0228	SFiberOpticDiagnosisInfo Sous-bloc pour la lecture des données de diagnostic de fibre optique réelles	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortFODataReal.FiberOpticDiagnosisInfo PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.FiberOpticDiagnosisInfo
0x0229	Réservé	—

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x022A	PDIRSubframeData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDIRSubframeData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.PDIRSubframeData PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PdevData.PDIRSubframeData
0x022B	SubframeBlock Sous-bloc pour la partie persistante de DFP	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDIRSubframeData.SubframeBlock PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.PDIRSubframeData.SubframeBlock PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PdevData.PDIRSubframeData.SubframeBlock
0x022C	Réservé	—
0x022D	PDTimeData	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.COContainerContent.COContainerBlock.PDTimeData PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.COContainerContent.COContainerBlock.PDTimeData
0x022E – 0x022F	Réservé	—
0x0230	PDNCDataCheck	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDNCDataCheck PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.PDNCDataCheck PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDExpectedData.PDNCDataCheck
0x0231 – 0x023F	Réservé	—
0x0240	PDInterfaceDataReal	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDInterfaceDataReal PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDRealData.PDInterfaceDataReal
0x0241 – 0x024F	Réservé	—
0x0250	PDInterfaceAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDInterfaceAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite.PDInterfaceAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead.PDRealData.PDInterfaceDataAdjust

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0251	PDPortStatistic	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDPortStatistic PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.PDPortStatistic
0x0252 – 0x03FF	Réservé	—
0x0400	MultipleBlockHeader	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDRealData.MultipleBlockHeader PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.MultipleBlockHeader
0x0401	COContainerContent	PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. CombinedObjectContainer.COContainerContent
0x0402 – 0x04FF	Réservé	—
0x0500	RecordDataReadQuery	PROFINETIO-ServiceResPDU.IODReadReq.RecordDataReadQuery
0x0501 – 0x05FF	Réservé	—
0x0600	FSHelloBlock	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceFSUDDataAdjust.FSHelloBlock PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceFSUDDataAdjust.FSHelloBlock PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDInterfaceFSUDDataAdjust.FSHelloBlock
0x0601	FSPParameterBlock	PROFINETIO-ServiceReqPDU.IODConnectReq.IOCRBlockReq. ARFSUBlock.ARFSUDDataAdjust.FSPParameterBlock
0x0602 – 0x607	Réserve à FastStartUp	—
0x0608	PDInterfaceFSUDDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDInterfaceFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. PDInterfaceFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. PDExpectedData.PDInterfaceFSUDDataAdjust
0x0609	ARFSUDDataAdjust	PROFINETIO-ServiceReqPDU.IODReadReq.RecordDataRead. ARFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODWriteReq.RecordDataWrite. ARFSUDDataAdjust PROFINETIO-ServiceReqPDU.IODConnectReq.ARFSUBlock. ARFSUDDataAdjust
0x060A – 0x60F	Réservé à FastStartUp	—
0x0610 – 0x06FF	Réservé	—

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x0700	AutoConfiguration Données d'auto configuration	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration
0x0701	AutoConfiguration Données de communication d'auto configuration	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration.ACCommunication
0x0702	AutoConfiguration Données de configuration d'auto configuration	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration.ACConfiguration
0x0703	AutoConfiguration Données isochrones d'auto configuration	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead. AutoConfiguration.ACIsynchronous
0x0704 – 0x07FF	Réservé	—
0x0800	Réservé aux profils relatifs aux économies d'énergie BlockType pour un service de demande	PROFINETIO- ServiceReqPDU.IODWriteReq.RecordDataWrite
0x0801	Réservé aux profils relatifs aux économies d'énergie BlockType pour un service de réponse	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead
0x0800 – 0x08FF	Réservé aux profils relatifs aux économies d'énergie	—
0x0900 – 0x09FF	Réservé à des profils relatifs à une séquence d'événements	—
0x0A00	Communication entre contrôleurs Télécharger la demande BLOB Query	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead.Uplo adBLOBQuery
0x0A01	Communication entre contrôleurs Télécharger BLOB	PROFINETIO- ServiceResPDU.IODReadRes.RecordDataRead.Uplo adBLOB
0x0A02	Communication entre contrôleurs Information de diagnostic imbriquée	PROFINETIO- ServiceReqPDU.IODReadReq.RecordDataRead.Nest edDiagnosisInfo
0x0A03 – 0x0AFF	Réservé à des profils relatifs à la communication entre contrôleurs	—
0x0B00 – 0x0EFF	Réserve	—
0x0F00	MaintenanceltItem	RTA-SDU.Alarmnotification- PDU.AlarmPayload.MaintenanceltItem
0x0F01	Charger les enregistrements sélectionnés dans Upload&RetrievalItem	RTA-SDU.Alarmnotification- PDU.AlarmPayload.Upload&RetrievalItem
0x0F02	iParameterItem	RTA-SDU.Alarmnotification- PDU.AlarmPayload.iParameterItem
0x0F03	Extraire les enregistrements sélectionnés dans Upload&RetrievalItem	RTA-SDU.Alarmnotification- PDU.AlarmPayload.Upload&RetrievalItem
0x0F04	Extraire tous les enregistrements stockés dans Upload&RetrievalItem	RTA-SDU.Alarmnotification- PDU.AlarmPayload.Upload&RetrievalItem
0x0F05 – 0x6FFF	Réservé	—
0x7000 – 0x7FFF	Réservé pour les objets de données spécifiques à l'utilisateur	—
0x8000	Réservé	—
0x8001	Alarm Ack High	RTA-SDU.AlarmAck-PDU
0x8002	Alarm Ack Low	RTA-SDU.AlarmAck-PDU
0x8008	IODWriteResHeader	PROFINETIO-ServiceResPDU.IODWriteRes

Valeur (hexadécimale)	Utilisé pour	Utilisé par
0x8009	IODReadResHeader	PROFINETIO-ServiceReqPDU.IODReadRes
0x800A – 0x8100	Réservé	—
0x8101	ARBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARBlockRes
0x8102	IOCRBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.IOCRBlockRes
0x8103	AlarmCRBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.AlarmCRBlockRes
0x8104	ModuleDiffBlock	PROFINETIO-ServiceResPDU.IODConnectRes.ModuleDiffBlock PROFINETIO-ServiceResPDU.IODWriteRes.RecordDataRead.ModuleDiffBlock PROFINETIO-ServiceReqPDU.IOXControlReq.ModuleDiffBlock
0x8105	PrmServerBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.PrmServerBlockRes
0x8106	ARServerBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARServerBlockRes
0x8107	ARRPCBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARRPCBlockRes
0x8108	ARVendorBlockRes	PROFINETIO-ServiceResPDU.IODConnectRes.ARVendorBlockRes
0x8109 – 0x810F	Réservé	—
0x8110	IODBlockRes, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceResPDU.IODControlRes.ControlBlockConnect (Prm End.rsp)
0x8111	IODBlockRes, doit uniquement être utilisé conjointement avec un événement d'alarme de connexion	PROFINETIO-ServiceResPDU.IODControlRes.ControlBlockPlug (Prm End.rsp)
0x8112	IOXBlockRes, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockConnect (Application Ready.rsp)
0x8113	IOXBlockRes, doit uniquement être utilisé conjointement avec un événement d'alarme de connexion	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockPlug (Application Ready.rsp)
0x8114	ReleaseBlockRes	PROFINETIO-ServiceResPDU.IODReleaseRes.ReleaseBlock
0x8115	Réservé	—
0x8116	IOXBlockRes, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockConnect (ReadyForCompanion.rsp)
0x8117	IOXBlockRes, doit uniquement être utilisé conjointement avec la phase d'établissement de connexion	PROFINETIO-ServiceResPDU.IOXControlRes.ControlBlockConnect (ReadyForRT_CLASS_3.rsp)
0x8118	IODBlockRes	PROFINETIO-ServiceResPDU.IODControlRes.ControlBlockConnect (PrmBegin.rsp)
Autre	Réservé	—

5.2.1.2 Codage du champ BlockLength

Ce champ doit être codé en type de données Unsigned16. Il doit contenir le nombre d'octets sans compter les champs BlockType et BlockLength.

5.2.1.3 Codage du champ BlockVersionHigh

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 359.

Tableau 359 – BlockVersionHigh

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	—
0x01	Version 1	Indique Version 1
0x02 – 0xFF	Réservé	—

5.2.1.4 Codage du champ BlockVersionLow

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 360.

Tableau 360 – BlockVersionLow

Valeur (hexadécimale)	Signification	Utilisation
0x00	Version 0	Indique la version 0
0x01	Version 1	Indique la version 1
0x02 – 0xFF	Version 2 à version 255	Indique la version 2 à la version 255

5.2.2 Section de codage relative à des champs spécifiques à RTA-SDU

5.2.2.1 Codage du champ AlarmType

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 361.

Tableau 361 – AlarmType

Valeur (hexadécimale)	Signification	Attaché à l'ASE de Diagnostic	USI structured AlarmPayload	AlarmPayload required	Alarm priority ^e
0x0000	Réservé	—	—	—	—
0x0001	Diagnostic	Oui	Oui	Oui ^d	Faible
0x0002	Processus	Non	Oui	Facultatif	Elevée
0x0003	Extraction ^a	Non	Oui	Facultatif	Faible
0x0004	Connexion	Non	Oui	Facultatif	Faible
0x0005	Status	Non	Recommandé	Facultatif	Faible
0x0006	Mise à jour	Non	Recommandé	Facultatif	Faible
0x0007	Redondance de support	Oui	Oui	Oui ^d	Faible
0x0008	Contrôlé par le superviseur	Non	Oui	Facultatif	Faible

Valeur (hexadécimale)	Signification	Attaché à l'ASE de Diagnostic	USI structured AlarmPayload	AlarmPayload required	Alarm priority ^e
	"Extraction" logique d'un sous-module pour rejeter la propriété				
0x0009	Libéré "Connexion" logique d'un sous-module pour retourner la propriété ou déclencher un reparamétrage	Non	Oui	Facultatif	Faible
0x000A	Sous-module incorrect de connexion ^c	Non	Oui	Facultatif	Faible
0x000B	Renvoi du sous-module ^f	Non	Oui	Facultatif	Faible
0x000C	Diagnostic disparaît	Oui	Oui	Facultatif	Faible
0x000D	Notification de discordance de communication multidiffusion	Oui	Oui	Oui ^d	Faible
0x000E	Notification de changement de données du port	Oui	Oui	Oui ^d	Faible
0x000F	Notification de données de synchronisation modifiées	Oui	Oui	Oui ^d	Faible
0x0010	Notification de problème de mode isochrone	Oui	Oui	Oui ^d	Faible
0x0011	Notification de problème lié au composant de réseau	Oui	Oui	Oui ^d	Faible
0x0012	Notification de données temporelles modifiées	Oui	Oui	Oui ^d	Faible
0x0013	Notification de problème de Combinaison de Trame Dynamique	Oui	Oui	Oui ^d	Faible
0x0014	Notification de problème de MRPD	Oui	Oui	Oui ^d	Faible
0x0015	Notification de Redondance du Système	Oui	Oui	Oui ^d	Faible
0x0016	Notification de discordance des multiples interfaces	Oui	Oui	Oui ^d	Faible
0x0017 – 0x001D	Réserve	—	—	—	—
0x001E	Notification de chargement et d'extraction	Non	Oui	Oui	Faible
0x001F	Pull module ^b	Non	Oui	Facultatif	Faible
0x0020 – 0x007F	Spécifique à un fabricant	Non	Recommandé	Facultatif	Faible
0x0080 – 0x00FF	Réservé aux profils	Non	OUI	Facultatif	Faible
0x0100 – 0xFFFF	Réservé	—	—	—	—

^a Avec ARProperties.PullModuleAlarmAllowed(=0) le numéro de sous-emplacement 0x0001 – 0x8FFF est utilisé en tant que "sous-module d'extraction" et le numéro de sous-emplacement 0 est utilisé en tant que "module d'extraction".

^b Avec ARProperties.PullModuleAlarmAllowed(=1) AlarmType(Pull) doit signaler l'extraction du sous-module et AlarmType(Pull module) doit signaler l'extraction du module.

^c Seulement une information pour l'application de CMCTL.

^d Un AlarmPayload peut ne pas exister en cas de disparition de diagnostic.

^e Affectation recommandée.

^f Seul cet AlarmType est utilisé pour les sous-modules SharedInput, ici aussi.

5.2.2.2 Codage du champ AlarmSpecifie

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 10: AlarmSpecifie.SequenceNumber

Ce champ doit être codé conformément à Tableau 362. Grâce à SequenceNumber, le destinataire détecte les doublons et répercute la valeur du champ dans AlarmAck.

Chaque priorité d'alarme (HIGH ou LOW) administre son propre SequenceNumber. Il convient qu'elle commence par zéro et qu'elle soit incrémentée à chaque AlarmNotification. Le récepteur d'alarme doit accepter une valeur arbitraire comme valeur de départ.

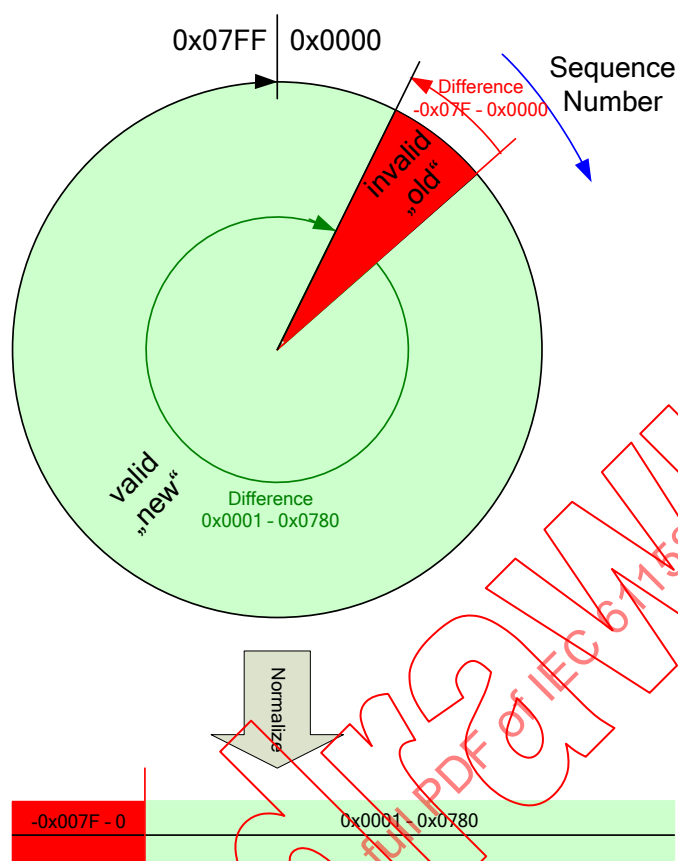
NOTE Le SequenceNumber peut commencer pour chaque AR d'un ensemble d'AR par chaque basculement avec une valeur arbitraire.

Dès qu'une alarme est répétée sur différentes AR d'un ARset, le même SequenceNumber doit être utilisé.

Pour la notification de la redondance du système, le SequenceNumber n'a pas de signification et doit être réglé à zéro.

Tableau 362 – AlarmSpecifie.SequenceNumber

Valeur (hexadécimale)	Signification
0x0000 – 0x07FF	Plage de valeurs admises



Légende

Anglais	Français
Sequence Number	Numéro de séquence
Difference 0x07F – 0x0000	Différence 0x07F – 0x0000
Invalid "old"	"Ancien" non valide
Valid "new"	"Nouveau" valide
Normalize	Normaliser

Figure 81 – Plage de valeurs AlarmSpecifie.SequenceNumber

Le Tableau 363 définit une fenêtre SequenceNumber maximale entre une ancienne et une nouvelle alarme.

Tableau 363 – Différence AlarmSpecifie.SequenceNumber

Valeur (hexadécimale)	Signification	Utilisation
-0x007F – 0x0000	Domaine rouge conformément à la Figure 81 1/16 de la plage SequenceNumber La destination de l'alarme considère l'alarme reçue comme ancienne	Alarme rejetée
0x0001 – 0x0780	Domaine vert conformément à la Figure 81 15/16 de la plage SequenceNumber La destination de l'alarme considère l'alarme reçue comme nouvelle	Alarme traitée

Bit 11: AlarmSpecifier.ChannelDiagnosis

Ce champ doit être codé avec les valeurs conformes au Tableau 364. Pour tous les autres AlarmTypes, ce champ doit être nul.

Tableau 364 – AlarmSpecifier.ChannelDiagnosis

Valeur (hexadécimale)	Signification	Utilisation dans AlarmType
0x00	Signifie que le sous-module (source de l'alarme) ne contient aucun ChannelDiagnosis, ExtChannelDiagnosis ni QualifiedChannelDiagnosis comportant Channel-Properties.Maintenance (=diagnosis) et Channel-Properties.Specifier (=appear)	Tous les AlarmTypes rattachés à ASE de Diagnostic
0x01	Signifie que le sous-module (source de l'alarme) contient au moins un ChannelDiagnosis, un ExtChannelDiagnosis ou un QualifiedChannelDiagnosis comportant Channel-Properties.Maintenance (=diagnosis) et Channel-Properties.Specifier (=appear)	

Bit 12: AlarmSpecifier.ManufacturerSpecificDiagnosis

Ce champ doit être codé avec les valeurs conformes au Tableau 365. Pour tous les autres AlarmTypes, ce champ doit être nul.

Tableau 365 – AlarmSpecifier.ManufacturerSpecificDiagnosis

Valeur (hexadécimale)	Signification	Utilisation dans AlarmType
0x00	Signifie que le sous-module (source de l'alarme) ne contient aucun ManufacturerSpecificDiagnosis comportant ChannelProperties.Maintenance (=diagnosis) et ChannelProperties.Specifier (=appear ou disappear)	Tous les AlarmTypes rattachés à ASE de Diagnostic
0x01	Signifie que le sous-module (source de l'alarme) contient au moins un ManufacturerSpecificDiagnosis comportant ChannelProperties.Maintenance (=diagnosis) et ChannelProperties.Specifier (=appear ou disappear)	

Bit 13: AlarmSpecifier.SubmoduleDiagnosisState

Ce champ doit être codé avec les valeurs conformes au Tableau 366. Pour tous les autres AlarmTypes, ce champ doit être nul.

Tableau 366 – AlarmSpecifier.SubmoduleDiagnosisState

Valeur (hexadécimale)	Signification	Utilisation dans AlarmType
0x00	Exempt d'erreur Il n'existe aucun DiagnosisData avec Channel-Properties.Maintenance (=diagnosis) et Channel-Properties.Specifier (=appear) au niveau du sous-module. De plus, il indique que tous les diagnostics signalés ont été supprimés. Une notification "disappears" individuelle peut être ignorée. Toutefois, même dans ce cas, un diagnostic spécifique au fabricant avec ChannelProperties.Maintenance (=diagnosis) et ChannelProperties.Specifier (=disappear) peut être présent.	Tous les AlarmTypes rattachés à ASE de Diagnostic
0x01	Il existe au moins un DiagnosisData avec Channel-Properties.Maintenance (=diagnosis) et Channel-Properties.Specifier (=appear) au niveau du sous-module.	

Bit 14: AlarmSpecifier.reserved

Ce champ doit être mis à zéro.

Bit 15: AlarmSpecifier.ARDiagnosisState

Ce champ doit être codé avec les valeurs conformes au Tableau 367. Pour tous les autres AlarmTypes, ce champ doit être nul.

Tableau 367 – AlarmSpecifier.ARDiagnosisState

Valeur (hexadécimale)	Signification	Utilisation dans AlarmType
0x00	Exempt d'erreur Il n'existe aucun DiagnosisData avec Channel-Properties.Maintenance (=diagnosis) et Channel-Properties.Specifier (=appear) au niveau de la relation entre applications. De plus, il indique que tous les diagnostics signalés ont été supprimés. Une notification "disappears" individuelle peut être ignorée. Toutefois, même dans ce cas, un diagnostic spécifique au fabricant avec ChannelProperties.Maintenance (=diagnosis) et ChannelProperties.Specifier (=disappear) peut être présent.	Tous les AlarmTypes rattachés à ASE de Diagnostic
0x01	Il existe au moins un DiagnosisData avec Channel-Properties.Maintenance (=diagnosis) et Channel-Properties.Specifier (=appear) au niveau de la relation entre applications.	

5.2.3 Section de codage liée aux champs d'adresse communs**5.2.3.1 Codage du champ API**

Ce champ, défini sous forme d'un numéro administratif, doit être codé en type de données Unsigned32 conformément au Tableau 368.

Tableau 368 – API

Valeur (hexadécimale)	Signification de l'API
0	Par défaut
0x00000001 – 0x0000FFFF	Utilisé pour les définitions de profil correspondant à 5.2.7.8
0x00010000 – 0xFFFFFFFF	Utilisé pour les définitions de profil au-delà de 5.2.7.8

NOTE API est l'acronyme d'Application Process Identifier (Identifiant de processus d'application)

5.2.3.2 Codage du champ SlotNumber

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 369.

Tableau 369 – SlotNumber

Valeur (hexadécimale)	Signification de SlotNumber
0 – 0x7FFF	Le premier emplacement utilisable des modules est zéro. Le dernier emplacement utilisable des modules est 0x7FFF. Peut contenir des espaces
0x8000 – 0xFFFF	Réservé

5.2.3.3 Codage du champ SubslotNumber

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 370.

Tableau 370 – SubslotNumber

Valeur (hexadécimale)	Signification de SubslotNumber
0x0000	ARProperties.PullModuleAlarmAllowed (=0) Doit être utilisé pour coder "Pull module" conjointement avec AlarmType (Pull).
	ARProperties.PullModuleAlarmAllowed (=1) Sous-emplacement utilisable. Le premier sous-emplacement utilisable des sous-modules est zéro.
0x0001 – 0x7FFF	Le dernier sous-emplacement utilisable des sous-modules est 0x7FFF. Peut contenir des espaces.
0x8000 – 0x8FFF	Par défaut Utilisé pour 16 sous-modules d'interface avec 255 ports au maximum. Peut contenir des espaces. Voir Formule (45)
	Si la redondance du système l'utilise, le sous-module d'interface est monté sur la gauche ou en bas à l'intérieur de la baie.
0x9000 – 0x9FFF	Utilisé pour 16 sous-modules d'interface avec 255 ports au maximum. Peut contenir des espaces. Voir Formule (45)
	Utilisé par la redondance du système avec une interface redondante pour le sous-module d'interface monté sur la droite ou en haut à l'intérieur de la baie.
0xA000 – 0xFFFF	Réserve

SubslotNumber doit être évalué avec 0xLIPP et les interfaces de comptage I (PP:= 00 signifie l'interface elle-même) et les ports de comptage PP comme présenté dans la Formule (45).

$$\text{SubslotNumber} = 0xL000 + 0x0I00 + 0x00PP \quad (45)$$

où

SubslotNumber	est la valeur correspondante
L	est la valeur de base pour cette formule 0x8000 or 0x9000
I	est le numéro de l'interface commençant par zéro
PP	est le numéro du port de l'interface commençant par un. Une valeur nulle signifie qu'il n'y a pas de port.

5.2.3.4 Codage du champ Index

5.2.3.4.1 Généralités

Ce champ doit être codé en type de données Unsigned16. La plage comprise entre zéro et 0x7FFF doit être utilisée pour adresser des objets de données d'enregistrement spécifiques à l'utilisateur. La plage comprise entre 0x8000 et 0xFFFF doit être utilisée pour la fonction spécifique au protocole ou d'autres extensions de protocole. La plage comprise entre 0x8000 et 0xFFFF est divisée en quatre sections. Chaque section permet de définir si les champs ARUUID, SlotNumber et SubslotNumber doivent être ignorés ou utilisés. Le champ facultatif TargetARUUID doit uniquement être présent et évalué conjointement avec les indices définis.

De plus, tous les services avec accès en écriture doivent uniquement utiliser l'espace d'adresses valide du contexte AR établi. Les services Read peuvent adresser des données d'enregistrement hors du cadre de ce contexte.

5.2.3.4.2 Utilisation des paramètres d'adresse

Les expressions ci-dessous illustrent l'évaluation des paramètres d'adresse en fonction de la plage d'index.

Les règles pour évaluer le paramètre address présentées dans le Tableau 371, le Tableau 372, le Tableau 373, le Tableau 374 et le Tableau 375 doivent être appliquées.

NOTE "Implicite" signifie une AR existant de l'extérieur. "Explicite" signifie l'AR existant de l'intérieur.

Tableau 371 – Expression 1 (spécifique au sous-emplacement)

RPCOperationNmb	ARUUID	ARUUID cible	Validité
Lecture implicite	NIL (AR implicite)	Évalué si requis par l'index	API, Numéro d'Emplacement, Numéro de Sous-emplacement, l'Index doit être évalué
Lecture implicite	Non NIL (AR explicite)	—	Non admis
Lecture/Ecriture explicite	NIL (AR implicite)	—	Non admis
Lecture/Ecriture explicite	Non NIL (AR établie)	Ignorer	API, Numéro d'Emplacement, Numéro de Sous-emplacement, l'Index doit être évalué

Tableau 372 – Expression 2 (spécifique à l'emplacement)

RPCOperationNmb	ARUUID	ARUUID cible	Validité
Lecture implicite	NIL (AR implicite)	Évalué si requis par l'index	API, Numéro d'Emplacement, l'Index doit être évalué
Lecture implicite	Non NIL (AR explicite)	—	Non admis
Lecture/Ecriture explicite	NIL (AR implicite)	—	Non admis
Lecture/Ecriture explicite	Non NIL (AR établie)	Ignorer	API, Numéro d'Emplacement, l'Index doit être évalué

Tableau 373 – Expression 3 (spécifique à la relation entre applications)

RPCOperationNmb	ARUUID	ARUUID cible	Validité
Lecture implicite	NIL (AR implicite)	Évalué si requis par l'index	L'Index doit être évalué
Lecture implicite	Non NIL (AR explicite)	—	Non admis
Lecture/Ecriture explicite	NIL (AR implicite)	—	Non admis
Lecture/Ecriture explicite	Non NIL (AR établie)	Ignorer	L'Index doit être évalué

Tableau 374 – Expression 4 (spécifique à l'API)

RPCOperationNmb	ARUUID	ARUUID cible	Validité
Lecture implicite	NIL (AR implicite)	Ignorer	API, l'Index doit être évalué
Lecture implicite	Non NIL (AR explicite)	—	Non admis
Lecture/Ecriture explicite	NIL (AR implicite)	—	Non admis
Lecture/Ecriture explicite	Non NIL (AR établie)	Ignorer	API, l'Index doit être évalué

Tableau 375 – Expression 5 (spécifique à l'appareil)

RPCOperationNmb	ARUUID	ARUUID cible	Validité
Lecture implicite	NIL (AR implicite)	Ignorer	L'Index doit être évalué
Lecture implicite	Non NIL (AR explicite)	—	Non admis
Lecture/Ecriture explicite	NIL (AR implicite)	—	Non admis
Lecture/Ecriture explicite	Non NIL (AR établie)	Ignorer	L'Index doit être évalué

5.2.3.4.3 Groupement de DiagnosisData pour les enregistrements de diagnostic

Les enregistrements de diagnostic contiennent des données différentes en fonction de l'index. Le groupement doit être conforme au Tableau 376. L'identifiant est utilisé au Tableau 378, au Tableau 379, au Tableau 380, au Tableau 381 et au Tableau 382.

Tableau 376 – Groupement de DiagnosisData

Identifiant	Signification
Diagnostic du codage de canal	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis avec ChannelProperties.Maintenance(=diagnosis) et ChannelProperties.Specifier(=appear)
Diagnostic de tous les codages	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis, ManufacturerSpecificDiagnosis avec ChannelProperties.Maintenance(=diagnosis) et ChannelProperties.Specifier(=appear)
Maintenance requise dans le codage de canal	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis avec ChannelProperties.Maintenance(=MaintenanceRequired) et ChannelProperties.Specifier(=appear)

Identifiant	Signification
Maintenance requise dans tous les codages	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis, ManufacturerSpecificDiagnosis avec ChannelProperties.Maintenance(=MaintenanceRequired) et ChannelProperties.Specifier(=appear)
Maintenance demandée dans le codage de canal	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis avec ChannelProperties.Maintenance(=MaintenanceDemanded) et ChannelProperties.Specifier(=appear)
Maintenance demandée dans tous les codages	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis, ManufacturerSpecificDiagnosis avec ChannelProperties.Maintenance(=MaintenanceDemanded) et ChannelProperties.Specifier(=appear)
Diagnostic, Maintenance, Qualifié et Etat	ChannelDiagnosis, ExtChannelDiagnosis, QualifiedChannelDiagnosis, ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=any) and ChannelProperties.Specifier(=appear), and ManufacturerSpecificDiagnosis with ChannelProperties.Maintenance(=diagnosis) and ChannelProperties.Specifier(=disappear)

5.2.3.4.4 Numéros attribués

Le codage des données d'enregistrement spécifiques à l'utilisateur doit être conforme au Tableau 377.

Tableau 377 – Index (spécifique à l'utilisateur)

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUUID, API, SlotNumber, SubslotNumber et TargetARUUUID
0 – 0x7FFF	RecordData spécifique à un utilisateur	Expression 1 s'applique.

Le codage des données d'enregistrement définies spécifiques au sous-module doit être conforme au Tableau 378.

Tableau 378 – Index (spécifique à un sous-emplacement)

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUUID, API, SlotNumber, SubslotNumber et TargetARUUUID
0x8000	ExpectedIdentificationData pour un sous-emplacement	Expression 1 s'applique.
0x8001	RealIdentificationData pour un sous-emplacement	Expression 1 s'applique.
0x8002 – 0x8009	Réservé	Réservé
0x800A	Diagnostic dans le codage de voie d'une sous-baie	Expression 1 s'applique.
0x800B	Diagnostic dans tous les codages d'une sous-baie	Expression 1 s'applique.
0x800C	Diagnostic, Maintenance, Qualifié et Etat pour un sous-emplacement	Expression 1 s'applique.
0x800D – 0x800F	Réservé	Réservé
0x8010	Maintenance requise dans le codage de voie d'une sous-baie	Expression 1 s'applique.

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUUID, API, SlotNumber, SubslotNumber et TargetARUUUID
0x8011	Maintenance exigée dans le codage de voie d'une sous-baie	Expression 1 s'applique.
0x8012	Maintenance requise dans tous les codages d'un sous-emplacement	Expression 1 s'applique.
0x8013	Maintenance demandée dans tous les codages d'un sous-emplacement	Expression 1 s'applique.
0x8014 – 0x801D	Réservé	Réservé
0x801E	SubstituteValues pour un sous-emplacement	Expression 1 s'applique.
0x801F	Réservé	Réservé
0x8020	PDIRSubframeData pour un sous emplacement	Expression 1 s'applique.
0x8021 – 0x8027	Réservé	Réservé
0x8028	RecordInputDataObjectElement pour un sous-emplacement	Expression 1 s'applique.
0x8029	RecordOutputDataObjectElement pour un sous-emplacement	Expression 1 s'applique.
0x802A	PDPortDataReal pour un sous-emplacement	Expression 1 s'applique.
0x802B	PDPortDataCheck pour un sous-emplacement	Expression 1 s'applique.
0x802C	PDIRData pour un sous-emplacement	Expression 1 s'applique.
0x802D	PDSyncData prévu pour un sous-emplacement avec valeur 0 attribuée à SyncID	Expression 1 s'applique.
0x802E	Réservé (version originelle)	Réservé
0x802F	PDPortDataAdjust pour un sous-emplacement	Expression 1 s'applique.
0x8030	IsochronousModeData pour un sous-emplacement	Expression 1 s'applique.
0x8031	Expected PDTimeData pour un sous-emplacement	Expression 1 s'applique.
0x8032 – 0x804F	Réservé	Réservé
0x8050	PDInterfaceMrpDataReal pour un sous-emplacement	Expression 1 s'applique.
0x8051	PDInterfaceMrpDataCheck pour un sous-emplacement	Expression 1 s'applique.
0x8052	PDInterfaceMrpDataAdjust pour un sous-emplacement	Expression 1 s'applique.
0x8053	PDPortMrpDataAdjust pour un sous-emplacement	Expression 1 s'applique.
0x8054	PDPortMrpDataReal pour un sous-emplacement	Expression 1 s'applique.
0x8055 – 0x805F	Réservé	Réservé
0x8060	PDPortFODDataReal pour un sous-emplacement	Expression 1 s'applique.
0x8061	PDPortFODDataCheck pour un sous-emplacement	Expression 1 s'applique.
0x8062	PDPortFODDataAdjust pour un sous-emplacement	Expression 1 s'applique.

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUIID, API, SlotNumber, SubslotNumber et TargetARUUIID
0x8063 – 0x806F	Réservé	Réservé
0x8070	PDNCDataCheck pour un sous-emplacement	Expression 1 s'applique.
0x8071	PDInterfaceAdjust pour un sous-emplacement	Expression 1 s'applique.
0x8072	PDPortStatistic pour un sous-emplacement	Expression 1 s'applique.
0x8073 – 0x807F	Réservé	Réservé
0x8080	PDInterfaceDataReal pour un sous-emplacement	Expression 1 s'applique.
0x8081 – 0x808F	Réservé	Réservé
0x8090	PDInterfaceFSUDDataAdjust prévu	Expression 1 s'applique.
0x8091 – 0x809F	Réservé	Réservé
0x80A0	Profils couvrant les économies d'énergie – Record_0	Expression 1 s'applique.
0x80A1 – 0x80AF	Réservé aux profils relatifs aux économies d'énergie	Expression 1 s'applique.
0x80B0	CombinedObjectContainer	Expression 1 s'applique.
0x80B1 – 0x80BF	Réservé	Réservé
0x80C0	Profils concernant une séquence d'événements – Record_0	Expression 1 s'applique.
0x80C1 – 0x80CF	Réservé à des profils relatifs à une séquence d'événements	Expression 1 s'applique.
0x80D0 – 0xAFFF	Réservé	Réservé
0xAFF0	I&M0	Expression 1 s'applique.
0xAFF1	I&M1	Expression 1 s'applique.
0xAFF2	I&M2	Expression 1 s'applique.
0xAFF3	I&M3	Expression 1 s'applique.
0xAFF4	I&M4	Expression 1 s'applique.
0xAFF5 – 0xAFFF	I&M5 – I&M15 ^a ^a Réservé aux données supplémentaires d'identification et de maintenance	Expression 1 s'applique.
0xB000 – 0xBFFF	Réservé aux profils	Expression 1 s'applique.

Le codage des données d'enregistrement définies spécifiques au module doit être conforme au Tableau 379.

Tableau 379 – Index (spécifique à l'emplacement)

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUID, API, SlotNumber, SubslotNumber et TargetARUUID
0xC000	ExpectedIdentificationData pour un emplacement	Expression 2 s'applique.
0xC001	RealIdentificationData pour un emplacement	Expression 2 s'applique.
0xC002 – 0xC009	Réservé	Réservé
0xC00A	Diagnostic dans le codage de voie d'une baie	Expression 2 s'applique.
0xC00B	Diagnostic dans tous les codages d'une baie	Expression 2 s'applique.
0xC00C	Diagnostic, Maintenance, Qualifié et Etat pour un emplacement	Expression 2 s'applique.
0xC00D – 0xC00F	Réservé	Réservé
0xC010	Maintenance requise dans le codage de canal d'un emplacement	Expression 2 s'applique.
0xC011	Maintenance demandée dans le codage de canal d'un emplacement	Expression 2 s'applique.
0xC012	Maintenance requise dans tous les codages d'un emplacement	Expression 2 s'applique.
0xC013	Maintenance demandée dans tous les codages d'un emplacement	Expression 2 s'applique.
0xC014 – 0xCFFF	Réservé	Réservé
0xD000 – 0xDFFF	Réservé aux profils	Expression 2 s'applique.

Le codage des données d'enregistrement définies spécifiques à la relation entre applications doit être conforme au Tableau 380.

Tableau 380 – Index (spécifique à une relation entre applications)

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUID, API, SlotNumber, SubslotNumber et TargetARUUID
0xE000	ExpectedIdentificationData pour une AR	Expression 3 s'applique.
0xE001	RealIdentificationData pour une AR	Expression 3 s'applique.
0xE002	ModuleDiffBlock pour une AR	Expression 3 s'applique.
0xE003 – 0xE009	Réservé	Réservé
0xE00A	Diagnostic dans le codage de canal d'une AR	Expression 3 s'applique.
0xE00B	Diagnostic dans tous les codages d'une AR	Expression 3 s'applique.
0xE00C	Diagnostic, Maintenance, Qualifié et Etat pour une AR	Expression 3 s'applique.
0xE00D – 0xE00F	Réservé	Réservé

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUID, API, SlotNumber, SubslotNumber et TargetARUUID
0xE010	Maintenance requise dans le codage de canal d'une AR	Expression 3 s'applique.
0xE011	Maintenance demandée dans le codage de canal d'une AR	Expression 3 s'applique.
0xE012	Maintenance requise dans tous les codages d'une AR	Expression 3 s'applique.
0xE013	Maintenance demandée dans tous les codages d'une AR	Expression 3 s'applique.
0xE014 – 0xE02F	Réservé	Réservé
0xE030	Réservé	Réservé
0xE031 – 0xE03F	Réservé	Réservé
0xE040	WriteMultiple	Expression 3 s'applique.
0xE041 – 0xE04F	Réservé	Réservé
0xE050	ARFSUDDataAdjust data for one AR Version originelle, uniquement utilisé pour ARProperties.StartupMode == Legacy	Expression 3 s'applique.
0xE051 – 0xE05F	Réservé à FastStartUp	Expression 3 s'applique.
0xE060 – 0xEBFF	Réservé	Réservé
0xEC00 – 0xEFFF	Réservé aux profils	Expression 3 s'applique.

Le codage des données d'enregistrement définies spécifiques à l'API doit être conforme au Tableau 381 si aucune relation entre applications n'a été établie.

Tableau 381 – Index (spécifique à l'API)

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUID, API, SlotNumber, SubslotNumber et TargetARUUID
0xF000	RealIdentificationData pour un API	Expression 4 s'applique.
0xF001 – 0xF009	Réservé	Réservé
0xF00A	Diagnostic dans le codage de canal d'un API	Expression 4 s'applique.
0xF00B	Diagnostic dans tous les codages d'un API	Expression 4 s'applique.
0xF00C	Diagnostic, Maintenance, Qualifié et Etat pour un API	Expression 4 s'applique.
0xF00D – 0xF00F	Réservé	Réservé
0xF010	Maintenance requise dans le codage de canal d'un API	Expression 4 s'applique.
0xF011	Maintenance demandée dans le codage de canal d'un API	Expression 4 s'applique.
0xF012	Maintenance requise dans tous les codages d'un API	Expression 4 s'applique.

0xF013	Maintenance demandée dans tous les codages d'un API	Expression 4 s'applique.
0xF014 – 0xF01F	Réservé	Réservé
0xF020	ARData pour un API	Expression 4 s'applique.
0xF021 – 0xF3FF	Réservé	Réservé
0xF400 – 0xF7FF	Réservé aux profils	Expression 4 s'applique.

Le codage des données d'enregistrement définies spécifiques à l'appareil doit être conforme au Tableau 382.

Tableau 382 – Index (spécifique à l'appareil)

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUID, API, SlotNumber, SubslotNumber et TargetARUUID
0xF800 – 0xF80B	Réservé	Réservé
0xF80C	Diagnostic, Maintenance, Qualifié et Etat pour un appareil	Expression 5 s'applique.
0xF80D – 0xF81F	Réservé	Réservé
0xF820	ARData	Expression 5 s'applique.
0xF821	APIData	Expression 5 s'applique.
0xF822 – 0xF82F	Réservé	Réservé
0xF830	LogBookData	Expression 5 s'applique.
0xF831	PdevData	Expression 5 s'applique.
0xF832 – 0xF83F	Réservé	Réservé
0xF840	I&M0FilterData	Expression 5 s'applique.
0xF841	PDRealData	Expression 5 s'applique.
0xF842	PDExpectedData	Expression 5 s'applique.
0xF843 – 0xF84F	Réservé	Réservé
0xF850	AutoConfiguration	Expression 5 s'applique.
0xF851 – F85F	Réservé	Réservé
0xF860	Communication entre contrôleurs Téléchargement GSD à l'aide de UploadBLOBQuery et UploadBLOB	Expression 5 s'applique.
0xF861	Communication entre contrôleurs Information de diagnostic imbriquée	Expression 5 s'applique.
0xF862 – 0xF86F	Réservé à la communication entre contrôleurs	Réservé
0xF870 – 0xFBFE	Réservé	Réservé
0xFBFF	Index déclencheur pour la surveillance de la connexion RPC à l'intérieur de la CMSM	Expression 5 s'applique.

Valeur (hexadécimale)	Signification de l'index	Signification des champs ARUUID, API, SlotNumber, SubslotNumber et TargetARUUID
0xFC00 – 0xFFFF	Reservé aux profils ^a	Expression 5 s'applique.
^a L'index est défini sous la forme d'un numéro administratif.		

5.2.4 Section de codage relative aux services AL

5.2.4.1 Codage du champ RecordDataLength

Ce champ doit être codé en type de données Unsigned32. Le champ RecordDataLength doit uniquement contenir le nombre d'octets de l'utilisateur.

5.2.4.2 Codage du champ SeqNumber

Ce champ doit être codé en type de données Unsigned16 et doit être incrémenté pour chaque IODWriteReq et chaque IODReadReq par le demandeur et reflété par le répondeur; la couche de protocole ne doit pas vérifier le SeqNumber.

Il convient que la vérification de l'exactitude de SeqNumber soit effectuée par l'application.

Le SeqNumber est valide dans le contexte d'un SessionKey et commence par zéro.

NOTE Le SeqNumber et le SessionKey permettent la détection des erreurs de séquence.

Ce champ doit contenir la valeur 0 pour une AR implicite.

5.2.4.3 Codage du champ ARType

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 383.

Tableau 383 – ARType

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001	IOCARSingl	—
0x0002 – 0x0005	Réservé	—
0x0006	IOSAR	L'AR du superviseur est une forme particulière d'IOCARSingl permettant la prise de contrôle de la propriété d'un sous-module
0x0007 – 0x000F	Réservé	—
0x0010	IOCARSingl utilisant RT_CLASS_3	Il s'agit d'une forme particulière d'IOCARSingl indiquant une communication RT_CLASS_3
0x0011 – 0x001F	Réservé	—
0x0020	IOCARSR	L'AR SR est une forme particulière d'IOCARSingl indiquant la redondance du système ou la configuration en cours d'exécution
0x0021 – 0xFFFF	Réservé	—

5.2.4.4 Codage du champ SessionKey

Ce champ doit être codé en type de données Unsigned16 et doit être augmentée d'un par connexion de CMInitiator. CMResponder doit utiliser cette valeur pour chaque PROFINETIOServiceReq- et PROFINETIOServiceResPDU consécutifs de cette session.

NOTE SessionKey permet à CMInitiator de détecter les erreurs de séquence lors de la phase d'établissement et de libération d'une relation entre applications.

Ce champ doit contenir la valeur zéro de la relation entre applications implicite.

5.2.4.5 Codage du champ CMInitiatorMacAdd

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ CMInitiatorMacAdd doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

5.2.4.6 Codage du champ IOCRMulticastMACAdd

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ IOCRMulticastMACAdd doit être réglée conformément aux adresses LAN MAC universelles en 48 bits de l'IEEE 802 au Tableau 384 et au Tableau 385.

L'adresse multidiffusion IP utilisée pour la relation de communication multidiffusion à l'aide de RT_CLASS_UDP doit être mise selon la RFC 2365.

Tableau 384 – IOCRMulticastMACAdd utilisant RT_CLASS_UDP

Adresse multidiffusion IP	Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifieur (hexadécimale)	FrameID associé	Utilisation
239.192.248.0	01-00-5E	40-F8-00	0xF800	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP
239.192.248.1 – 239.192.251.254	01-00-5E	40-F8-01 à 40-FB-FE	0xF801 – 0xFBFE	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP
239.192.251.255	01-00-5E	40-FB-FF	0xFBFF	Utilisé pour les relations de communication multidiffusion conjointement avec RT_CLASS_UDP

Tableau 385 – IOCRMulticastMACAdd utilisant RT_CLASS_2 ou RT_CLASS_3

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifieur (hexadécimale)	Signification
01-0E-CF	00-00-00 à 00-00-FF	Réservé pour d'autres applications
01-0E-CF	00-01-00	Réservé pour d'autres adresses multidiffusion dans le contexte du Type 10
01-0E-CF	00-01-01	Adresse multidiffusion de destination RT_CLASS_3
01-0E-CF	00-01-02	Adresse multidiffusion de trame non valide RT_CLASS_3
01-0E-CF	00-01-03 à 00-01-FF	Réservé pour d'autres adresses multidiffusion dans le contexte du Type 10

Valeur OUI (Multidiffusion) (hexadécimale)	Valeur ExtensionIdentifiant (hexadécimale)	Signification
01-0E-CF	00-02-00 à 00-02-FF	Adresse de communication multidiffusion RT_CLASS_2
01-0E-CF	00-03-00 à 00-03-FF	Réservé pour d'autres adresses multidiffusion dans le contexte du Type 10
01-0E-CF	00-04-00 to FF-FF-FF	Réservé pour d'autres applications
03-00-00 à FF-FF-00	00-00-00	Plage d'adresses de multidiffusion administrée au niveau local pour le Transfert Rapide

NOTE L'octet 1 contient le bit d'adresse individuelle/de groupe (bit de poids faible).

L'identificateur Organizationally Unique Identifier (OUI) comme le définit IEEE 802 pour le Type 10 est 00-0E-CF et doit être réglé conformément au Tableau 386.

Tableau 386 – OUI du Type 10

Valeur (hexadécimale)	Signification
00-0E-CF	Monodiffusion individuelle administrée au niveau global
01-0E-CF	Adresse (multidiffusion) de groupe administrée au niveau global
02-0E-CF	Monodiffusion administrée au niveau local
03-0E-CF	Adresse (multidiffusion) de groupe administrée au niveau local

5.2.4.7 Codage du champ CMResponderMacAdd

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ CMResponderMacAdd doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

NOTE L'octet 1 contient le bit d'adresse individuelle/de groupe (bit de poids faible).

5.2.4.8 Codage du champ ARProperties

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0 – 2: ARProperties.State

Ce champ doit être mis conformément au Tableau 387.

Tableau 387 – ARProperties.State

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Active
0x02 – 0x07	Réservé

Bit 3: ARProperties.SupervisorTakeoverAllowed

Ce champ doit être mis conformément au Tableau 388.

Tableau 388 – ARProperties.SupervisorTakeoverAllowed

Valeur (hexadécimale)	Signification
0x00	Non admis
0x01	Autorisé

Bit 4: ARProperties.ParameterizationServer

Ce champ doit être mis conformément au Tableau 389.

Tableau 389 – ARProperties.ParameterizationServer

Valeur (hexadécimale)	Signification
0x00	PrmServer Externe
0x01	CM Initiator

Bit 5 – 7: ARProperties.reserved_1

Ce champ doit être mis conformément à 3.4.2.2.

Bit 8: ARProperties.DeviceAccess

Ce champ doit être mis conformément au Tableau 390.

Tableau 390 – ARProperties.DeviceAccess

Valeur (hexadécimale)	Signification
0x00	Seuls les sous-modules issus de ExpectedSubmoduleBlock sont accessibles
0x01	L'accès au sous-module est contrôlé par l'application de l'appareil d'E/S

Bit 9-10: ARProperties.CompanionAR

Ce champ doit être mis conformément au Tableau 391.

Tableau 391 – ARProperties.CompanionAR

Valeur (hexadécimale)	Signification	Utilisation
0x00	AR unique	Etablissement de connexion RT_CLASS_1, RT_CLASS_2, RT_CLASS_3 ou RT_CLASS_UDP
0x01	Première AR d'une paire d'accompagnement et une AR d'accompagnement doit suivre	A usage ultérieur
0x02	AR d'accompagnement	A usage ultérieur
0x03	Réservé	—

Bit 11: ARProperties.AcknowledgeCompanionAR

Ce champ doit être mis conformément au Tableau 392.

Tableau 392 – ARProperties.AcknowledgeCompanionAR

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucune AR d'accompagnement ni d'acquiescement pour l'AR d'accompagnement requise	Etablissement de connexion RT_CLASS_1, RT_CLASS_2, RT_CLASS_3 ou RT_CLASS_UDP
0x01	AR d'accompagnement avec acquiescement	A usage ultérieur

Bit 12 – 23: ARProperties.reserved_2

Ce champ doit être mis conformément à 3.4.2.2.

Bit 24 – 29: ARProperties.reserved_3

Ce champ doit être mis à zéro.

Bit 30: ARProperties.StartupMode

Ce champ doit être mis conformément au Tableau 393.

Tableau 393 – ARProperties.StartupMode

Valeur (hexadécimale)	Signification	Utilisation
0x00	Legacy (version originelle)	Recommandé Mode de compatibilité pour les nœuds d'une version originelle. Le mode deux fournisseurs pour le démarrage de RT_CLASS_3 est requis
0x01	Advanced (avancé)	Obligatoire pour cette norme.

Bit 31: ARProperties.PullModuleAlarmAllowed

Ce champ doit être mis conformément au Tableau 394.

Tableau 394 – ARProperties.PullModuleAlarmAllowed

Valeur (hexadécimale)	Signification	Utilisation
0x00	AlarmType(=Pull) doit signaler l'extraction du sous-module et du module. Le numéro de sous-emplacement zéro doit coder l'extraction du module conjointement avec AlarmType(=Pull).	Obligatoire
0x01	AlarmType(=Pull) doit signaler l'extraction du sous-module. AlarmType(=Pull module) doit signaler l'extraction du module. Le numéro de sous-emplacement 0 – 0x8FFF doit coder l'extraction du sous-module conjointement avec AlarmType(=Pull).	Facultatif

5.2.4.9 Codage du champ IOCRProperties

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0 – 3: IOCRProperties.RTClass

Ce champ doit être mis conformément au Tableau 395.

Tableau 395 – IOCRProperties.RTClass

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	—
0x01	Utilisation de la gamme 7 de FrameID pour les IOCR Voir Tableau 30	Data-RTC-PDU IOCRProperties.RTClass == 0x02 doit être utilisé en remplacement par l'outil d'ingénierie. Doit être pris en charge par l'appareil de commande E/S et le superviseur E/S, pour des raisons d'héritage des appareils E/S. Il convient qu'elle ne soit pas générée par un appareil E/S.
0x02	Utilisation de la gamme 6 de FrameID pour les IOCR Voir Tableau 29	Data-RTC-PDU
0x03	Utilisation de la gamme 3 de FrameID pour les IOCR Voir Tableau 26	Data-RTC-PDU
0x04	Utilisation de la gamme 7 de FrameID pour les IOCR Voir Tableau 30	UDP-RTC-PDU
0x05 – 0x07	Réservé	—

Bit 4 – 12: IOCRProperties.reserved_1

Ce champ doit être mis à zéro.

Bit 13 – 23: IOCRProperties.reserved_2

Ce champ doit être mis conformément à 3.4.2.2.

Bit 24 – 31: IOCRProperties.reserved_3

Ce champ doit être mis à zéro.

5.2.4.10 Codage du champ NumberOfIODataObjects

Ce champ doit être codé en type de données Unsigned16.

5.2.4.11 Codage du champ NumberOfIOCS

Ce champ doit être codé en type de données Unsigned16.

5.2.4.12 Codage du champ IOCRReference

Ce champ doit être codé en type de données Unsigned16. Il s'agit d'une balise d'identification du CR utilisée dans IOCRBlockReq et IOCRBlockRes pour faire référence à DataItem. De plus, il est utilisé dans PDIRData de l'ASE Physical Device.

5.2.4.13 Codage du champ IOCRTagHeader

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0-11: IOCRTagHeader.IOCRVLANID

Ce champ doit être mis conformément au Tableau 396.

Tableau 396 – IOCRTagHeader.IOCRVLANID

Valeur (hexadécimale)	Signification
0x000	Aucun VLAN
0x001	VLAN par défaut
0x002 – 0xFFFF	Conformément à l'IEEE 802.1Q

Bit 12: IOCRTagHeader.reserved

Ce champ doit être mis à zéro.

Bit 13 – 15: IOCRTagHeader.IOUserPriority

Ce champ doit être mis conformément au Tableau 397.

Tableau 397 – IOCRTagHeader.IOUserPriority

Valeur (hexadécimale)	Signification
0x00 – 0x05	Réservé
0x06	Priorité IO CR
0x07	Réservé

5.2.4.14 Codage du champ IOCRTType

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 398.

Tableau 398 – IOCRTType

Valeur (hexadécimale)	Signification
0x0000	Réservé
0x0001	CR d'entrée
0x0002	CR de sortie
0x0003	CR du fournisseur multidiffusion
0x0004	CR du consommateur multidiffusion
0x0005 – 0xFFFF	Réservé

5.2.4.15 Codage du champ CMInitiatorActivityTimeoutFactor

Ce champ doit être codé en type de données Unsigned16 avec des valeurs conformes au Tableau 399 et au Tableau 400.

Tableau 399 – CMInitiatorActivityTimeoutFactor avec ARProperties.DeviceAccess:=0

Valeur (décimale)	Signification	Utilisation
0	Réservé	—
1 – 1 000	Avec une base de temps de 100 ms	L'appareil E/S surveille la durée entre la réponse Connect et la première activité suivante de l'appareil de commande d'E/S
1 001 – 65 535	Réservé	—

Tableau 400 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=1 ou ARProperties.StartupMode:=1

Valeur (décimale)	Signification	Utilisation
0 – 99	Réservé	—
100 – 1 000	Avec une base de temps de 100 ms	Valeur par défaut: 200 L'appareil E/S surveille la durée entre la réponse Connect et l'activité d'enregistrement Read et Write suivante de l'appareil de commande d'E/S
1 001 – 65 535	Réservé	—

Ce champ doit être utilisé à partir du CMDEV pour calculer CMInitiatorActivityTimeout à l'aide de la Formule (46).

$$\text{CMInitiatorActivityTimeout} = \text{CMInitiatorActivityTimeoutFactor} \times 100 \text{ ms} \quad (46)$$

où

CMInitiatorActivityTimeout est la temporisation de l'activité de l'initiateur CM

CMInitiatorActivityTimeoutFactor est le facteur de temporisation de l'activité de l'initiateur CM

Ce champ doit être utilisé à partir du CMCTL pour calculer CMInitiatorTriggerTimeoutFactor utilisé pour l'établissement de l'AR avec les valeurs conformes à la Formule (47) et au Tableau 401.

$$\text{CMInitiatorTriggerTimeoutFactor} = \text{CMInitiatorActivityTimeoutFactor} \text{ DIV } 3 \quad (47)$$

où

CMInitiatorActivityTimeoutFactor est le facteur de temporisation de l'activité de l'initiateur CM

CMInitiatorTriggerTimeoutFactor est le facteur de temporisation du déclenchement de l'initiateur CM

Tableau 401 – CMInitiatorTriggerTimeoutFactor

Valeur (décimale)	Signification	Utilisation
0 – 32	Réservé	—
33 – 333	Avec une base de temps de 100 ms	Utilisé pour la surveillance de la connexion lors du démarrage jusqu'à ce que la surveillance de la connexion PPM / CPM survienne.
334 – 65 535	Réservé	—

La valeur de CMInitiatorTriggerTimeout doit être calculée à l'aide de la Formule (48).

$$\text{CMInitiatorTriggerTimeout} = \text{CMInitiatorTriggerTimeoutFactor} \times 100 \text{ ms} \quad (48)$$

où

CMInitiatorTriggerTimeout est la temporisation du déclenchement de l'initiateur CM

CMInitiatorTriggerTimeoutFactor est le facteur de temporisation du déclenchement de l'initiateur CM

5.2.4.16 Codage du champ StationNameLength

Ce champ doit être codé en type de données Unsigned16.

5.2.4.17 Codage du champ CMInitiatorStationName

Ce champ doit être codé conformément aux règles de 4.3.1.4.15.1.

5.2.4.18 Codage du champ CMResponderStationName

Ce champ doit être codé conformément aux règles de 4.3.1.4.15.1.

5.2.4.19 Codage du champ ParameterServerStationName

Ce champ doit être codé conformément aux règles de 4.3.1.4.15.1.

5.2.4.20 Codage du champ ProviderStationName

Ce champ doit être codé conformément aux règles de 4.3.1.4.15.1.

5.2.4.21 Codage du champ IODataObjectFrameOffset

Ce champ doit être codé en type de données Unsigned16. Il est utilisé dans IOCRBlockReq pour faire référence au décalage de DataItem. Il doit être compris entre 0 et 1 439. Le nombre maximal d'octets utilisés ne doit pas dépasser la taille C_SDU maximale.

5.2.4.22 Codage du champ IOCSFrameOffset

Ce champ doit être codé en type de données Unsigned16. Il est utilisé dans IOCRBlockReq pour faire référence au décalage de DataItem. Il doit être compris entre 0 et 1 439. Le nombre maximal d'octets utilisés ne doit pas dépasser la taille C_SDU maximale.

5.2.4.23 Codage du champ LengthIOCS

Ce champ doit être codé en type de données Unsigned8 et doit être mis conformément au Tableau 402.

Tableau 402 – LengthIOCS

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Un octet pour IOCS
0x02 – 0xFF	Réservé

5.2.4.24 Codage du champ LengthIOPS

Ce champ doit être codé en type de données Unsigned8 et doit être mis conformément au Tableau 403.

Tableau 403 – LengthIOPS

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Un octet pour IOPS
0x02 – 0xFF	Réservé

5.2.4.25 Codage du champ LengthData

Ce champ doit être codé en type de données Unsigned16. Les valeurs doivent être comprises entre 0 et 1 439.

5.2.4.26 Codage du champ NumberOfAPIs

Ce champ doit être codé en type de données Unsigned16.

NOTE La valeur 0 est uniquement utilisée conjointement avec ARProperties.DeviceAccess =1.

5.2.4.27 Codage du champ AlarmCRProperties

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0: AlarmCRProperties.Priority

Ce champ doit être mis conformément au Tableau 404.

Tableau 404 – AlarmCRProperties.Priority

Valeur (hexadécimale)	Signification
0x00	Priorité utilisateur (par défaut). La priorité donnée par l'utilisateur est utilisée et deux ressources d'alarme sont disponibles
0x01	Priorité déterminée Utiliser uniquement la priorité basse, la priorité utilisateur est ignorée et une seule ressource d'alarme est disponible

Un appareil E/S servant de puits d'alarme doit toujours prendre en charge AlarmCRProperties.Priority="User priority" et l'appareil de commande E/S servant de source d'alarme peut choisir s'il se limite ou non à une ressource.

Bit 1: AlarmCRProperties.Transport

Ce champ doit être mis conformément au Tableau 405.

Tableau 405 – AlarmCRProperties.Transport

Valeur (hexadécimale)	Signification	Utilisation
0x00	RTA_CLASS_1	CR d'alarme utilise DATA-RTA-PDU
0x01	RTA_CLASS_UDP	CR d'alarme utilise UDP-RTA-PDU

Bit 2 – 23: AlarmCRProperties.reserved_1

Ce champ doit être mis conformément à 3.4.2.2.

Bit 24 – 31: AlarmCRProperties.reserved_2

Ce champ doit être mis à zéro.

5.2.4.28 Codage du champ AlarmCRTagHeaderHigh

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0-11: AlarmCRTagHeaderHigh.AlarmCRVLANID

Ce champ doit être mis conformément au Tableau 406.

Tableau 406 – AlarmCRTagHeaderHigh.AlarmCRVLANID

Valeur (hexadécimale)	Signification
0x000	Aucun VLAN
0x001	VLAN par défaut
0x002 – 0xFFFF	Conformément à l'IEEE 802.1Q

Bit 12: AlarmCRTagHeaderHigh.reserved

Ce champ doit être mis à zéro.

Bit 13 – 15: AlarmCRTagHeaderHigh.AlarmUserPriority

Ce champ doit être mis conformément au Tableau 407.

Tableau 407 – AlarmCRTagHeaderHigh.AlarmUserPriority

Valeur (hexadécimale)	Signification
0x00 – 0x05	Réservé
0x06	Priorité de CR d'alarme haute
0x07	Réservé

5.2.4.29 Codage du champ AlarmCRTagHeaderLow

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0-11: AlarmCRTagHeaderLow.AlarmCRVLANID

Ce champ doit être mis conformément au Tableau 408.

Tableau 408 – AlarmCRTagHeaderLow.AlarmCRVLANID

Valeur (hexadécimale)	Signification
0x000	Aucun VLAN
0x001	VLAN par défaut
0x002 – 0xFFFF	Conformément à l'IEEE 802.1Q

Bit 12: AlarmCRTagHeaderLow.reserved

Ce champ doit être mis à zéro.

Bit 13 – 15: AlarmCRTagHeaderLow.AlarmUserPriority

Ce champ doit être mis conformément au Tableau 409.

Tableau 409 – AlarmCRTagHeaderLow.AlarmUserPriority

Valeur (hexadécimale)	Signification
0x00 – 0x04	Réservé
0x05	Priorité de CR d'alarme basse
0x06 – 0x07	Réservé

5.2.4.30 Codage du champ AlarmSequenceNumber

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 410.

Tableau 410 – AlarmSequenceNumber

Valeur (hexadécimale)	Signification	Utilisation
0x000 – 0x7FF	Reflète le numéro de séquence de la notification d'alarme de connexion	Assure la relation entre la notification d'alarme de connexion et ControlBlockPlug dans Application Ready
0x800 – 0xFFFF	Réservé	–

5.2.4.31 Codage du champ AlarmCRType

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 411.

Tableau 411 – AlarmCRType

Valeur (hexadécimale)	Signification
0x0000	Réservé
0x0001	CR d'alarme
0x0002 – 0xFFFF	Réservé

5.2.4.32 Codage du champ RTATimeoutFactor

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 412. La base de temps est de 100 ms.

RTATimeout doit être calculé à l'aide de la Formule (49).

$$\text{RTATimeout} = \text{RTATimeoutFactor} \times 100 \text{ ms} \quad (49)$$

où

RTATimeout est la temporisation RTA

RTATimeoutFactor est le facteur de temporisation RTA

Tableau 412 – RTATimeoutFactor

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001	Obligatoire	Par défaut Surveillance d'acquittement de DATA-RTA-PDU ou UDP-RTA-PDU
0x0002 – 0x0064	Obligatoire	Surveillance d'acquittement de DATA-RTA-PDU ou UDP-RTA-PDU
0x0065 – 0xFFFF	Facultatif	Surveillance d'acquittement de DATA-RTA-PDU ou UDP-RTA-PDU

NOTE La valeur de RTATimeoutFactor dépend de la durée d'acheminement de DATA-RTA-PDU ou de UDP-RTA-PDU entre deux homologues.

5.2.4.33 Codage du champ RTARetries

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 413.

Tableau 413 – RTARetries

Valeur (hexadécimale)	Signification	Signification
0x0000 – 0x0002	Réservé	Réservé
0x0003	Obligatoire	Par défaut
0x0004 – 0x000F	Obligatoire	—
0x0010 – 0xFFFF	Réservé	Réservé

5.2.4.34 Codage du champ PROFINETIOConstantValue

Ce champ doit être codé sous la forme d'une structure de type de données contenant les éléments suivants.

- Data1 en Unsigned32 (0xDEA00000)
- Data2 en Unsigned16 (0x6C97)
- Data3 en Unsigned16 (0x11D1)
- Data4 sous la forme d'une matrice de deux Unsigned8, Octet 1 (0x82) à Octet 2 (0x71)

NOTE L'ordre de transmission des valeurs Unsigned32 ou Unsigned16 est fonction de RPC Flag Drep (petit-boutiste ou gros-boutiste) s'il fait partie de RPCHeader ou de NDREPMAP PDU. S'il fait partie des PDU NDRDataxxx, seul le format gros-boutiste est utilisé.

La valeur du champ PROFINETIOConstantValue doit être DEA00000-6C97-11D1-8271.

5.2.4.35 Codage du champ AddressResolutionProperties

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0 – 2: AddressResolutionProperties.Protocol

Ce champ doit être mis conformément au Tableau 414.

Tableau 414 – AddressResolutionProperties.Protocol

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	—
0x01	DNS	Le consommateur multidiffusion utilise DNS pour résoudre l'adresse MAC source du fournisseur multidiffusion.
0x02	DCP	Le consommateur multidiffusion utilise DCP pour résoudre l'adresse MAC source du fournisseur multidiffusion.
0x03 – 0x07	Réservé	—

Bit 3 – 7: AddressResolutionProperties.reserved_1

Ce champ doit être mis conformément à 3.4.2.2.

Bit 8 – 15: AddressResolutionProperties.reserved_2

Ce champ doit être mis à zéro.

Bit 16 – 31: AddressResolutionProperties.Factor

Ce champ doit être codé avec les valeurs conformes au Tableau 415. La base de temps est d'une seconde. AddressResolutionInterval doit être calculé à l'aide de la Formule (50).

Tableau 415 – AddressResolutionProperties.Factor

Valeur (hexadécimale)	Signification
0x0000	Réservé
0x0001 – 0x0064	Obligatoire
0x0065 – 0xFFFF	Facultatif

$$\text{AddressResolutionInterval} = \text{AddressResolutionProperties.Factor} \times 1 \text{ s} \quad (50)$$

où

AddressResolutionInterval est l'intervalle de résolution de l'adresse

AddressResolutionProperties.Factor est le facteur pour le calcul de l'intervalle de résolution de l'adresse

5.2.4.36 Codage du champ MCTimeoutFactor

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 416. La base de temps doit être 100 ms. MCIMonitoringInterval doit être calculé à l'aide de la Formule (51).

Tableau 416 – MCTimeoutFactor

Valeur (hexadécimale)	Signification
0x0000 – 0x0064	Obligatoire
0x0065 – 0xFFFF	Réservé

$$\text{MCIMonitoringInterval} = \text{MCTimeoutFactor} \times 100 \text{ ms} \quad (51)$$

où

MCIMonitoringInterval est l'intervalle de surveillance MCI

MCTimeoutFactor

est le facteur pour le calcul de l'intervalle de surveillance MCI

5.2.4.37 Codage des champs liés à Instance, DeviceID et VendorID

5.2.4.37.1 Généralités

Les champs Unsigned16 Instance, DeviceID et VendorID sont divisés en deux champs Unsigned8 afin de définir l'ordre des octets à l'intérieur d'ObjectUUID OctetString.

5.2.4.37.2 Codage du champ InstanceLow

Ce champ doit être codé en type de données Unsigned8.

5.2.4.37.3 Codage du champ InstanceHigh

Ce champ doit être codé en type de données Unsigned8.

Il est recommandé d'attribuer la valeur zéro à InstanceHigh et InstanceLow pour les appareils à instance unique.

5.2.4.37.4 Codage du champ DeviceIDLow

Ce champ, défini sous la forme d'un numéro administratif, doit être codé en type de données Unsigned8 par le fabricant de l'appareil.

La valeur zéro pour DeviceIDHigh et DeviceIDLow est réservée.

5.2.4.37.5 Codage du champ DeviceIDHigh

Ce champ, défini sous la forme d'un numéro administratif, doit être codé en type de données Unsigned8 par le fabricant de l'appareil.

La valeur zéro pour DeviceIDHigh et DeviceIDLow est réservée.

5.2.4.37.6 Codage du champ VendorIDLow

Ce champ, défini sous la forme d'un numéro administratif, doit être codé en type de données Unsigned8 avec les valeurs conformes au Tableau 417.

Tableau 417 – VendorIDLow

Valeur (hexadécimale)	Signification
0x00	La valeur zéro pour VendorIDHigh et VendorIDLow est réservée aux appareils de la CEI 61158-6-3.
0x01 – 0xFF	Numéro administratif

5.2.4.37.7 Codage du champ VendorIDHigh

Ce champ, défini sous la forme d'un numéro administratif, doit être codé en type de données Unsigned8 avec les valeurs conformes au Tableau 418.

Tableau 418 – VendorIDHigh

Valeur (hexadécimale)	Signification
0x00	La valeur 0 pour VendorIDHigh et VendorIDLow est réservée aux appareils de la CEI 61158-6-3.
0x01 – 0xEF	Numéro administratif
0xFF	La valeur 0xFF est réservée aux profils.

5.2.4.38 Codage du champ ModuleIdentNumber

Ce champ doit être codé en type de données Unsigned32. Ce champ doit être codé avec les valeurs conformes au Tableau 419.

Tableau 419 – ModuleIdentNumber

Valeur (hexadécimale)	Signification
0x00000000	Réservé
0x00000001 – 0xFFFFFFFF	Spécifique à un fabricant

5.2.4.39 Codage du champ SubmoduleIdentNumber

Ce champ doit être codé en type de données Unsigned32. Ce champ doit être codé avec les valeurs conformes au Tableau 420.

Tableau 420 – SubmoduleIdentNumber

Valeur (hexadécimale)	Signification ARProperties.PullModuleAlarm- Allowed := 0	Signification ARProperties.PullModuleAlarm- Allowed := 1
0x00000000	Attribuée au sous-emplacement zéro uniquement accompagnée de Pull alarm et de SubslotNumber:= 0 comme identification de "Pull module" de la fonction. Spécifique à un fabricant pour les sous-emplacements > 0	Spécifique à un fabricant pour tous les autres sous-baies
0x00000001 – 0xFFFFFFFF	Spécifique à un fabricant pour tous les autres sous-baies	

La numérotation pour l'identification d'appareil est hiérarchique. Le niveau supérieur est DeviceIdentNumber, qui contient un numéro administratif attribué de façon univoque. Le deuxième niveau est ModuleIdentNumber, qui est unique dans le domaine d'application de DeviceIdentNumber. Le troisième niveau est SubmoduleIdentNumber, qui est unique dans le domaine d'application de ModuleIdentNumber.

NOTE 1 Unique signifie que les propriétés du sous-module associé (la structure de données, le diagnostic et le paramétrage, par exemple) définies dans la présente norme sont identifiées.

NOTE 2 Le matériel d'un sous-module est identifié par VendorID et OrderID.

5.2.4.40 Codage du champ NumberOfARs

Ce champ doit être codé en type de données Unsigned16.

5.2.4.41 Codage du champ NumberOfIOCRs

Ce champ doit être codé en type de données Unsigned16.

5.2.4.42 Codage du champ SubmoduleDataLength

Ce champ doit être codé en type de données Unsigned16. La plage doit être comprise entre 0 et 1 439.

5.2.4.43 Codage du champ NumberOfModules

Ce champ doit être codé en type de données Unsigned16.

5.2.4.44 Codage du champ NumberOfSlots

Ce champ doit être codé en type de données Unsigned16.

5.2.4.45 Codage du champ NumberOfSubmodules

Ce champ doit être codé en type de données Unsigned16.

5.2.4.46 Codage du champ NumberOfSubslots

Ce champ doit être codé en type de données Unsigned16.

5.2.4.47 Codage du champ ARUUID

Ce champ doit être codé en type de données UUID conformément au Tableau 421 et au Tableau 422.

Tableau 421 – ARUUID

Valeur (UUID)	Signification	Utilisation
00000000-0000-0000-0000-000000000000	Réservé	La valeur NIL indique l'utilisation de l'AR implicite.
Autre	Valide pour une AR	Valeurs non ambiguës qui identifient des AR. Voir Tableau 422 pour IOCARSR

Tableau 422 – ARUUID conjointement avec ARType:= IOCARSR

Valeur	Description
aaaaaaaa-bbbb-cccc-dddd-eeeeeeeezzzz	Identifie une relation entre applications où
	aaaaaaa bbbb (6 octets) Est similaire à l'ARUUID standard
	cccc dddd eeeeeeee (8 octets) Est utilisé comme informations de configuration relatives au contenu de Connect.req.
	Bit 0-2: 01: 1 ^{ère} AR d'un ensemble d'AR 02: 2 ^{ème} AR d'un ensemble d'AR 03: 3 ^{ème} AR d'un ensemble d'AR 04: 4 ^{ème} AR d'un ensemble d'AR zzzz (2 octets) Autres: Réservé
	Bit 3-4: 02: Le point d'extrémité de communication doit attribuer deux AR pour AR-set Autres: Réservé
L'ARUUID d'un IOCARSR a pour seule différence le Bit 0-2.	
Bit 5-15: Réservé	

5.2.4.48 Codage du champ TargetARUUID

Ce champ doit être codé en type de données UUID conformément au Tableau 423.

Tableau 423 – TargetARUUID

Valeur (UUID)	Signification	Utilisation
00000000-0000-0000-0000-000000000000	Réservé	La valeur NIL indique l'utilisation de l'AR implicite.
autre	Valide pour une AR	Toutes les autres valeurs identifient des AR établies.

5.2.4.49 Codage du champ ActualLocalTimeStamp

Ce champ doit être codé en type de données Unsigned64 conformément au Tableau 424.

Tableau 424 – ActualLocalTimeStamp

Valeur hexadécim	Signification	Utilisation
0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	Contient la valeur du nombre de cycles en cours lors de la lecture du journal.	—

5.2.4.50 Codage du champ LocalTimeStamp

Ce champ doit être codé en type de données Unsigned64 conformément au Tableau 425.

Tableau 425 – LocalTimeStamp

Valeur (hexadécimale)	Signification	Utilisation
0x0000000000000000 – 0xFFFFFFFFFFFFFFFF	Contient la valeur du nombre de cycles en cours lors du stockage de l'entrée dans le journal.	—

5.2.4.51 Codage du champ NumberOfLogEntries

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 426.

Tableau 426 – NumberOfLogEntries

Valeur	Signification	Utilisation
0	Réservé	—
16	Le nombre minimal doit être de 16 pour un appareil d'E/S	—
128	Le nombre minimal doit être de 128 pour un appareil de commande d'E/S	—
autre	Nombre d'entrées de journal	—

5.2.4.52 Codage du champ EntryDetail

Ce champ doit être codé en type de données Unsigned32 en fonction du comportement de la machine de protocole et conformément au Tableau 427.

Tableau 427 – EntryDetail

Valeur (hexadécimale)	Signification	Utilisation
0x00000000	Aucun détail	—
autre	Valeur issue de la machine de protocole de signalisation	—

5.2.4.53 Codage des champs AdditionalValue1 et AdditionalValue2

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 428.

Tableau 428 – AdditionalValue1 et AdditionalValue2

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Aucune information supplémentaire ou réponse positive	IODReadRes, IODWriteRes, IODWriteMultipleRes
autre	Informations utilisateur supplémentaires dans les réponses négatives	IODReadRes, IODWriteRes, IODWriteMultipleRes

5.2.4.54 Codage du champ ControlBlockProperties

Le codage de ce champ doit être conforme à 3.4.2.4, au Tableau 429, au Tableau 430 et au Tableau 431.

Tableau 429 – ControlBlockProperties conjointement avec ControlCommand.ApplicationReady avec ARProperties.StartupMode:=1

Position de bit	Valeur (hexadécimale)	Signification
Bit 0 – 15	—	Doit être mis conformément à 3.4.2.2.

Tableau 430 – ControlBlockProperties conjointement avec ControlCommand.ApplicationReady avec ARProperties.StartupMode:=0

Position de bit	Valeur (hexadécimale)	Signification
Bit 0	0x00	Attente de ControlCommand.ReadyForCompanion explicite
	0x01	ControlCommand.ReadyForCompanion implicite
Bit 1	0x00	Attente de ControlCommand.ReadyForRT_CLASS_3 explicite
	0x01	ControlCommand.ReadyForRT_CLASS_3 implicite
Bit 2 – 15	—	Doit être mis conformément à 3.4.2.2.

Tableau 431 – ControlBlockProperties conjointement avec les autres valeurs du champ ControlCommand

Position de bit	Valeur (hexadécimale)	Signification
Bit 0 – 15	—	Doit être mis conformément à 3.4.2.2.

5.2.4.55 Codage du champ ControlCommand

Le codage de ce champ doit être conforme à 3.4.2.4. Un seul des bits suivants doit être mis. Les bits individuels doivent avoir la signification ci-dessous:

Bit 0: ControlCommand.PrmEnd

Ce champ doit être mis conformément au Tableau 432.

Tableau 432 – ControlCommand.PrmEnd

Valeur (hexadécimale)	Signification	Utilisation
0x00	Pas de PrmEnd	—
0x01	L'appareil de commande E/S a terminé la transmission du paramètre de démarrage enregistré.	IODControlReq

Bit 1: ControlCommand.ApplicationReady

Ce champ doit être mis conformément au Tableau 433.

Tableau 433 – ControlCommand.ApplicationReady

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucun Application Ready	—
0x01	L'appareil E/S a terminé le démarrage de son application ou d'un nouveau module, ou un sous-module a été connecté et est prêt à fonctionner.	IOXControlReq

Bit 2: ControlCommand.Release

Ce champ doit être mis conformément au Tableau 434.

Tableau 434 – ControlCommand.Release

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucun Release	—
0x01	L'appareil de commande E/S met fin à la relation entre applications.	IODReleaseReq

Bit 3: ControlCommand.Done

Ce champ doit être mis conformément au Tableau 435.

Tableau 435 – ControlCommand.Done

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucun Done	—
0x01	Acquittement envoyé.	IODReleaseRes, IOXControlRes, IODControlRes

Bit 4: ControlCommand.ReadyForCompanion

Ce champ doit être mis conformément au Tableau 436.

Tableau 436 – ControlCommand ReadyForCompanion

Valeur (hexadécimale)	Signification	Utilisation
0x00	Indisponible pour l'accompagnement	—
0x01	L'appareil E/S a mis fin au démarrage de son application et est prêt pour la relation entre applications d'accompagnement.	IOXControlReq

Bit 5: ControlCommand.ReadyForRT_CLASS_3

Ce champ doit être mis conformément au Tableau 437.

Tableau 437 – ControlCommand.ReadyForRT_CLASS_3

Valeur (hexadécimale)	Signification	Utilisation
0x00	Indisponible pour RT_CLASS_3	—
0x01	L'appareil E/S a mis fin au démarrage de son application et est prêt pour la communication RT_CLASS_3.	IOXControlReq

Bit 6: ControlCommand.PrmBegin

Ce champ doit être mis conformément au Tableau 438.

Tableau 438 – ControlCommand.PrmBegin

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucun PrmBegin	—
0x01	L'appareil de commande E/S commence la transmission du paramètre de démarrage stocké.	IODControlReq

Bit 7 à 15: ControlCommand.reserved

Ce champ doit être mis à zéro.

5.2.4.56 Codage du champ DataDescription

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 1: DataDescription.Type

Ce champ doit être mis conformément au Tableau 439.

Tableau 439 – DataDescription.Type

Valeur (hexadécimale)	Signification
0x00	Réserve
0x01	Entrée
0x02	Sortie
0x03	Réserve

Bit 2-15: DataDescription.reserved

Ce champ doit être mis à zéro.

5.2.4.57 Codage du champ DataLength

Ce champ doit être codé en type de données Unsigned16. Les valeurs doivent contenir la longueur de C_SDU ou de CSF_SDU conformément au Tableau 440.

NOTE Le champ DataLength est au minimum la somme de tous les DataItems. Il peut être plus grand.

Tableau 440 – Valeurs de DataLength

Valeur (décimale)	Domaine	Signification
40 – 1 440	RT_CLASS_1, RT_CLASS_2 et RT_CLASS_3	Plage valide
12 – 1 440	RT_CLASS_UDP	Plage valide
autre	—	Réservé

5.2.4.58 Codage du champ GAP

Ce champ doit être codé en type de données Unsigned8. Les valeurs doivent être mises à zéro.

5.2.4.59 Codage du champ Padding

Ce champ doit être codé en type de données Unsigned8. Les valeurs doivent être mises à zéro.

5.2.4.60 Codage du champ RTCPadding

Ce champ doit être codé en type de données Unsigned8.

5.2.4.61 Codage du champ RWPadding

Ce champ doit être codé en type de données Unsigned8. Les valeurs doivent être mises à zéro.

5.2.4.62 Codage du champ SendClockFactor

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 441. La base de temps est de 31,25 µs.

Tableau 441 – Valeurs de SendClockFactor

Valeur (décimale)	LengthOfPeriod	Signification
0	—	Réservé
1	31,25 µs	Facultatif
2	62,5 µs	Facultatif
3	93,75 µs	Facultatif
4	125 µs	Facultatif
5 – 7	—	Facultatif
8	250 µs	Facultatif
9 – 15	—	Facultatif
16	500 µs	Facultatif
17 – 31	—	Facultatif
32	1 ms	Obligatoire
33 – 63	—	Facultatif
64	2 ms	Facultatif
65 – 127	—	Facultatif
128	4 ms	Facultatif
129 – 65 535	—	Réservé

Les combinaisons valides de SendClockFactor avec ReductionRatio sont présentées dans le Tableau 442.

LengthOfPeriod correspond à une plage de valeurs comprise entre 31,25 µs et 4 000 µs. Une trame Ethernet contient au maximum 1 518 ou 1 522 octets. Pour transférer ce type de trame à une vitesse de 100 Mbit/s et de 1 Gbit/s, il convient que LengthOfPeriod soit respectivement supérieur à 122,24 µs et 12,224 µs.

La valeur de LengthOfPeriod doit être calculée à l'aide de la Formule (52).

$$\text{LengthOfPeriod} = \text{SendClockFactor} \times 31,25 \mu\text{s} \quad (52)$$

où

LengthOfPeriod est la longueur de la période

SendClockFactor facteur de l'horloge d'envoi, la longueur d'une phase

Utilisé dans le bloc PDSyncData, ce champ doit uniquement être valide si le champ SyncProperties.SyncID contient la valeur 0.

5.2.4.63 Codage du champ ReductionRatio

Ces champs doivent être codés en type de données Unsigned16 avec les valeurs conformes au Tableau 442, au Tableau 443 et au Tableau 444.

Les valeurs admises doivent être calculées conjointement avec le Tableau 441 à l'aide de la Formule (53).

Tableau 442 – Valeurs de ReductionRatio pour RT_CLASS_1 et RT_CLASS_2

Valeur (décimale)	Signification	Utilisation
1	Obligatoire	Transmission et réception à chaque horloge d'envoi
2	Obligatoire	Transmission et réception toutes les deux horloges d'envoi
4	Obligatoire	Transmission et réception toutes les quatre horloges d'envoi
8	Obligatoire	Transmission et réception toutes les huit horloges d'envoi
16	Obligatoire	Transmission et réception toutes les seize horloges d'envoi
32	Obligatoire	Transmission et réception toutes les 32 horloges d'envoi
64	Obligatoire	Transmission et réception toutes les 64 horloges d'envoi
128	Obligatoire	Transmission et réception toutes les 128 horloges d'envoi
256	Obligatoire	Transmission et réception toutes les 256 horloges d'envoi
512	Obligatoire	Transmission et réception toutes les 512 horloges d'envoi
16 385 – 65 535	Réservé	—
autre	Facultatif	—

Tableau 443 – Valeurs de ReductionRatio pour RT_CLASS_3

Valeur (décimale)	Signification	Utilisation
1	Obligatoire	Transmission et réception à chaque horloge d'envoi
2	Obligatoire	Transmission et réception toutes les deux horloges d'envoi
4	Obligatoire	Transmission et réception toutes les quatre horloges d'envoi
8	Obligatoire	Transmission et réception toutes les huit horloges d'envoi
16	Obligatoire	Transmission et réception toutes les seize horloges d'envoi
16 385 – 65 535	Réservé	—
autre	Facultatif	—

Tableau 444 – Valeurs de ReductionRatio pour RT_CLASS_UDP

Valeur (décimale)	Signification	Utilisation
1	Facultatif	Transmission et réception à chaque horloge d'envoi
2	Facultatif	Transmission et réception toutes les deux horloges d'envoi
4	Facultatif	Transmission et réception toutes les quatre horloges d'envoi
8	Facultatif	Transmission et réception toutes les huit horloges d'envoi
16	Facultatif	Transmission et réception toutes les seize horloges d'envoi
32	Facultatif	Transmission et réception toutes les 32 horloges d'envoi
64	Facultatif	Transmission et réception toutes les 64 horloges d'envoi
128	Obligatoire	Transmission et réception toutes les 128 horloges d'envoi
256	Obligatoire	Transmission et réception toutes les 256 horloges d'envoi
512	Obligatoire	Transmission et réception toutes les 512 horloges d'envoi
1 024	Obligatoire	Transmission et réception toutes les 1 024 horloges d'envoi
2 048	Obligatoire	Transmission et réception toutes les 2 048 horloges d'envoi
4 096	Obligatoire	Transmission et réception toutes les 4 096 horloges d'envoi
8 192	Obligatoire	Transmission et réception toutes les 8 192 horloges d'envoi
16 384	Obligatoire	Transmission et réception toutes les 16 384 horloges d'envoi
16 385 – 65 535	Réservé	—
autre	Facultatif	—

5.2.4.64 Codage du champ Phase

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 445. Ils doivent être inférieurs ou égaux au ReductionRatio connexe.

Tableau 445 – Valeurs de Phase

Valeur (décimale)	Signification	Utilisation
0	Réservé	—
1 – 16 384	Obligatoire	Mais ne doit pas dépasser la valeur de ReductionRatio
16 385 – 65 535	Réservé	—

5.2.4.65 Codage du champ Sequence

Ces champs doivent être codés en type de données Unsigned16 conformément au Tableau 446.

Tableau 446 – Valeurs de Sequence

Valeur (hexadécimale)	Signification	Utilisation
0	Obligatoire, séquence non définie	Les PPM utilisent une séquence aléatoire pour chaque phase
0x0001 – 0xFFFF	Facultatif, séquence définie	Les PPM utilisent une séquence donnée pour chaque phase

NOTE Pour RT_CLASS_3, il y a une séquence donnée utilisée, même si le paramètre Sequence est mis à zéro.

5.2.4.66 Codage du champ DataHoldFactor

Ces champs doivent être codés en tant que type de données Unsigned16. La base de temps est égale au produit de SendClockFactor par le ReductionRatio du consommateur surveillé.

DataHoldTime doit être calculé à l'aide de la Formule (53).

$$\text{DataHoldTime} = \text{DataHoldFactor} \times \text{SendClockFactor} \times \text{ReductionRatio} \times 31,25 \mu\text{s} \quad (53)$$

où

DataHoldTime est le temps de maintien des données
 DataHoldFactor est le facteur de maintien des données
 SendClockFactor est le facteur de l'horloge d'envoi
 ReductionRatio est le rapport de réduction

La plage de valeurs de Data-RTC-PDU est comprise entre 0x0003 et 0x1E00. DataHoldTime doit être inférieur ou égal à 1,92 s.

La plage de valeurs d'UDP-RTC-PDU est comprise entre 0x0003 et 0xF000. DataHoldTime doit être inférieur ou égal à 61,44 s.

Pour une trame, l'utilisation de DataHoldFactor est donnée au Tableau 447.

Tableau 447 – DataHoldFactor d'une trame

Valeur (hexadécimale)	Signification	Description
0x0000	Réservé	—
0x0001 – 0x0002	Facultatif	La valeur 1 met fin à la relation entre applications d'une trame manquée, la "valeur 2" met fin à la relation entre applications de deux trames manquées.
0x0003 – 0x00FF	Obligatoire	Une expiration du délai met fin à la relation entre applications.
0x0100 – 0x1E00	Facultatif	Une expiration du délai met fin à la relation entre applications.
0x1E01 – 0xFFFF	Réservé	—

Pour une sous-trame, l'utilisation de DataHoldFactor est donnée au Tableau 448.

Tableau 448 – DataHoldFactor d'une Sous-trame

Valeur (hexadécimale)	Signification	Description
0x0000	Réservé	—
0x0001 – 0x0002	Facultatif	La valeur 1 met fin à la relation entre applications d'une trame manquée, la "valeur 2" met fin à la relation entre applications de deux trames manquées.
0x0003 – 0x001F	Obligatoire	Une expiration du délai met fin à la relation entre applications.
0x0020 – 0x1E00	Facultatif	Une expiration du délai met fin à la relation entre applications.
0x1E01 – 0xFFFF	Réservé	—

5.2.4.67 Codage du champ FrameSendOffset

Ce champ doit être codé en type de données Unsigned32 et mis conformément au Tableau 449, à la Figure 82 et à la Formule (54). La base de temps est de 1 ns.

NOTE L'espace minimal prévu entre deux trames RT_CLASS_3 est de 1 120 ns.

Tableau 449 – Valeurs de FrameSendOffset

Valeur (décimale)	Signification	Utilisation
0 – 0x00001388	Décalage d'envoi relatif par rapport au début du cycle correspondant	Facultatif
0x00001389 – 0x003B28FF	Décalage d'envoi relatif par rapport au début du cycle correspondant	Obligatoire pour RT_CLASS_3 Facultatif pour RT_CLASS_1, RT_CLASS_2 et RT_CLASS_UDP
0x003B2900 – 0x003D08FF	Décalage d'envoi relatif par rapport au début du cycle correspondant Utilisable uniquement si la taille de trame répond à la Formule (54)	Obligatoire pour RT_CLASS_3 Facultatif pour RT_CLASS_1, RT_CLASS_2 et RT_CLASS_UDP
0x003D0900 – 0xFFFFF0FE	Réservé	Non utilisé
0xFFFFF0FF	Service au mieux Les machines de protocole du fournisseur envoient la trame dès que possible	Obligatoire Pas utilisé pour PDIRFrameData

$$\text{FrameSendOffset}[\text{Frame}] + \text{Duration}[\text{Frame}] < \text{SendClockFactor} \times 31,25 \mu\text{s} \quad (54)$$

où

FrameSendOffset	est le décalage d'envoi de trame
Frame	est la trame actuelle
Duration	est la durée
SendClockFactor	est le facteur de l'horloge d'envoi, la longueur d'une phase

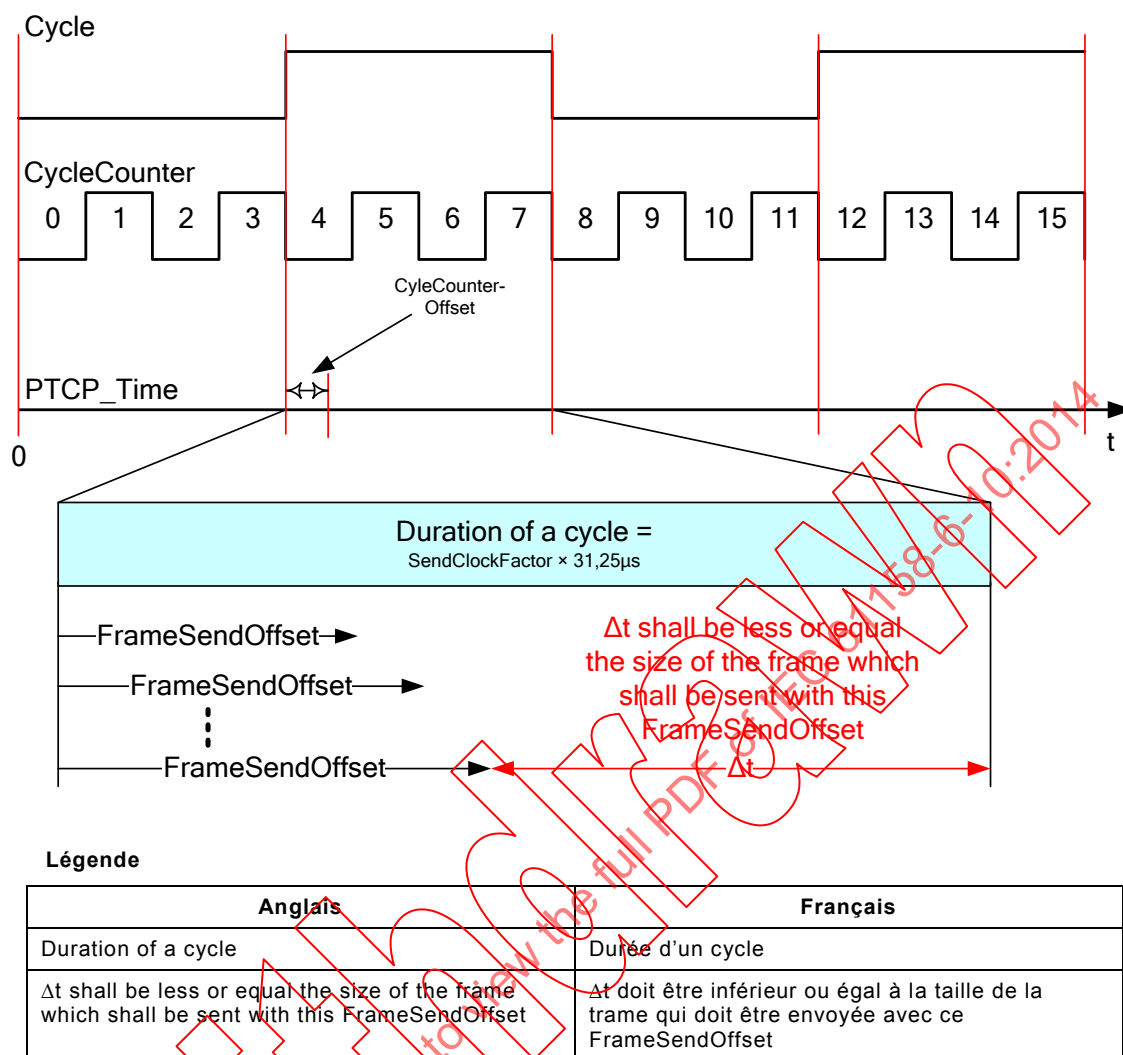


Figure 82 – FrameSendOffset par rapport à la durée d'un cycle

5.2.4.68 Codage du champ ModuleState

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 450.

Tableau 450 – ModuleState

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Aucun module	Le module n'est pas connecté, par exemple
0x0001	Module incorrect ^a	Par exemple:ModuleIdentNumber erroné
0x0002	Module correct	Le module est correct, mais au moins un sous-module est verrouillé, incorrect ou manquant
0x0003	Substitut ^b	Le module n'est pas celui demandé, mais l'appareil E/S a pu s'adapter par ses propres connaissances
0x0004 – 0xFFFF	Réservé	—
^a Dans la mesure du possible, il convient d'éviter ce codage. La substitution peut être "Substitute" pour le module et "Wrong" (WR) pour chaque sous-module. Voir Tableau 457 pour des combinaisons valides. ^b Ce codage peut être utilisé en combinaison à "SubmoduleState.IdentInfo:= Wrong (WR)".		

5.2.4.69 Codage du champ SubmoduleState

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

5.2.4.69.1 Codage si SubmoduleState.FormatIndicator == 1

Bit 0 – 2: SubmoduleState.AddInfo

Ce champ doit être mis conformément au Tableau 451.

Tableau 451 – SubmoduleState.AddInfo

Valeur (hexadécimale)	Signification	Description
0x00	Aucune	—
0x01	Prise de contrôle non admise	Ce sous-module n'est pas disponible pour la prise de contrôle par IOSAR.
0x02 – 0x07	Réservé	—

Bit 3: SubmoduleState.QualifiedInfo

Ce champ doit être mis conformément au Tableau 452.

Tableau 452 – SubmoduleState.QualifiedInfo

Valeur (hexadécimale)	Signification	Description
0x00	Aucun QualifiedInfo disponible	Aucun canal du sous-module ne contient QualifiedChannelDiagnosis
0x01	QualifiedInfo disponible	Au moins un canal du sous-module contient QualifiedChannelDiagnosis

Bit 4: SubmoduleState.MaintenanceRequired

Ce champ doit être mis conformément au Tableau 453.

Tableau 453 – SubmoduleState.MaintenanceRequired

Valeur (hexadécimale)	Signification	Description
0x00	No MaintenanceRequired available	Aucun canal de sous-module ne requiert de maintenance
0x01	MaintenanceRequired available	Au moins un canal de sous-module requiert une maintenance

Bit 5: SubmoduleState.MaintenanceD demanded

Ce champ doit être mis conformément au Tableau 454.

Tableau 454 – SubmoduleState.MaintenanceD demanded

Valeur (hexadécimale)	Signification	Description
0x00	Aucun MaintenanceD demanded disponible	Aucun canal du sous-module n'exige de maintenance
0x01	MaintenanceD demanded disponible	Au moins un canal du sous-module exige une maintenance

Bit 6: SubmoduleState.DiagInfo

Ce champ doit être mis conformément au Tableau 455.

Tableau 455 – SubmoduleState.DiagInfo

Valeur (hexadécimale)	Signification	Description
0x00	Aucun DiagnosisData disponible	Aucun DiagnosisData n'est disponible/enregistré pour ce sous-module.
0x01	DiagnosisData disponible	Un DiagnosisData (avec ChannelProperties.Specifier:= Appears) est disponible pour ce sous-module: il peut être lu avec les enregistrements correspondants.

Bit 7 – 10: SubmoduleState.ARInfo

Ce champ doit être mis conformément au Tableau 456.

Tableau 456 – SubmoduleState.ARInfo

Valeur (hexadécimale)	Signification	Description
0x00	Propriétaire	Cette AR est propriétaire du sous-module
0x01	ApplicationReadyPending (ARP)	Cette AR est propriétaire du sous-module, mais elle est bloquée. Vérification de paramètre en cours, par exemple
0x02	Superordinated Locked (SQ)	Cette AR n'est pas propriétaire du sous-module. Elle est bloquée par des moyens superordonnés
0x03	Verrouillé par l'IOC	Cette AR n'est pas propriétaire du sous-module. Elle est détenue par une autre IOAR
0x04	Verrouillé par l'IOS	Cette AR n'est pas propriétaire du sous-module. Elle est détenue par une autre IOSAR
0x05 – 0x0F	Réservé	—

Bit 11 – 14: SubmoduleState.IdentInfo

Ce champ doit être mis conformément au Tableau 457.

Tableau 457 – SubmoduleState.IdentInfo

Valeur (hexadécimale)	Signification	Utilisable conjointement avec "Wrong module" de ModuleState
0x00	OK	Non
0x01	Substitute (SU)	Non
0x02	Wrong (WR)	Oui
0x03	NoSubmodule (NO)	Oui
0x04 – 0x0F	Réservé	—

Bit 15: SubmoduleState.FormatIndicator

Ce champ doit être mis conformément au Tableau 458.

Tableau 458 – SubmoduleState.FormatIndicator

Valeur (hexadécimale)	Signification	Utilisation
0x00	Le codage utilise SubmoduleState.Detail	Doit être pris en charge par un appareil de commande E/S et un superviseur d'E/S.
0x01	Le codage utilise SubmoduleState.IdentInfo, .ARInfo et .AddInfo	Doit être utilisé par un appareil E/S, un appareil de commande E/S et un superviseur d'E/S.

5.2.4.69.2 Codage si SubmoduleState.FormatIndicator == 0**Bit 0 – 14: SubmoduleState.Detail**

Ce champ doit être mis conformément au Tableau 459.

Tableau 459 – SubmoduleState.Detail

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Aucun sous-module	—
0x0001	Sous-module incorrect	—
0x0002	Verrouillé par un appareil de commande E/S	—
0x0003	Réservé	—
0x0004	Application prête en attente	Doit uniquement être utilisé conjointement avec la demande IOXControl (application prête)
0x0005	Réserve	—
0x0006	Réserve	—
0x0007	Substitut	Le sous-module n'est pas celui demandé, mais l'appareil E/S a pu s'adapter Dans cas, il convient que l'appareil E/S s'adapte (à la nouvelle longueur de l'entrée ou de la sortie, par exemple)
0x0008 – 0x7FFF	Réserve	—

NOTE Le champ SubmoduleState fait uniquement l'objet d'une réponse en cas de différence entre les données de configuration prévues et les données réelles. SubmoduleState "GOOD" n'est pas défini car, pour ce type de sous-modules, aucun état n'est signalé dans ModuleDiffBlock.

Bit 15: SubmoduleState.FormatIndicator

Ce champ doit être mis conformément au Tableau 458.

5.2.4.70 Codage du champ SubmoduleProperties

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 1: SubmoduleProperties.Type

Ce champ doit être mis conformément au Tableau 460.

Tableau 460 – SubmoduleProperties.Type

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucune donnée d'entrée et de sortie	Sous-module sans donnée E/S , traité comme une entrée sans donnée (pour IOCS/IOPS), un Input DataDescription Block suit
0x01	Données d'entrée	Sous-module avec données d'entrée, un Input DataDescription Block suit
0x02	Données de sortie	Sous-module avec données de sortie, un Output DataDescription Block suit
0x03	Données d'entrée et de sortie	Sous-module avec données d'entrée et de sortie, un Input et un Output DataDescription Block suit

Bit 2: SubmoduleProperties.SharedInput

Ce champ doit être mis conformément au Tableau 461.

Tableau 461 – SubmoduleProperties.SharedInput

Valeur (hexadécimale)	Signification	Utilisation
0x00	Appareil de commande E/S	Appareil de commande E/S Peut être utilisé avec toutes les valeurs possibles de SubmoduleProperties.Type.
0x01	Appareil de commande E/S partagé	Seul l'appareil de commande E/S partagé doit utiliser cette valeur. L'appareil de commande E/S partagé ne reçoit pas d'alarmes et se limite à l'accès en lecture. La valeur ne doit pas être utilisée avec SubmoduleProperties.Type=0x02.

Bit 3: SubmoduleProperties.ReduceInputSubmoduleDataLength

Ce champ doit être mis conformément au Tableau 462.

Tableau 462 – SubmoduleProperties.ReduceInputSubmoduleDataLength

Valeur (hexadécimale)	Signification	Utilisation
0x00	Prévu	Utiliser l'entrée SubmoduleDataLength prévue pour I-CR
0x01	Zéro	Réduire l'entrée SubmoduleDataLength à zéro pour I-CR

Bit 4: SubmoduleProperties.ReduceOutputSubmoduleDataLength

Ce champ doit être mis conformément au Tableau 463.

Tableau 463 – SubmoduleProperties.ReduceOutputSubmoduleDataLength

Valeur (hexadécimale)	Signification	Utilisation
0x00	Prévu	Utiliser la sortie SubmoduleDataLength prévue pour O-CR
0x01	Zéro	Réduire la sortie SubmoduleDataLength à zéro pour O-CR

Bit 5: SubmoduleProperties.DiscardIOXS

Ce champ doit être mis conformément au Tableau 464.

Tableau 464 – SubmoduleProperties.DiscardIOXS

Valeur (hexadécimale)	Signification	Utilisation
0x00	Prévu	Fusionner tous les IOXS prévus pour ce sous-module dans les trames des CR correspondants
0x01	Zéro	Ecarter tous les IOXS prévus pour ce sous-module dans les trames des CR correspondants

Bit 6 – 7: SubmoduleProperties.reserved_1

Ce champ doit être mis conformément à 3.4.2.2.

Bit 8 – 15: SubmoduleProperties.reserved_2

Ce champ doit être mis à zéro.

5.2.4.71 Codage du champ ModuleProperties

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 7: ModuleProperties.reserved_1

Ce champ doit être mis conformément à 3.4.2.2.

Bit 8 – 15: ModuleProperties.reserved_2

Ce champ doit être mis à zéro.

5.2.4.72 Codage du champ SubstitutionMode

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 465.

Tableau 465 – SubstitutionMode

Valeur (hexadécimale)	Signification	Utilisation
0x0000	ZERO	Les sorties sont mises à zéro ou inactives
0x0001	Dernière valeur	Conserver la dernière valeur d'application valide
0x0002	Valeur de remplacement	Met la sortie à la valeur de remplacement configurée
0x0003 – 0x00FF	Réservé	—
0x0100 – 0x01FF	Réservé aux profils	Doit être utilisé pour les profils
0x0200 – 0xFFFF	Réservé	—

5.2.4.73 Codage du champ SubstituteActiveFlag

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 466.

Tableau 466 – SubstituteActiveFlag

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Fonctionnement	Les sorties sont mises conformément au processus de contrôle normal (substitut inactif)
0x0001	Substitut	Les sorties sont mises conformément à la fonction de substitution (substitut actif)
0x0002 – 0xFFFF	Réservé	—

5.2.4.74 Codage du champ InitiatorUDPRTPort

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 467.

Tableau 467 – InitiatorUDPRTPort

Valeur (hexadécimale)	Signification	Utilisation
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Facultatif Le port peut être bloqué par d'autres protocoles
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Utilisable
0x8892	Par défaut pour l'envoi individuel IANA_PNIO_UDP_UNICAST_PORT	Recommandé
0x8893	Par défaut pour la multidiffusion IANA_PNIO_UDP_MULTICAST_PORT	Recommandé
0xC000 – 0xFFFF	IANA_FREE_PORT fait office de port dynamique et/ou privé	Utilisable

5.2.4.75 Codage du champ ResponderUDPRTPort

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 468.

Tableau 468 – ResponderUDPRTPort

Valeur (hexadécimale)	Signification	Utilisation
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Facultatif Le port peut être bloqué par d'autres protocoles
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Utilisable
0x8892	Par défaut pour l'envoi individuel IANA_PNIO_UDP_UNICAST_PORT	Recommandé
0x8893	Par défaut pour la multidiffusion IANA_PNIO_UDP_MULTICAST_PORT	Recommandé
0xC000 – 0xFFFF	IANA_FREE_PORT fait office de port dynamique et/ou privé	Utilisable

5.2.4.76 Codage du champ InitiatorRPCServerPort

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 469.

Tableau 469 – InitiatorRPCServerPort

Valeur (hexadécimale)	Signification	Utilisation
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Réservé
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Utilisable
0xC000 – 0xFFFF	IANA_FREE_PORT fait office de port dynamique et/ou privé	Recommandé

5.2.4.77 Codage du champ ResponderRPCServerPort

Ce champ doit être codé en type de données Unsigned16. Ce champ doit être codé avec les valeurs conformes au Tableau 470.

Tableau 470 – ResponderRPCServerPort

Valeur (hexadécimale)	Signification	Utilisation
0x0000 – 0x03FF	IANA_WELL_KNOWN_PORT	Réservé
0x0400 – 0xBFFF	IANA_DEFINED_PORT	Utilisable
0xC000 – 0xFFFF	IANA_FREE_PORT fait office de port dynamique et/ou privé	Recommandé

5.2.4.78 Codage du champ LocalAlarmReference

Ce champ doit être codé en type de données Unsigned16. Ce champ doit contenir la valeur de la référence de l'appareil de commande E/S à l'intérieur d'AlarmCRBlockReq et doit contenir la valeur de la référence de l'appareil E/S à l'intérieur d'AlarmCRBlockRes.

Dans ARData, ce champ contient la référence de l'appareil de commande E/S reçue par AlarmCRBlockReq.

5.2.4.79 Codage du champ RemoteAlarmReference

Ce champ doit être codé en type de données Unsigned16. Ce champ doit contenir la valeur de la référence de l'appareil E/S transmise dans AlarmCRBlockRes à l'appareil de commande E/S, dans ARData.

5.2.4.80 Codage du champ MaxAlarmDataLength

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 471.

Tableau 471 – MaxAlarmDataLength

Valeur (hexadécimale)	Signification
0x00C8	Obligatoire Taille maximale d'AlarmNotification-PDU qu'accepte l'AlarmEndpoint.
0x00C9 – 0x0598	Facultatif Taille maximale d'AlarmNotification-PDU qu'accepte l'AlarmEndpoint.
autre	Réservé

Un AlarmEndpoint ne doit pas créer des AlarmNotification-PDU plus grands que ceux qui ont été négociés pendant l'établissement d'AR.

5.2.4.81 Codage du champ ParameterServerProperties

Ce champ doit être codé en type de données Unsigned32. Il est réservé à un usage ultérieur.

5.2.5 Section de codage relative à ARVendorBlock

5.2.5.1 Codage du champ APStructureIdentifieur

Ce champ, défini sous forme d'un numéro administratif, doit être codé en type de données Unsigned16 conformément au Tableau 472 et au Tableau 473.

Tableau 472 – APStructureIdentifieur avec API := 0

Valeur (hexadécimale)	Signification
0 – 0x7FFF	Spécifique au vendeur Cette clé définit, en conjonction avec l'API clé, la structure de Data dans les éléments ARVendorBlockReq et ARVendorBlockRes.
0x8000	Règles d'identification étendues Utilisé pour le profil commun "Control to control communication"
0x8001 – 0xFFFF	Numéro administratif Cette clé définit, en conjonction avec l'API clé, la structure de Data dans les éléments ARVendorBlockReq et ARVendorBlockRes.

Tableau 473 – APStructureIdentifieur avec API != 0

Valeur (hexadécimale)	Signification
0 – 0xFFFF	Numéro administratif pour les profils d'application Cette clé définit, en conjonction avec l'API clé, la structure de Data dans les éléments ARVendorBlockReq et ARVendorBlockRes.

5.2.5.2 Codage du champ ExtendedIdentificationUUID

Ce champ doit être codé en type de données UUID.

5.2.5.3 Codage du champ ExtendedIdentificationVersionHigh

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 474.

Tableau 474 – ExtendedIdentificationVersionHigh

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	—
0x01	Version 1	Indique Version 1
0x02 – 0xFF	Version 2 à version 255	Indique la version 2 à la version 255

5.2.5.4 Codage du champ ExtendedIdentificationVersionLow

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 475.

Tableau 475 – ExtendedIdentificationVersionLow

Valeur (hexadécimale)	Signification	Utilisation
0x00	Version 0	Indique la version 0
0x01	Version 1	Indique la version 1
0x02 – 0xFF	Version 2 à version 255	Indique la version 2 à la version 255

5.2.6 Section de codage relative à PNIOStatus

5.2.6.1 Généralités

En general, la valeur ErrorCode=0, ErrorDecode=0, ErrorCode1=0 et ErrorCode2=0 doit être utilisée pour indiquer "correct".

De plus, en cas de combinaison illégale de paramètres d'adresse dans une IODReadReq, la valeur ErrorCode="IODReadRes", ErrorDecode="PNIORW", ErrorCode1="access-invalid area" et ErrorCode2 peut être utilisée pour indiquer le paramètre défectueux.

NOTE Une combinaison d'adresses illégale est par exemple TargetARUID==zero et ARUID==zero dans le service ReadExpectedIdentification.

5.2.6.2 Codage du champ ErrorCode

Ce champ doit être codé en type de données Unsigned8. L'utilisation est définie au Tableau 476.

Tableau 476 – Valeurs de ErrorCode pour des réponses négatives

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	Cas particulier "No Error": ErrorCode = 0, ErrorDecode = 0, ErrorCode1 = 0, ErrorCode2 = 0
0x01 – 0x1F	Réservé	—
0x20 – 0x3F	Spécifique à un fabricant	Au sein de LogBookData
0x40 – 0x80	Réservé	—
0x81	PNIO	Utilisé pour toutes les erreurs qui ne sont pas traitées autre part.
0x82 – 0xCE	Réservé	—
0xCF	RTA error	Au sein de ERR-RTA-PDU et de UDP-RTA-PDU
0xD0 – 0xD9	Réservé	—
0xDA	AlarmAck	Au sein de DATA-RTA-PDU et de UDP-RTA-PDU
0xDB	IODConnectRes	Au sein de CL-RPC-PDU
0xDC	IODReleaseRes	Au sein de CL-RPC-PDU
0xDD	IODControlRes, IOXControlRes	Au sein de CL-RPC-PDU
0xDE	IODReadRes	Au sein de CL-RPC-PDU uniquement utilisé avec ErrorDecode=PNIORW
0xDF	IODWriteRes	Au sein de CL-RPC-PDU uniquement utilisé avec ErrorDecode=PNIORW

Valeur (hexadécimale)	Signification	Utilisation
0xE0 – 0xEF	Réservé	—
0xF0 – 0xFF	Réservé	—

5.2.6.3 Codage du champ ErrorDecode

Ce champ doit être codé en type de données Unsigned8. L'utilisation est définie au Tableau 477.

Tableau 477 – Valeurs de ErrorDecode

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	Voir Tableau 476
0x01 – 0x7F	Réservé	—
0x80	PNIORW ^a	Utilisé en contexte avec des codes d'erreur utilisateur des services Read et Write
0x81	PNIO	Utilisé en contexte avec d'autres services ou des erreurs internes (par exemple, erreurs RPC)
0x82	Spécifique à un fabricant	Utilisé dans le contexte de LogBookData
0x83 – 0xFF	Réservé	—
^a Equivalent à la CEI 61158-6-3 (DPV1)		

5.2.6.4 Codage des champs ErrorCode1 et ErrorCode2

5.2.6.4.1 Généralités

Ces champs doivent être codés en type de données Unsigned8.

Le champ ErrorDecode définit le codage et la signification de ErrorCode1 et ErrorCode2.

5.2.6.4.2 Valeur PNIORW d'ErrorDecode

La valeur PNIORW de l'ErrorDecode indique que les paramètres ErrorCode1 doivent être mis conformément au Tableau 478. L'ErrorCode1 se compose d'ErrorClass et d'ErrorDecode.

Tableau 478 – Codage d'ErrorCode1 avec ErrorDecode PNIORW

ErrorCode1		
ErrorClass (décimale) Bit7 – 4	Signification	ErrorCode (décimale) Bit3 – 0
0 à 9	Non spécifié	Réservé

ErrorCode1		
ErrorClass (décimale) Bit7 – 4	Signification	ErrorCode (décimale) Bit3 – 0
10	Application	0 = erreur de lecture 1 = erreur d'écriture 2 = échec de module 3 = non spécifié 4 = non spécifié 5 = non spécifié 6 = non spécifié 7 = occupé 8 = conflit de versions 9 = fonctionnalité non prise en charge 10 = Spécifique à un utilisateur 1 11 = Spécifique à un utilisateur 2 12 = Spécifique à un utilisateur 3 13 = Spécifique à un utilisateur 4 14 = Spécifique à un utilisateur 5 15 = Spécifique à un utilisateur 6
11	Accès	0 = index non valide 1 = erreur de longueur d'écriture 2 = emplacement / sous-emplacement non valide 3 = conflit de types 4 = zone / API non valide 5 = conflit d'états 6 = accès non permis 7 = plage non valide 8 = paramètre non valide 9 = type non valide 10 = sauvegarde 11 = Spécifique à un utilisateur 7 12 = Spécifique à un utilisateur 8 13 = Spécifique à un utilisateur 9 14 = Spécifique à un utilisateur 10 15 = Spécifique à un utilisateur 11

ErrorCode1		
ErrorClass (décimale) Bit7 – 4	Signification	ErrorCode (décimale) Bit3 – 0
12	Ressource	0 = conflit de contraintes de lecture 1 = conflit de contraintes d'écriture 2 = ressource occupée 3 = ressource indisponible 4 = non spécifié 5 = non spécifié 6 = non spécifié 7 non spécifié 8 = Spécifique à un utilisateur 12 9 = Spécifique à un utilisateur 13 10 = Spécifique à un utilisateur 14 11 = Spécifique à un utilisateur 15 12 = Spécifique à un utilisateur 16 13 = Spécifique à un utilisateur 17 14 = Spécifique à un utilisateur 18 15 = Spécifique à un utilisateur 19
13 à 15	User specific	Spécifique à un utilisateur
NOTE Les valeurs "Non spécifiées" sont utilisées pour donner des codes de la version originelle et sont destinées à être transmises sans être modifiées à l'application.		

La valeur PNORW de l'ErrorDecode indique que le paramètre ErrorCode2 doit être mis conformément au Tableau 479.

Tableau 479 – Codage de ErrorCode2 avec PNORW d'ErrorDecode

ErrorCode2 (hexadécimale)	Signification	Utilisation
0x00 - 0xFF	User specific	—

5.2.6.4.3 Valeur PNIO d'ErrorDecode

La valeur PNIO de l'ErrorDecode indique que le paramètre ErrorCode1 doit être mis conformément au Tableau 480, au Tableau 481, au Tableau 482, au Tableau 483, au Tableau 484, au Tableau 485, au Tableau 486 et aux définitions de 4.10.2.3.6.

Tableau 480 – Valeurs d'ErrorCode1 et d'ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 1)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
0	Réservé	0 - 255	Réservé
1	Erreur de Paramètre de Connexion ARBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		13	Erreur dans le Paramètre NameOfStation
		14 - 255	Réservé
2	Erreur de Paramètre de Connexion IOCRBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		28	Erreur dans la production du Paramètre IOCSFrameOffset
		29 - 255	Réservé
3	Erreur de Paramètre de Connexion ExpectedSubmoduleBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		16	Erreur dans le Paramètre LengthIOCS production
		17 - 255	Réservé
4	Erreur de Paramètre de Connexion AlarmCRBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		15	Erreur dans le Paramètre AlarmCRTagHeaderLow
		16 - 255	Réservé
5	Erreur de Paramètre de Connexion PrmServerBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		8	Erreur dans le Paramètre ParameterServerStationName
		9 - 255	Réservé
6	Erreur de Paramètre de Connexion MCRBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		8	Erreur dans le Paramètre ProviderStationName
		9 - 255	Réservé
7	Erreur de Paramètre de Connexion ARRPCBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		4	Erreur dans le Paramètre InitiatorRPCServerPort
		5 - 255	Réservé
8	Erreur de Paramètre d'Enregistrement de Lecture Ecriture Enregistrement défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		12	Erreur dans le Paramètre TargetARUUIID
		13 - 255	Réservé

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
9	Erreur de Paramètre de Connexion IRInfoBlock défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		5	Erreur dans le Paramètre IRDataUUID
		6 - 255	Réservé
10	Erreur de Paramètre de Connexion SRInfoBlock défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		5	Erreur dans le Paramètre SRProperties
		6 - 255	Réservé
11	Erreur de Paramètre de Connexion ARFSUBBlock défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		5	FastStartUpBlock
		6 - 255	Réservé
12	Erreur de Paramètre de Connexion ARVendorBlockReq défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		5	Erreur dans le Paramètre API
		6	Erreur dans le Paramètre Data*
		7 - 255	Réservé
		...	...
13 - 19	Réservé	0 - 255	Réservé
20	Erreur de Paramètre IODControl ControlBlockConnect défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		9	Erreur dans le Paramètre ControlBlockProperties
		10 - 255	Réservé
21	Erreur de Paramètre IODControl ControlBlockPlug défectueux	0	Erreur dans le Paramètre BlockType
		...	...
		9	Erreur dans le Paramètre ControlBlockProperties
22	Erreur de Paramètre IOXControl ControlBlockConnect défectueux après l'établissement d'une connexion	0	Erreur dans le Paramètre BlockType
		...	...
		7	Erreur dans le Paramètre ControlBlockProperties
23	Erreur de Paramètre IOXControl ControlBlockPlug défectueux après une alarme de connexion	8 - 255	Réservé
		0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
23		7	Erreur dans le Paramètre ControlBlockProperties

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		8 - 255	Réservé
24	Erreur de Paramètre IODControl ControlBlockPrmBegin défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		9	Erreur dans le Paramètre ControlBlockProperties
		10 - 255	Réservé
25	Erreur de Paramètre IODControl SubmoduleListBlock défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		7	Erreur dans le Paramètre SubslotNumber
		8 - 255	Réservé
26 - 39	Réservé	0 - 255	Réservé
40	Erreur de Paramètre de Libération ReleaseBlock défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		7	Erreur dans le Paramètre ControlBlockProperties
		8 - 255	Réservé
41 - 49	Réservé	0 - 255	Réservé
50	Erreur de Paramètre de Réponse ARBlockRes défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		8	Erreur dans le Paramètre ResponderUDPRTPort
		9 - 255	Réservé
51	Erreur de Paramètre de Réponse IOCRBlockRes défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		6	Erreur dans le Paramètre FrameID
		7 - 255	Réservé
52	Erreur de Paramètre de Réponse AlarmCRBlockRes défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		6	Erreur dans le Paramètre MaxAlarmDataLength
		7 - 255	Réservé
53	Erreur de Paramètre de Réponse ModuleDiffBlock défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		13	Erreur dans le Paramètre SubmoduleState
		14 - 255	Réservé
54	Erreur de Paramètre de Réponse	0	Erreur dans le Paramètre BlockType

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
	ARRPCBlockRes défectueux	1	Erreur dans le Paramètre Length
		...	...
		4	Erreur dans le Paramètre ResponderRPCServerPort
		5 - 255	Réservé
55	Erreur de Paramètre de Réponse ARServerBlockRes défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		4	Erreur dans le Paramètre CMResponderStationName
		5 - 255	Réservé
56	Erreur de Paramètre de Réponse ARVendorBlockRes défectueux	0	Erreur dans le Paramètre BlockType
		1	Erreur dans le Paramètre Length
		...	...
		5	Erreur dans le Paramètre API
		6	Erreur dans le Paramètre Data*
		7 - 255	Réservé
57 - 59	Réservé	0 - 255	Réservé

Tableau 481 – Valeurs d'ErrorCode1 et ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 2 – acquittement d'alarme)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
60	Codes d'Erreur AlarmAck	0	Type d'alarme non pris en charge
		1	Etat du Sous-module faux
		2	IOCARSR Sauvegarde – Alarme non exécutée
		3 - 255	Réservé

Tableau 482 – Valeurs d'ErrorCode1 et ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 3 – machines)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
65	ALPMI	0	Etat non valide
		1	ACK-PDU incorrect
		2	Non valide
		3	Etat incorrect
		4 - 255	Réservé
66	ALPMR	0	Etat non valide
		1	Mauvais PDU de notification
		2	Non valide

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		3	Etat incorrect
		4 - 255	Réservé
67	LMPM	0 - 255	Utilisation:voir machines de protocole et stocké en tant qu'entrées de LogBook
68	MAC	0 - 255	Utilisation:voir machines de protocole et stocké en tant qu'entrées de LogBook
69	RPC	0 - 255	voir Tableau 486
70	APMR	0	Etat non valide
		1	LMPM a signalé une erreur
		2 - 255	Réservé
71	APMS	0	Etat non valide
		1	LMPM a signalé une erreur
		2	Temporisation
		3 - 255	Réservé
72	CPM	0	Etat non valide
		1 - 255	Réservé
73	PPM	0	Etat non valide
		1 - 255	Réservé
74	Utilisé par DCPUCS	0	Etat non valide
		1	LMPM a signalé une erreur
		2	Temporisation
		3 - 255	Réservé
75	Utilisé par DCPUCR	0	Etat non valide
		1	LMPM a signalé une erreur
		2 - 255	Réservé
76	Utilisé par DCPMCS	0	Etat non valide
		1	LMPM a signalé une erreur
		2 - 255	Réservé
77	Utilisé par DCPMCR	0	Etat non valide
		1	LMPM a signalé une erreur
		2 - 255	Réservé
78	FSPM	0 - 255	Utilisation:voir machines de protocole et stocké en tant qu'entrées de LogBook
79 - 99	Réservé	0 - 255	Réservé
121 - 199	Réservé	0 - 255	Réservé
221 - 252	Réservé	0 - 255	Réservé

Tableau 483 – Valeurs d'ErrorCode1 et ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 4 – appareil de commande E/S)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
62	CMCTL	0	Conflit d'états
		1	Temporisation
		2	Aucune donnée envoyée
		3	Sans ressource
		4 - 255	Réservé
63	CTLDINA (successeur du NRPM)	0	Pas de DCP actif / pas de liaison
		1	RealStationName inconnu du DNS
		2	DCP no RealStationName
		3	DCP multiple RealStationName
		4	DCP no StationName
		5	Aucune adresse IP
		6	DCP set error
		7 - 255	Réservé
100	CTLSDM	0	Etat non valide
		1	CTLSDM a signalé une erreur
		2 - 255	Réservé
101	CTLRDI	0	Etat non valide
		1	CTLRDI a signalé une erreur
		2 - 255	Réservé
102	CTLRDR	0	Etat non valide
		1	CTLRDR a signalé une erreur
		2 - 255	Réservé
103	CTLWRI	0	Etat non valide
		1	CTLWRI a signalé une erreur
		2 - 255	Réservé
104	CTLWRR	0	Etat non valide
		1	CTLWRR a signalé une erreur
		2 - 255	Réservé
105	CTLIO	0	Etat non valide
		1	CTLIO a signalé une erreur
		2 - 255	Réservé
106	CTLSU	0	Etat non valide
		1	AR add provider or consumer failed (AR échec d'ajout de fournisseur ou de consommateur)
		2	AR alarm-open failed (échec alarme ouverture)
		3	AR alarm-ack-send (envoi acquittement alarme)
		4	AR alarm-send (envoi alarme)
		5	AR alarm-ind
6 - 255	Réservé		

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
107	CTLRPC	0	Etat non valide
		1	CTLRPC a signalé une erreur
		2 - 255	Réservé
108	CTLPBE	0	Etat non valide
		1	CTLPBE a signalé une erreur
		2 - 255	Réservé
109 - 120	Réservé	0 - 255	Réservé

Tableau 484 – Valeurs d'ErrorCode1 et ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 5 – appareil E/S)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
61	CMDEV	0	Conflit d'états
		1	Ressource
		2 - 255	Utilisation: voir machines de protocole et stocké en tant qu'entrées de LogBook
64	CMRPC (successeur du RMPM)	0	ArgsLength non valide
		1	Blocs inconnus
		2	IOCR manquant
		3	Nombre de blocs incorrect (par exemple, nombre d'AlarmCRBlock incorrect)
		4	Sans ressources AR
		5	AR UUID inconnu
		6	Conflit d'états
		7	Sans Fournisseur, Consommateur ou Ressources d'AR
		8	Sans mémoire
		9	Pdev déjà possédé
		10	ARset ~~et~~ lors de l'établissement de la connexion
		11	ARset l'établissement de la connexion
		12 - 255	Réservé
200	CMSM	0	Etat non valide
		1	CMSM a signalé une erreur
		2 - 255	Réservé
201	Réservé	0 - 255	Réservé
202	CMRDR	0	Etat non valide
		1	CMRDR a signalé une erreur
		2 - 255	Réservé
203	Réservé	0 - 255	Réservé
204	CMWRR	0	Etat non valide

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		1	AR n'est pas à l'état Primary. Dans ce cas, l'écriture d'un enregistrement n'est pas autorisée.
		2	CMWRR a signalé une erreur
		3 - 255	Réservé
205	CMIO	0	Etat non valide
		1	CMIO a signalé une erreur
		2 - 255	Réservé
206	CMSU	0	Etat non valide
		1	AR add provider or consumer failed (AR échec d'ajout de fournisseur ou de consommateur)
		2	AR alarm-open failed (échec alarme ouverture)
		3	AR alarm-send (envoi alarme)
		4	AR alarm-ack-send (envoi acquittement alarme)
		5	AR alarm-ind
207	Réservé	6 - 255	Réservé
		0 - 255	Réservé
208	CMINA	0	Etat non valide
		1	CMINA a signalé une erreur
		2 - 255	Réservé
209	CMPBE	0	Etat non valide
		1	CMPBE a signalé une erreur
		2 - 255	Réservé
210	CMDMC	0	Etat non valide
		1	CMDMC a signalé une erreur
		2 - 255	Réservé
211 - 220	Réservé	0 - 255	Réservé

Tableau 485 – Valeurs d'ErrorCode1 et ErrorCode2 pour ErrorDecode avec la valeur PNIO (partie 6 – raisons de l'arrêt prématuré)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
253	Utilisé par RTA pour les erreurs de protocole (RTA_ERR_CLS_PROTOCOL)	0	Réservé
		1	Erreur au sein de la coordination des numéros de séquence (RTA_ERR_CODE_SEQ) error
		2	Instance fermée (RTA_ERR_ABORT)
		3	AR out of memory (RTA_ERR_ABORT)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		4	AR add provider or consumer failed (AR échec d'ajout de fournisseur ou de consommateur) (RTA_ERR_ABORT)
		5	AR consumer DHT expired (RTA_ERR_ABORT)
		6	AR cmi timeout (RTA_ERR_ABORT)
		7	AR alarm-open failed (échec alarme ouverture) (RTA_ERR_ABORT)
		8	AR alarm-send.cnf(-) (RTA_ERR_ABORT)
		9	AR alarm-ack-send.cnf(-) (RTA_ERR_ABORT)
		10	AR alarm data too long (RTA_ERR_ABORT)
		11	AR alarm.ind(err) (RTA_ERR_ABORT)
		12	AR rpc-client call.cnf(-) (RTA_ERR_ABORT)
		13	AR abort.req (RTA_ERR_ABORT)
		14	AR re-run aborts existing (RTA_ERR_ABORT)
		15	AR release.ind received (RTA_ERR_ABORT)
		16	AR device deactivated (RTA_ERR_ABORT)
		17	AR removed (RTA_ERR_ABORT)
		18	AR protocol violation (RTA_ERR_ABORT)
		19	AR name resolution error (RTA_ERR_ABORT)
		20	AR RPC-Bind error (RTA_ERR_ABORT)
		21	AR RPC-Connect error (RTA_ERR_ABORT)
		22	AR RPC-Read error (RTA_ERR_ABORT)
		23	AR RPC-Write error (RTA_ERR_ABORT)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		24	AR RPC-Control error (RTA_ERR_ABORT)
		25	AR forbidden pull or plug after check.rsp and before in-data.ind (RTA_ERR_ABORT)
		26	AR AP removed (RTA_ERR_ABORT)
		27	AR link down (RTA_ERR_ABORT)
		28	AR could not register multicast-mac address (RTA_ERR_ABORT)
		29	Not synchronized (cannot start companion-ar) (RTA_ERR_ABORT)
		30	Wrong topology (cannot start companion-ar) (RTA_ERR_ABORT)
		31	DCP, station-name changed (RTA_ERR_ABORT)
		32	DCP, reset to factory or factory reset (RTA_ERR_ABORT)
		33	Can't start companion-AR because a 0xLIPP submodule in the first AR... (RTA_ERR_ABORT)
		34	No IRDATA record yet (RTA_ERR_ABORT)
		35	Pdev, owner is leaving (RTA_ERROR_ABORT)
		36	Pdev, no port offers required speed/duplexity (RTA_ERROR_ABORT)
		37	IP-Suite [of the IOC] changed by means of DCP_set(IPParameter) or local engineering (RTA_ERROR_ABORT)
		38	IOCARSR RDHT expired (RTA_ERROR_ABORT)
		39	IOCARSR Pdev, parametrization impossible (RTA_ERROR_ABORT)
		40	Remote application ready timeout expired (RTA_ERROR_ABORT)

Valeur ErrorCode1 (décimal)	Signification	Valeur ErrorCode2 (décimale)	Signification/Utilisation
		41	IOCARSR Redundant interface lost or access to the peripherals impossible (RTA_ERROR_ABORT)
		42 - 200	Réservé
		201 - 255	Spécifique à un fabricant
254	Réservé	0 - 255	Réservé
255	User specific	0 - 254	Spécifique à un utilisateur
		255	Recommended for "User abort" without further detail

Tableau 486 – Valeurs d'ErrorCode2 pour ErrorCode1 = RPC

Valeur (décimale)	Définition	Signification
0	Réservé	—
1	CLRPC_ERR_REJECTED	Le mappeur ou le serveur de point d'extrémité n'a pas rejeté l'appel. Pour plus de détails, voir le Tableau 255 et le Tableau 256.
2	CLRPC_ERR_FAULTED	Le serveur présentait un défaut lors de l'exécution de l'appel. Pour plus de détails, voir le Tableau 255 et le Tableau 256.
3	CLRPC_ERR_TIMEOUT	Le mappeur ou le serveur de point d'extrémité n'a pas répondu.
4	CLRPC_ERR_IN_ARGS	Diffusion ou peut-être "ndr_data" trop grand(e)
5	CLRPC_ERR_OUT_ARGS	Le serveur a renvoyé plus que "alloc_len"
6	CLRPC_ERR_DECODE	Le résultat d'une recherche d'un mappeur de point d'extrémité ne pourrait pas être décodé
7	CLRPC_ERR_PNIO_OUT_ARGS	Les Out-args ne sont pas "PN IO signature", trop courts ou incohérents
8	CLRPC_ERR_PNIO_APP_TIMEOUT	L'appel RPC a été terminé après la temporisation d'application de RPC
9 à 255	Réservé	—

5.2.6.4.4 Valeur ManufacturerSpecific d'ErrorDecode

La valeur ManufacturerSpecific de l'ErrorDecode indique que les paramètres ErrorCode1 et ErrorCode2 doivent être mis conformément au Tableau 487 et au Tableau 488.

Tableau 487 – Codage d'ErrorCode1 pour ErrorDecode avec la valeur ManufacturerSpecific

ErrorCode1 (hexadecimal)	Signification	Utilisation
0x00 – 0xFF	Spécifique à un fabricant	—

Tableau 488 – Codage d'ErrorCode2 pour ErrorDecode avec la valeur ManufacturerSpecific

ErrorCode2 (hexadécimale)	Signification	Utilisation
0x00 – 0xFF	Spécifique à un fabricant	—

5.2.7 Section de codage relative aux enregistrements I&M

5.2.7.1 Généralités

Toutes les données du type VisibleString doivent être justifiées à gauche. Si le texte est plus court que la longueur de chaîne définie, l'espace doit être rempli par des blancs.

5.2.7.2 Codage du champ IM_Serial_Number

Ce champ doit être codé en type de données VisibleString[16]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de blancs si sa longueur est inférieure à 16 octets.

Il identifie de façon unique une entité produite; il est donc une extension de VendorID, de DeviceID, de ModuleIdentNumber et de SubmoduleIdentNumber.

5.2.7.3 Codage du champ IM_Hardware_Revision

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 489.

Tableau 489 – IM_Hardware_Revision

Valeur (hexadécimale)	Signification
0x0000 – 0xFFFF	Révision matérielle

5.2.7.4 Codage du champ IM_SWRevision_Functional_Enhancement

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 490.

Tableau 490 – IM_SWRevision_Functional_Enhancement

Valeur (hexadécimale)	Signification
0x00 – 0xFF	Amélioration fonctionnelle

5.2.7.5 Codage du champ IM_SWRevision_Bug_Fix

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 491.

Tableau 491 – IM_SWRevision_Bug_Fix

Valeur (hexadécimale)	Signification
0x00 – 0xFF	Réparation de bogues

5.2.7.6 Codage du champ IM_SWRevision_Internal_Change

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 492.

Tableau 492 – IM_SWRevision_Internal_Change

Valeur (hexadécimale)	Signification
0x00 – 0xFF	Modification interne

5.2.7.7 Codage du champ IM_Revision_Counter

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 493.

Tableau 493 – IM_Revision_Counter

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Compteur de révision	Obligatoire
0x0001 – 0xFFFF	Compteur de révision	Facultatif

5.2.7.8 Codage du champ IM_Profile_ID

Ce champ, défini en tant que numéro administratif, doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 494.

Tableau 494 – IM_Profile_ID

Valeur (hexadécimale)	Signification
0x0000	Pas d'appareil de profil
0x0001 – 0xF6FF	Numéro administratif
0xF700 – 0xFFFE	Réservé aux profils
0xFFFF	Réservé

NOTE 1 Conformément à 5.2.3.1.

NOTE 2 A l'avenir, la présente norme peut étendre ce champ à Unsigned32.

5.2.7.9 Codage du champ IM_Profile_Specific_Type

Ce champ, défini sous la forme d'un numéro administratif, doit être codé en type de données Unsigned16 avec les valeurs conformes au Tableau 495 et au Tableau 496.

Tableau 495 – IM_Profile_Specific_Type conjointement avec IM_Profile_ID:= 0x0000

Valeur (hexadécimale)	Signification
0x0000	Non spécifié
0x0001	Appareil de commande normalisé, par exemple Contrôleur Logique Programmable
0x0002	Station basée sur PC
0x0003	Module E/S
0x0004	Module de Communication
0x0005	Module de l'Interface
0x0006	Composant de l'Infrastructure du Réseau Actif
autre	—

Tableau 496 – IM_Profile_Specific_Type conjointement avec IM_Profile_ID range 0x0001 – 0xF6FF

Valeur (hexadécimale)	Signification
0x0000	Non spécifié si le profil ne le définit pas autrement
0x0001 – 0xFFFF	L'utilisation doit être définie par les profils et sera traitée comme une sous-version du profil

5.2.7.10 Codage du champ IM_Version_Major

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 497.

Tableau 497 – IM_Version_Major

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Doit être mis dans cette version
0x02 – 0xFF	Réservé

5.2.7.11 Codage du champ IM_Version_Minor

Ce champ doit être codé en type de données Unsigned8 avec des valeurs conformes au Tableau 498.

Tableau 498 – IM_Version_Minor

Valeur (hexadécimale)	Signification
0x00	Réservé
0x01	Doit être mis dans cette version
0x02 – 0xFF	Réservé

5.2.7.12 Codage du champ IM_Supported

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0: IM_Supported.Profile_specific

Il convient que ce champ soit réglé à zéro.

NOTE Un profil d'application peut définir un comportement différent.

Bit 1: IM_Supported.I&M1

Ce champ doit être mis conformément au Tableau 499.

Tableau 499 – IM_Supported.I&M1

Valeur (hexadécimale)	Signification
0x00	Non pris en charge
0x01	L'enregistrement permet un accès en lecture et en écriture.

L'application peut refuser un accès en écriture.

Bit 2: IM_Supported.I&M2

Ce champ doit être mis conformément au Tableau 499.

Bit 3: IM_Supported.I&M3

Ce champ doit être mis conformément au Tableau 499.

Bit 4: IM_Supported.I&M4

Ce champ doit être mis conformément au Tableau 499.

Bit 5: IM_Supported.I&M5

Ce champ doit être mis conformément au Tableau 499.

Bit 6: IM_Supported.I&M6

Ce champ doit être mis conformément au Tableau 499.

Bit 7: IM_Supported.I&M7

Ce champ doit être mis conformément au Tableau 499.

Bit 8: IM_Supported.I&M8

Ce champ doit être mis conformément au Tableau 499.

Bit 9: IM_Supported.I&M9

Ce champ doit être mis conformément au Tableau 499.

Bit 10: IM_Supported.I&M10

Ce champ doit être mis conformément au Tableau 499.

Bit 11: IM_Supported.I&M11

Ce champ doit être mis conformément au Tableau 499.

Bit 12: IM_Supported.I&M12

Ce champ doit être mis conformément au Tableau 499.

Bit 13: IM_Supported.I&M13

Ce champ doit être mis conformément au Tableau 499.

Bit 14: IM_Supported.I&M14

Ce champ doit être mis conformément au Tableau 499.

Bit 15: IM_Supported.I&M15

Ce champ doit être mis conformément au Tableau 499.

5.2.7.13 Codage du champ IM_Tag_Function

Ce champ doit être codé en type de données VisibleString[32]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de blancs si sa longueur est inférieure à 32 octets.

NOTE VisibleString[32] rempli du caractère vide ' ' indique le rétablissement de la valeur d'usine.

5.2.7.14 Codage du champ IM_Tag_Function

Ce champ doit être codé en type de données VisibleString[22]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de blancs si sa longueur est inférieure à 22 octets.

NOTE VisibleString[22] rempli du caractère vide ' ' indique le rétablissement de la valeur d'usine.

5.2.7.15 Codage du champ IM_Date

Ce champ doit être codé en type de données VisibleString[16] avec les valeurs conformes au Tableau 500.

Le format de chaîne "YYYY-MM-DD'T'HH:MM'Z'" or "YYYY-MM-DD'T'HH:MM'±HH:MM'" est conforme à l'ISO 8601.

La lettre 'T' indique qu'une valeur d'heure suit la date et la lettre 'Z' indique que la valeur de l'heure est une heure UTC.

Pour des raisons historiques, le 'T' est remplacé par " " et les informations relatives au fuseau horaire 'Z' ou '±HH:MM' sont écartées.

Tableau 500 – IM_Date

Octet	Signification	Utilisation
0 - 3	YYYY	Année à quatre chiffres
4	"_"	Trait d'union (–) comme séparateur
5 - 6	MM	Mois à deux chiffres
7	"_"	Trait d'union (–) comme séparateur
8 - 9	DD	Jour à deux chiffres
10	" "	Caractère vide (' ') comme séparateur
11 - 12	HH	Heure à deux chiffres
13	":"	Caractère deux points (':') comme séparateur
14 - 15	MM	Minute à deux chiffres

NOTE VisibleString[16] rempli du caractère vide ' ' indique le rétablissement de la valeur d'usine.

5.2.7.16 Codage du champ IM_Descriptor

Ce champ doit être codé en type de données VisibleString[54]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de blancs si sa longueur est inférieure à 54 octets.

NOTE VisibleString[54] rempli du caractère vide ' ' indique le rétablissement de la valeur d'usine.

5.2.7.17 Codage du champ IM_Signature

Ce champ doit être codé en type de données OctetString[54]. Le champ doit être mis à une valeur spécifique au fabricant et être rempli de zéros si sa longueur est inférieure 54 octets

NOTE OctetString[54] nul ou la valeur par défaut de l'application indique le rétablissement des valeurs d'usine.

5.2.8 Section de codage relative à Alarm et Diagnosis PDUs

5.2.8.1 Codage du champ UserStructureIdentifier

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 501. Il identifie la structure du champ Data d'AlarmNotification et celle du champ Data des données d'alarme.

Tableau 501 – UserStructureIdentifier

Valeur (hexadécimale)	Signification	Utilisation
0x0000 – 0x7FFF	ManufacturerSpecific	ManufacturerSpecific Diagnosis doit être utilisé conjointement avec les AlarmTypes suivants: Tous les AlarmTypes rattachés à ASE de Diagnostic ManufacturerSpecific Diagnosis doit être utilisé conjointement avec les enregistrements suivants: Tous les enregistrements contenant Diagnosis Data Conjointement avec d'autres types d'alarme, l'utilisation est spécifique au fabricant.
0x8000	ChannelDiagnosis	ChannelDiagnosis doit être utilisé conjointement avec les AlarmTypes suivants: Tous les AlarmTypes rattachés à ASE de Diagnostic ChannelDiagnosis doit être utilisé conjointement avec les enregistrements suivants: Tous les enregistrements contenant Diagnosis Data
0x8001	Multiple	Réservé
0x8002	ExtChannelDiagnosis	ExtChannelDiagnosis doit être utilisé conjointement avec les AlarmTypes suivants: Tous les AlarmTypes rattachés à ASE de Diagnostic ExtChannelDiagnosis doit être utilisé conjointement avec les enregistrements suivants: Tous les enregistrements contenant Diagnosis Data
0x8003	QualifiedChannel-Diagnosis	QualifiedChannelDiagnosis doit être utilisé conjointement avec les AlarmTypes suivants: Tous les AlarmTypes rattachés à ASE de Diagnostic QualifiedChannelDiagnosis doit être utilisé conjointement avec les enregistrements suivants: Tous les enregistrements contenant Diagnosis Data
0x8004 – 0x80FF	Réservé	—

Valeur (hexadécimale)	Signification	Utilisation
0x8100	Maintenance	Maintenance doit être utilisé conjointement avec les AlarmTypes suivants: Tous les AlarmTypes rattachés à ASE de Diagnostic De plus, AlarmNotification doit uniquement acheminer ce bloc si la source de l'alarme contient au moins une entrée Maintenance Required, Maintenance Demanded ou Qualifier_x. Sinon, le bloc doit être ignoré.
0x8101 – 0x81FF	Réservé	—
0x8200	Upload&Retrieval	Upload&Retrieval doit être utilisé conjointement avec la notification de chargement et d'extraction.
0x8201	iParameter	iParameter doit être utilisé conjointement avec la notification de chargement et d'extraction.
0x8202 – 0x8FFF	Réservé	—
0x8300	Sequence of events	Sequence of events doit être utilisé conjointement avec les AlarmTypes suivants: Process alarm
0x8301 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	—
0xA000 – 0xFFFF	Réservé	—

5.2.8.2 Codage du champ ChannelErrorType

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 502.

Tableau 502 – ChannelErrorType

Valeur (hexadécimale)	Signification	Texte attribué
0x0000	Réservé	Erreur inconnue
0x0001	Court-circuit	Court-circuit
0x0002	Sous-tension	Sous-tension
0x0003	Surtension	Surtension
0x0004	Surcharge	Surcharge
0x0005	Température excessive	Température excessive
0x0006	Saut de ligne	Saut de ligne
0x0007	Valeur limite supérieure dépassée	Valeur limite supérieure dépassée
0x0008	Valeur limite inférieure dépassée	Valeur limite inférieure dépassée
0x0009	Erreur	Erreur
0x000A	Simulation active	Simulation active
0x000B	Réservé	Erreur inconnue
0x000C	Réservé	Erreur inconnue
0x000D	Réservé	Erreur inconnue
0x000E	Réservé	Erreur inconnue
0x000F	Par défaut pour le paramètre manquant ^b	Le canal a besoin de paramètres (supplémentaires). Aucun ou peu de paramètres sont écrits

Valeur (hexadécimale)	Signification	Texte attribué
0x0010	Par défaut pour le défaut de paramétrage ^b	Défaut de paramétrage. Des paramètres incorrects ou trop de paramètres sont écrits
0x0011	Par défaut pour le défaut d'alimentation électrique ^b	Défaut d'alimentation électrique
0x0012	Par défaut en cas de fonde/ouverture de fusible ^b	Fusible fondu/ouvert
0x0013	Par défaut pour le défaut de communication ^b	Défaut de communication. Numéro de séquence incorrect/séquence incorrecte
0x0014	Par défaut pour le défaut de mise à la terre ^b	Défaut de mise à la terre
0x0015	Par défaut en cas de perte de point de référence ^b	Perte de point de référence
0x0016	Par défaut en cas de perte d'événement de processus/erreur d'échantillonnage ^b	Perte d'événement de processus/erreur d'échantillonnage
0x0017	Par défaut pour l'avertissement de seuil ^b	Avertissement de seuil
0x0018	Par défaut pour la sortie désactivée ^b	Sortie désactivée
0x0019	Par défaut pour l'événement de sécurité ^b	Événement de sécurité
0x001A	Par défaut pour le défaut externe ^b	Défaut externe
0x001B	Spécifique à un fabricant	Spécifique à un fabricant
0x001C	Spécifique à un fabricant	Spécifique à un fabricant
0x001D	Spécifique à un fabricant	Spécifique à un fabricant
0x001E	Spécifique à un fabricant	Spécifique à un fabricant
0x001F	Par défaut pour le défaut temporaire ^b	Défaut temporaire
0x0020 – 0x00FF	Réservé aux profils communs ^a	Numéro administratif central pour distinguer sans ambiguïté les profils communs
0x0100 – 0x7FFF	Spécifique à un fabricant	Spécifique à un fabricant
0x8000	Transmission de données impossible	Data Transmission Impossible
0x8001	Discordance distante	Remote Mismatch
0x8002	Discordance de redondance de support	Media Redundancy Mismatch
0x8003	Discordance de synchronisation	Sync Mismatch
0x8004	Discordance IsochronousMode	IsochronousMode Mismatch
0x8005	Discordance de CR multidiffusion	Multicast CR Mismatch
0x8006	Réservé	Réservé
0x8007	Discordance de fibre optique	Informations de liaisons par fibre optique.
0x8008	Discordance de fonction de composant réseau	Un problème de fonctionnalité de réseau s'est produit
0x8009	Discordance temporelle	Base de temps inexistante ou problèmes de précision
0x800A	Discordance des fonctions de combinaison de trames dynamique	Problèmes DFP surviennent
0x800B	Redondance de support avec discordance de duplication prévue	Problèmes MRPD surviennent
0x800C	Discordance de redondance du Système	System Redundancy Mismatch
0x800D	Discordance de plusieurs interfaces	Informations relatives à des problèmes d'interfaces multiples

Valeur (hexadécimale)	Signification	Texte attribué
0x800E	Indication de diagnostic imbriquée	Communication entre contrôleurs Utilisé pour propager le diagnostic dans des structures hiérarchiques
0x800F	Réservé	Erreur inconnue
0x8010 – 0x8FFF	Réservé	Erreur inconnue
0x9000 – 0x9FFF	Reservé aux profils ^a	Spécifique au profil
0xA000 – 0xFFFF	Réservé	Erreur inconnue
^a Numéro administratif central pour distinguer sans ambiguïté les profils		
^b Pour des appareils hérités "spécifiques au fabricant"		

5.2.8.3 Codage du champ ChannelNumber

Ce champ doit être codé comme type de données Unsigned16 avec les valeurs conformes au Tableau 503.

Tableau 503 – ChannelNumber

Valeur (hexadécimale)	Signification
0x0000 – 0x7FFF	Spécifique à un fabricant
0x8000	Sous-module Il faut utiliser un seul codage pour ChannelProperties.Direction.
0x8001 – 0xFFFF	Réservé

NOTE ChannelNumber 0x8000 fait référence au sous-module lui-même.

Un canal distinct est adressé par ChannelNumber, outre ChannelProperties.Direction.

En fonction du champ ChannelProperties.Accumulative, la signification suivante doit être appliquée:

- ChannelProperties.Accumulative=1, le champ ChannelNumber doit contenir le numéro du plus petit canal du groupe concerné
- ChannelProperties.Accumulative=0, le champ ChannelNumber doit contenir le numéro du canal concerné

5.2.8.4 Codage du champ ChannelProperties

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0 – 7: ChannelProperties.Type

Ce champ doit être mis conformément au Tableau 504.

Tableau 504 – ChannelProperties.Type

Valeur (hexadécimale)	Signification	Utilisation
0x00	—	Doit être utilisé si le champ ChannelNumber contient la valeur 0x8000 (sous-module) De plus, il doit être utilisé si aucun des types définis ci-dessous n'est approprié.

Valeur (hexadécimale)	Signification	Utilisation
0x01	1 Bit	La longueur des données de ce canal est de 1 Bit.
0x02	2 Bit	La longueur des données de ce canal est de 2 Bit.
0x03	4 Bit	La longueur des données de ce canal est de 4 Bit.
0x04	8 Bit	La longueur des données de ce canal est de 8 Bit.
0x05	16 Bit	La longueur des données de ce canal est de 16 Bit.
0x06	32 Bit	La longueur des données de ce canal est de 32 Bit.
0x07	64 Bit	La longueur des données de ce canal est de 64 Bit.
0x08 – 0xFF	Réservé	—

Bit 8: ChannelProperties.Accumulative

Ce champ doit être mis à un s'il s'agit d'un diagnostic systématique de plusieurs canaux. Autrement, il doit être mis à zéro.

Bit 9 – 10: ChannelProperties.Maintenance

Ce champ doit être mis conformément au Tableau 505, au Tableau 506 et au Tableau 507.

Les dépendances avec les éléments du champ ChannelProperties sont spécifiées au Tableau 506 et avec celles avec Alarmnotification et RecordDataRead(DiagnosisData) au Tableau 507.

Tableau 505 – ChannelProperties.Maintenance

MaintenanceRequired Bit 9	MaintenanceDemanded Bit 10	Utilisation
0x00	0x00	Diagnostic
0x01	0x00	Maintenance requise
0x00	0x01	Maintenance exigée
0x01	0x01	QualifiedChannelQualifier ^a
^a Cela indique que la sévérité est codée dans le champ QualifiedChannelQualifier (voir Figure 83). Toutes les valeurs de ChannelProperties.Maintenance pourraient être utilisées en combinaison avec Qualified Diagnosis.		

Tableau 506 – Combinaisons valides au sein de ChannelProperties

Maintenance	Spécificateur	Signification	Valide avec
Diagnostic	0x00	Tous les ^a Diagnosis, MaintenanceRequired, MaintenanceDemanded et QualifiedDiagnosis consécutifs disparaissent	ChannelDiagnosis
	0x01	Diagnosis apparaît	ChannelDiagnosis, ManufacturerSpecificDiagnosis, ExtChannelDiagnosis
	0x02	Diagnosis disparaît	
	0x03	Un "diagnosis" disparaît, mais l'autre reste	
Maintenance requise	0x00	Réservé	ChannelDiagnosis, ManufacturerSpecificDiagnosis, ExtChannelDiagnosis
	0x01	MaintenanceRequired apparaît	
	0x02	MaintenanceRequired disparaît	
	0x03	MaintenanceRequired disparaît mais l'autre reste	
Maintenance exigée	0x00	Réservé	ChannelDiagnosis, ManufacturerSpecificDiagnosis, ExtChannelDiagnosis
	0x01	MaintenanceDemanded apparaît	
	0x02	MaintenanceDemanded disparaît	
	0x03	MaintenanceDemanded disparaît mais l'autre reste	
Qualified-Channel-Qualifier	0x00	Réservé	QualifiedChannelDiagnosis
	0x01	QualifiedDiagnosisQualifier apparaît	
	0x02	QualifiedDiagnosisQualifier disparaît	
	0x03	QualifiedDiagnosisQualifier disparaît mais l'autre reste	

^a "Consécutif" signifie que tous les ExtChannelErrorTypes du ChannelErrorType livré et le ChannelErrorType lui-même disparaissent. Doit être utilisé dans AlarmNotification uniquement.

Tableau 507 – Combinaisons valides pour Alarmnotification et Record-DataRead(DiagnosisData)

Maintenance	Spécificateur	Signification	Valide avec
Diagnostic	0x00	Tous les ^a Diagnosis, MaintenanceRequired, MaintenanceDemanded et QualifiedDiagnosis consécutifs disparaissent	Alarmnotification
	0x01	Diagnosis apparaît	Alarmnotification et Record-DataRead(DiagnosisData)
	0x02	Diagnosis disparaît	Alarmnotification RecordDataRead(DiagnosisData) ne doit être utilisé que conjointement avec ManufacturerSpecificDiagnosis
	0x03	Un "diagnosis" disparaît, mais l'autre reste	Alarmnotification
Maintenance	0x00	Réservé	—

Maintenance	Spécificateur	Signification	Valide avec
requis	0x01	MaintenanceRequired apparaît	Alarmnotification et Record-DataRead(DiagnosisData)
	0x02	MaintenanceRequired disparaît	Alarmnotification
	0x03	MaintenanceRequired disparaît mais l'autre reste	
Maintenance exigée	0x00	Réservé	—
	0x01	MaintenanceDemanded apparaît	Alarmnotification et Record-DataRead(DiagnosisData)
	0x02	MaintenanceDemanded disparaît	Alarmnotification
	0x03	MaintenanceDemanded disparaît mais l'autre reste	
Qualified diagnosis	0x00	Réservé	—
	0x01	QualifiedDiagnosis apparaît	Alarmnotification et Record-DataRead(DiagnosisData)
	0x02	QualifiedDiagnosis disparaît	Alarmnotification
	0x03	QualifiedDiagnosis disparaît mais l'autre reste	
a "Consécutif" signifie que tous les ExtChannelErrorTypes du ChannelErrorType livré et le ChannelErrorType lui-même disparaissent.			

Bit 11 – 12: ChannelProperties.Specifier

Ce champ doit être codé avec les valeurs conformes au Tableau 508. Le champ ChannelProperties.Specifier est lié au canal adressé et à la sévérité (diagnostic, maintenance ou qualified). Les dépendances avec les éléments du champ ChannelProperties sont spécifiées au Tableau 506.

Tableau 508 – ChannelProperties.Specifier

Valeur (hexadécimale)	Signification	Utilisation
0x00	Tous les éléments consécutifs disparaissent	Voir Tableau 506 L'ASE de Diagnostic ne contient plus d'entrées (de n'importe quelle sévérité) pour cette voie
0x01	Apparaît	Un événement apparaît et/ou existe encore L'ASE de Diagnostic contient cela et également d'autres entrées possibles pour cette voie.
0x02	Disparaît	Un événement disparaît et/ou n'existe plus L'ASE de Diagnostic ne contient plus d'entrées de la sévérité de cette voie
0x03	Disparaît mais l'autre reste	Un événement disparaît L'ASE de Diagnostic contient encore d'autres entrées de la même sévérité pour cette voie

NOTE Les données acheminées d'AlarmNotification dépendent de l'heure de création et peuvent différer du contenu réel de l'ASE de Diagnostic. AlarmNotification est utilisé pour acheminer les modifications et l'ASE de Diagnostic, pour contenir les statuts réels.

Bit 13 – 15: ChannelProperties.Direction

Ce champ doit être mis conformément au Tableau 509.

Tableau 509 – ChannelProperties.Direction

Valeur (hexadécimale)	Signification
0x00	Spécifique à un fabricant
0x01	Entrée
0x02	Sortie
0x03	Entrée/Sortie
0x04 – 0x07	Réservé

5.2.8.5 Codage du champ ExtChannelErrorType

Ce champ doit être codé en type de données Unsigned16. Sa valeur dépend du champ ChannelErrorType. Les valeurs doivent être mises conformément au Tableau 510.

Tableau 510 – ExtChannelErrorType

Valeur (hexadécimale)	Signification
0x0000 – 0xFFFF	La définition dépend de ChannelErrorType (voir les tableaux ci-dessous).

Conjointement avec ChannelErrorTypes, ExtChannelErrorType doit être mis conformément au Tableau 512, Tableau 513, Tableau 514, Tableau 515, Tableau 516, Tableau 517, Tableau 518, Tableau 519, Tableau 520, Tableau 521, Tableau 522, Tableau 523, Tableau 524, Tableau 525, et au Tableau 526.

De plus, une plage de valeurs de ExtChannelErrorType est associée au type de ChannelErrorType conformément au Tableau 511.

Tableau 511 – Combinaisons autorisées de ChannelErrorType et d'ExtChannelAddValue

ChannelErrorType	ExtChannelErrorType	ExtChannelAddValue	Utilisation
Normative	Normative	Normative	Autorisé
Normative	Spécifique au vendeur	Spécifique au vendeur	Autorisé
Normative	Spécifique au profil	Spécifique au profil	Autorisé
Spécifique au profil	Spécifique au profil	Spécifique au profil	Autorisé
Spécifique au profil	Spécifique au vendeur	Spécifique au vendeur	Autorisé
Spécifique au profil	Normative	Normative	Interdit
Spécifique au vendeur	Spécifique au vendeur	Spécifique au vendeur	Autorisé
Spécifique au vendeur	Normative	Normative	Interdit
Spécifique au vendeur	Spécifique au profil	Spécifique au profil	Interdit

Tableau 512 – ExtChannelErrorType pour ChannelErrorType 0 – 0xFF

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Informations cumulées	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 513 – ExtChannelErrorType pour ChannelErrorType 0x0100 – 0x7FFF

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Informations cumulées	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 514 – ExtChannelErrorType pour ChannelErrorType "Transmission de données impossible"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Discordance d'état de liaison – Liaison interrompue	Alarme/diagnostic
0x8001	Discordance MAUType	Alarme/diagnostic
0x8002	Discordance de retard de ligne	Alarme/diagnostic
0x8003 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 515 – ExtChannelErrorType pour ChannelErrorType "Discordance distante"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Nom de l'homologue de la discordance de station	Alarme/diagnostic
0x8001	Nom de l'homologue de la discordance de port	Alarme/diagnostic
0x8002	Discordance RT_CLASS_3 homologue ^a	Alarme/diagnostic
0x8003	Discordance MAUType homologue	Alarme/diagnostic
0x8004	Discordance de domaine MRP homologue	Alarme/diagnostic

Valeur (hexadécimale)	Signification	Utilisation
0x8005	Aucun homologue détecté	Alarme/diagnostic
0x8006	Réservé	—
0x8007	Discordance de retard de ligne homologue	Alarme/diagnostic
0x8008	Discordance PTCP homologue ^b	Alarme/diagnostic
0x8009	Discordance de Longueur de Préambule Homologue	Alarme/diagnostic
0x800A	Discordance de Fragmentation Homologue	Alarme/diagnostic
0x800B – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

^a Cet événement se produit si le RTClass3_PortStatus.State distant n'est pas RTCLASS3_RUN. Cela pourrait également se produire si le IRDATAUUID distant est différent du IRDATAUUID local.

^b Cet événement se produit si le PTCP_SubdomainUUID distant est égal au PTCP_SubdomainUUID local et si le PTCP_MasterSourceAddress distant est différent du PTCP_MasterSourceAddress local.

Tableau 516 – ExtChannelErrorType pour ChannelErrorType "Discordance de redondance de support"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Echec du rôle du gestionnaire	Alarme/diagnostic
0x8001	Anneau MRP ouvert	Alarme/diagnostic
0x8002	Réservé	—
0x8003	Plusieurs gestionnaires	Alarme/diagnostic
0x8004 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

La source d'une "Discordance de Redondance de Support" est toujours un port. Si deux ports sont affectés, celui ayant le plus petit numéro doit être utilisé.

Tableau 517 – ExtChannelErrorType pour ChannelErrorType "Discordance de Synchronisation" et pour ChannelErrorType "Discordance Temporelle"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Aucun message de synchronisation reçu	Alarme/diagnostic
0x8001 – 0x8002	Réservé	—
0x8003	Gigue hors limite	Alarme/diagnostic
0x8004 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 518 – ExtChannelErrorType pour ChannelErrorType "Discordance de mode isochrone"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Echec du temps de sortie – Mise à jour de la sortie manquante ou irrecevable	Alarme/diagnostic
0x8001	Echec du temps d'entrée	Alarme/diagnostic
0x8002	Echec de signe de vie du maître – Erreur de mise à jour MLS détectée	Alarme/diagnostic
0x8003 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 519 – ExtChannelErrorType pour ChannelErrorType "Discordance CR multidiffusion"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Expiration du délai CR du consommateur multidiffusion	Alarme/diagnostic
0x8001	Echec de la résolution d'adresse	Alarme/diagnostic
0x8002 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 520 – ExtChannelErrorType pour ChannelErrorType "Discordance de fibre optique"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Bilan de puissance	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 521 – ExtChannelErrorType pour ChannelErrorType "Discordance de fonction de composant réseau"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Trame supprimée – Aucune ressource	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 522 – ExtChannelErrorType pour ChannelErrorType "Discordance de Fonction de Combinaison de Trame Dynamique"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000 – 0x80FF	Réservé	—
0x8100	Erreur tardive de trame pour FrameID "0x0100"	Alarme/diagnostic
0x8101 + 0x8FFE	Voir Formule (55)	Alarme/diagnostic
0x8FFF	Erreur tardive de trame pour FrameID "0x0FFF"	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

La source d'une "Discordance de Fonction de Combinaison de Trame Dynamique" est le sous-module d'interface utilisant la fonction de combinaison DFP.

L'ExtChannelErrorType utilisé doit être calculé à l'aide de la Formule (55).

$$\text{ExtChannelErrorType} = 0x8000 + \text{FrameID} \quad (55)$$

où

ExtChannelErrorType	est le ExtChannelErrorType utilisé pour le signal d'un événement
0x8000	est la valeur de base de ChannelNumber
FrameID	est le FrameID de la trame émettant l'événement

Tableau 523 – ExtChannelErrorType pour ChannelErrorType "Redondance de Support avec discordance de duplication prévue"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000 – 0x86FF	Réservé	—
0x8700	Duplication MRPD nulle pour FrameID "0x0700"	Alarme/diagnostic
0x8701 + 0x8FFE	Voir Formule (56)	Alarme/diagnostic

Valeur (hexadécimale)	Signification	Utilisation
0x8FFF	Duplication MRPD nulle pour FrameID "0x0FFF"	Alarme/diagnostic
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

La source d'une "Redondance de Support avec discordance de duplication prévue" est le sous-module d'interface utilisant la fonction MRPD.

NOTE La recherche du plus petit SlotNumber puis du plus petit SubslotNumber de cet emplacement identifie la combinaison SlotNumber/SubslotNumber plus petite.

L'ExtChannelErrorType utilisé doit être calculé à l'aide de la Formule (56).

$$\text{ExtChannelErrorType} = 0x8000 + \text{FrameID} \quad (56)$$

où

ExtChannelErrorType est le ExtChannelErrorType utilisé pour le signal d'un événement
 0x8000 est la valeur de base de ChannelNumber
 FrameID est le FrameID de la trame émettant l'événement

Tableau 524 – ExtChannelErrorType pour ChannelErrorType "Discordance de Redondance du Système"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Événement de redondance du système	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

La source d'une "Discordance de redondance du système" est le sous-module dont la combinaison API/SlotNumber/SubslotNumber est la plus faible, faisant partie de l'AR.

Tableau 525 – ExtChannelErrorType pour ChannelErrorType "Discordances interfaces multiples"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Discordance StandardGateway	Alarme/diagnostic
0x8001	NameOfStation de l'interface n'est pas sans équivoque	Alarme/diagnostic
0x8002	Plage IPAddress de l'interface n'est pas sans équivoque	Alarme/diagnostic
0x8003	Mode MultipleInterfaceMode.NameOfDevice conflictuel	Alarme/diagnostic

Valeur (hexadécimale)	Signification	Utilisation
0x8004 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

Tableau 526 – ExtChannelErrorType pour ChannelErrorType "Indication de diagnostic imbriquée"

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé	—
0x0001 – 0x7FFF	Spécifique à un fabricant	Alarme/diagnostic
0x8000	Informations sur l'entité subordonnée	Alarme/diagnostic
0x8001 – 0x8FFF	Réservé	—
0x9000 – 0x9FFF	Réservé aux profils	Alarme/diagnostic
0xA000 – 0xFFFF	Réservé	—

5.2.8.6 Codage du champ ExtChannelAddValue

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 527.

Tableau 527 – Valeurs pour ExtChannelAddValue

Valeur (hexadécimale)	Signification
0x00000000	Pas d'informations supplémentaires
0x00000001 – 0xFFFFFFFF	La définition dépend de ChannelError et de ExtChannelErrorType

5.2.8.6.1 Codage du champ ExtChannelAddValue pour ChannelErrorType = 0 – 0x7FFF

Ce champ doit être codé avec les valeurs conformes au Tableau 528 si les champs ExtChannelErrorType et ChannelErrorType contiennent respectivement les valeurs 0x8000 et 0 – 0x7FFF.

Tableau 528 – Valeurs pour Informations Cumulées

Position de bit	Valeur (hexadécimale)	Signification
Bit 0	0x00	ChannelNumber n'est pas concerné
	0x01	ChannelNumber est concerné
Bit 1	0x00	ChannelNumber + 1 n'est pas concerné
	0x01	ChannelNumber + 1 est concerné
Bit 2	0x00	ChannelNumber + 2 n'est pas concerné
	0x01	ChannelNumber + 2 est concerné
...	...	...
Bit 30	0x00	ChannelNumber + 30 n'est pas concerné
	0x01	ChannelNumber + 30 est concerné

Position de bit	Valeur (hexadécimale)	Signification
Bit 31	0x00	ChannelNumber + 31 n'est pas concerné
	0x01	ChannelNumber + 31 est concerné

5.2.8.6.2 Codage du champ ExtChannelAddValue pour ChannelErrorType "Discordance de fibre optique"

Ce champ doit être codé avec les valeurs conformes au Tableau 529 si les champs ExtChannelErrorType et ChannelErrorType contiennent respectivement les valeurs 0x8000 et 0x8007.

Tableau 529 – Valeurs de "Discordance de fibre optique" – "Bilan de puissance"

Valeur (hexadécimale)	Signification
0x00 – 0x3E7	PowerBudget par incréments de 0,1 dB [0..99,9 dB]
0x03E8 – 0xFFFFFFFF	Réservé
0xFFFFFFFF	Inconnu

5.2.8.6.3 Codage du champ ExtChannelAddValue pour ChannelErrorType "Discordance de fonction de composant réseau"

Ce champ doit être codé avec les valeurs conformes au Tableau 530 si les champs ExtChannelErrorType et ChannelErrorType contiennent respectivement les valeurs 0x8000 et 0x8008.

Tableau 530 – Valeurs de "Discordance de fonction de composant réseau" – "Trame supprimée"

Valeur (hexadécimale)	Signification
0x00000000 – 0x00FFFFFF	Nombre de trames supprimées en cas d'absence de ressource
0x01000000 – 0xFFFFFFFF	Réservé

5.2.8.6.4 Codage du champ ExtChannelAddValue pour ChannelErrorType "Discordance distante"

Ce champ doit être codé avec les valeurs conformes au Tableau 531 si les champs ExtChannelErrorType et ChannelErrorType contiennent respectivement les valeurs 0x8007 et 0x8001.

Tableau 531 – Valeurs de "Discordance distante" – "Discordance Cable Delay homologue"

Valeur (hexadécimale)	Signification	Utilisation
0x00 – 0x32	Erreur de mesure	Pas de signal
0x33 – 0x3B9ACA00	Différence CableDelay entre les homologues	Ecart de signal
0x3B9ACA01 – 0xFFFFFFFF	Réservé	—

5.2.8.6.5 Codage du champ ExtChannelAddValue pour ChannelErrorType "Discordances interfaces multiples"

Ce champ doit être codé avec les valeurs conformes au Tableau 532 si les champs ExtChannelErrorType et ChannelErrorType contiennent respectivement les valeurs 0x8003 et 0x800D.

Tableau 532 – Valeurs pour "Discordance d'interfaces multiples" – "mode MultipleInterfaceMode.NameOfDevice conflictuel"

Valeur (hexadécimale)	Signification	Utilisation
0x00	Réservé	—
0x01	MultipleInterfaceMode.NameOfDevice de cette interface est réglé à zéro mais une autre interface est réglée à un.	Incohérence
0x02	MultipleInterfaceMode.NameOfDevice de cette interface est réglé à un mais une autre interface est réglée à zéro.	Incohérence
autre	Réservé	—

5.2.8.7 Codage du champ QualifiedChannelQualifier

Ce champ doit être codé en type de données Unsigned32. Les valeurs doivent être mises conformément au Tableau 533. Un seul bit ne doit être mis dans un QualifiedChannelDiagnosis.

Tableau 533 – Valeurs pour QualifiedChannelQualifier

Position de bit	Valeur (hexadécimale)	Signification	Utilisation
Bit 0	Réservé	—	—
Bit 1	Réservé	—	—
Bit 2	0x00	Qualifier_2 non mis	Spécifique au profil
	0x01	Qualifier_2 est mis	
Bit 3	0x00	Qualifier_3 non mis	
	0x01	Qualifier_3 est mis	
...	...	...	
Bit 30	0x00	Qualifier_30 non mis	
	0x01	Qualifier_30 est mis	
Bit 31	0x00	Qualifier_31 non mis	
	0x01	Qualifier_31 est mis	

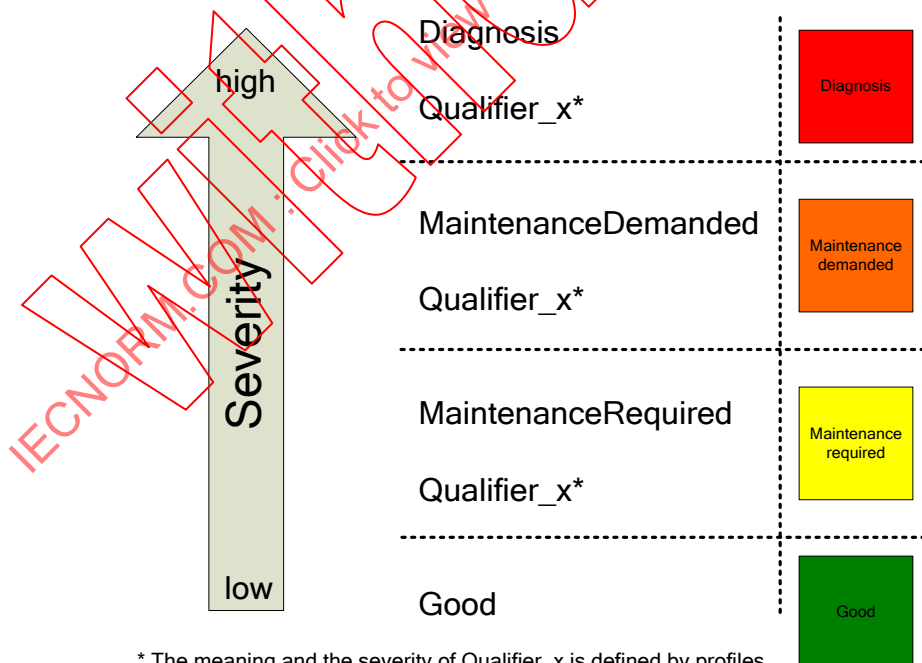
5.2.8.8 Codage du champ MaintenanceStatus

Ce champ doit être codé en type de données Unsigned32. Au moins l'un des bits ci-dessous doit être mis à un afin d'acheminer ce bloc d'alarme conformément au Tableau 534. Autrement, ce bloc doit être ignoré.

Tableau 534 – Valeurs pour MaintenanceStatus

Position de bit	Nom du bit	Valeur (hexadécimale)	Signification
Bit 0	MaintenanceRequired	0x00	Aucune information requise de maintenance disponible
		0x01	Informations requises de maintenance disponibles
Bit 1	MaintenanceDemanded	0x00	Aucune information demandée de maintenance disponible
		0x01	Informations demandées de maintenance disponibles
Bit 2	Qualifier_2	0x00	Aucune information disponible
		0x01	Informations disponibles
Bit 3	Qualifier_3	0x00	Aucune information disponible
		0x01	Informations disponibles
...	—	—	...
Bit 30	Qualifier_30	0x00	Aucune information disponible
		0x01	Informations disponibles
Bit 31	Qualifier_31	0x00	Aucune information disponible
		0x01	Informations disponibles

La classification de diagnostic, maintenance et de qualifié selon leur sévérité est présentée à la Figure 83.

**Légende**

Anglais	Français
high	haute
low	basse

Anglais	Français
Severity	Sévérité
Diagnosis	Diagnostic
Good	Bon
* The meaning and the severity of Qualifier_x is defined by profiles	* La signification et la sévérité de Qualifier_x sont définies par les profils
Maintenance demanded	Maintenance exigée
Maintenance required	Maintenance requise

Figure 83 – Classification selon la sévérité de diagnosis, maintenance et qualified

La signification et la sévérité de Qualifier_x doivent être définies par un profil.

5.2.9 Section de codage relative au chargement et à l'extraction

5.2.9.1 Codage du champ URRecordIndex

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 535.

Tableau 535 – URRecordIndex

Valeur (hexadécimale)	Signification
0x00000000 – 0x0000FFFF	Index de l'enregistrement utilisé
0x00010000 – 0xFFFFFFFF	Réserve

5.2.9.2 Codage du champ URRecordLength

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes au Tableau 536.

Tableau 536 – URRecordLength

Valeur (hexadécimale)	Signification
0x00000000	Enregistrements stockés en écriture dans l'appareil E/S.
0x00000001 – 0xFFFFFFFF	Enregistrements stockés en écriture dans l'appareil E/S, si la longueur de chaque enregistrement sélectionné est inférieure ou égale à URRecordLength.

5.2.10 Section de codage relative à iParameter

5.2.10.1 Codage du champ iPar_Req_Header

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes à la CEI 61784-3-3.

5.2.10.2 Codage du champ Max_Segm_Size

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes à la CEI 61784-3-3.

5.2.10.3 Codage du champ Transfer_Index

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes à la CEI 61784-3-3.

5.2.10.4 Codage du champ Total_iPar_Size

Ce champ doit être codé en type de données Unsigned32 avec des valeurs conformes à la CEI 61784-3-3.

5.2.11 Section de codage relative aux Données de l'Interface de l'Appareil Physique

5.2.11.1 Codage du champ MultipleInterfaceMode

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0: MultipleInterfaceMode.NameOfDevice

Ce champ doit être mis conformément au Tableau 537.

Tableau 537 – MultipleInterfaceMode.NameOfDevice

Valeur (hexadécimale)	Signification
0x00	Par défaut Le champ LLDPChassis contient le NameOfStation et le champ PortID de LLDP contient le nom du port.
0x01	Le champ LLDPChassis contient NameOfDevice, il est défini par des moyens locaux. Le champ PortID de LLDP contient le nom du port et NameOfStation.

Bit 1 – 15: MultipleInterfaceMode.reserved_1

Ce champ doit être mis à zéro.

Bit 16 – 31: MultipleInterfaceMode.reserved_2

Ce champ doit être mis conformément à 3.4.2.2.

5.2.12 Section de codage relative aux données de port de l'appareil physique

5.2.12.1 Codage du champ OwnPortName

Ce champ doit être codé en type de données OctetString conformément à 4.3.1.4.16.

5.2.12.2 Codage du champ LengthOwnPortName

Ce champ doit être codé en type de données Unsigned8 conformément à 4.3.1.4.16.

5.2.12.3 Codage du champ NumberOfPeers

Ce champ doit être codé en type de données Unsigned8.

5.2.12.4 Codage du champ LengthPeerPortName

Ce champ doit être codé en type de données Unsigned8.

5.2.12.5 Codage du champ PeerPortName

5.2.12.5.1 Utilisation conjointement avec PDPortDataCheck.CheckPeers

Ce champ doit être codé conformément à 4.3.1.4.16.

5.2.12.5.2 Utilisation conjointement avec PDPorDataReal

Ce champ doit être codé conformément à 4.3.1.4.16. Il doit contenir la partie NameOfPort du LLDP_PortID reçu, s'il est codé d'après 4.3.1.4.16. Sinon, LengthPeerPortName doit être zéro.

5.2.12.6 Codage du champ LengthPeerStationName

Ce champ doit être codé en type de données Unsigned8.

5.2.12.7 Codage du champ PeerStationName

5.2.12.7.1 Généralités

Ce champ doit être codé en type de données OctetString[] avec de 1 à 255 octets.

5.2.12.7.2 Utilisation conjointement avec PDPorDataCheck.CheckPeers

Ce champ doit être codé conformément à 4.3.1.4.15.

5.2.12.7.3 Utilisation conjointement avec PDPorDataReal

Ce champ doit être codé conformément à 4.3.1.4.15. Il doit contenir la partie NameOfStation du LLDP_PortID ou du LLDP_ChassisID reçu, s'il est codé d'après 4.3.1.4.15. Sinon, LengthPeerStationName doit être zéro.

5.2.12.8 Codage du champ LengthOwnStationName

Ce champ doit être codé en type de données Unsigned8 conformément à 4.3.1.4.15.

5.2.12.9 Codage du champ OwnStationName

Ce champ doit être codé en type de données OctetString[] conformément au 4.3.1.4.15.

5.2.12.10 Codage du champ LineDelay

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0 – 30: LineDelay.Value

Ce champ doit être mis conformément au Tableau 538, au Tableau 539, à la Figure 27 et à la Formule (26).

Tableau 538 – LineDelay.Value avec LineDelay.FormatIndicator == 0

Valeur (hexadécimale)	Signification
0x00000000	Retard de ligne et délai de câble inconnus
0x00000001 – 0x7FFFFFFF	Retard de ligne en nanosecondes

Tableau 539 – LineDelay.Value avec LineDelay.FormatIndicator == 1

Valeur (hexadécimale)	Signification
0x00000000	Réservé
0x00000001 – 0x7FFFFFFF	Délai de câble en nanosecondes

Bit 31: LineDelay.FormatIndicator

Ce champ doit être mis conformément au Tableau 540.

Tableau 540 – LineDelay.FormatIndicator

Valeur (hexadécimale)	Signification	Utilisation
0x00	LineDelay.Value est codé en retard de ligne	Par défaut, si le retard de ligne ou le délai de câble n'est pas connu Doit être utilisé conjointement avec PDPortDataCheck
0x01	LineDelay.Value est codé en délai de câble	Par défaut, si le délai de câble est connu Doit être utilisé conjointement avec PDPortDataReal

5.2.12.11 Codage du champ PeerMACAddress

Ce champ doit être codé en type de données OctetString[6]. La valeur du champ PeerMACAddress doit être conforme aux adresses LAN MAC universelles en 48 bits de l'IEEE 802.

NOTE L'octet 1 contient le bit d'adresse individuelle/de groupe (bit de poids faible).

5.2.13 Section de codage relative aux données IR de l'appareil physique**5.2.13.1 Codage du champ RxPort**

Ce champ doit être codé en type de données Unsigned8, avec des valeurs conformes au Tableau 541.

Tableau 541 – RxPort

Valeur (hexadécimale)	Signification
0x00	Interface locale
0x01	Port 1
0x02	Port 2
...	...
0xFF	Port 255

5.2.13.2 Codage du champ NumberOfTxPortGroups

Ce champ doit être codé en type de données Unsigned8 et doit être mis conformément au Tableau 542. Il doit uniquement dénombrer les entrées TxPortGroupArray qui ont abouti.

Tableau 542 – NumberOfTxPortGroups

Valeur (hexadécimale)	Signification
0x01, 0x03, 0x05, 0x07, 0x09, 0x0A, 0x0C, 0x0F	Valeurs admises
0x11, 0x13, 0x15, 0x17, 0x19, 0x1A, 0x1C, 0x1F	Valeurs admises
0x21	Valeurs admises
autre	Réservé

5.2.13.3 Codage du champ TxPortGroupArray

Le champ TxPortGroupArray est une matrice d'octets qui doit contenir entre un et 33 octets appelés octets TxPortGroup. Un octet de Tx Port Group doit consister en au moins une entrée TxPort et au plus huit entrées TxPort appelées entrée de TxPortGroup 0 à entrée de TxPortGroup 7. Par conséquent, le nombre d'octets de TxPortGroup correspond au nombre de ports sur un appareil et doit être calculé comme suit

$$N = M_{\text{highest}} \text{ DIV } 8 + 1 \quad (57)$$

où

N	est le nombre d'octets de TxPortGroup ou le nombre d'éléments matriciels
M_{highest}	est le nombre le plus élevé de TxPorts au sein d'un appareil (255 maximum)
8	est la taille de bit d'un octet

$$G = m \text{ DIV } 8 + 1 \quad (58)$$

où

G	est le nombre d'octets de TxPortGroup contenant le TxPort
m	est le nombre du TxPort actuel avec $1 \leq m \leq M$
8	est la taille de bit d'un octet

$$B = m \text{ MOD } 8 \quad (59)$$

où

B	est le nombre du bit dans l'octet de TxPortGroup contenant TxPort
m	est le nombre du TxPort actuel avec $1 \leq m \leq M$
8	est la taille de bit d'un octet

NOTE Le terme DIV signifie division sans reste. Le terme MOD signifie reste de la division.

Le dernier octet TxPortGroup (octet N) peut ne pas contenir toutes les 8entrées TxPort. Si $M_{\text{highest}} \text{ MOD } 8 \neq 7$, l'octet N n'est pas totalement rempli et les bits restants doivent être zéros (bits de bourrage).

TxPortGroup des appareils TxPorts doit être structuré dans l'ordre croissant sans espace. Ce codage de ce champ doit être conforme à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0: TxPortEntry_0

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 0$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Tableau 543 – TxPortEntry

Valeur (hexadécimale)	Signification
0x00	Emission désactivée
0x01	Emission activée

TxPort de l'injection locale est toujours placé dans TxPortEntry_0 du premier octet TxPortGroup.

Bit 1: TxPortEntry_1

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 1$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Bit 2: TxPortEntry_2

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 2$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Bit 3: TxPortEntry_3

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 3$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Bit 4: TxPortEntry_4

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 4$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Bit 5: TxPortEntry_5

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 5$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Bit 6: TxPortEntry_6

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 6$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

Bit 7: TxPortEntry_7

Ce bit doit être défini avec les valeurs conformes au Tableau 543 si le TxPort avec le numéro qui correspond à $m \text{ MOD } 8 = 7$ est présent. Un bit de bourrage doit être utilisé en l'absence de TxPort.

5.2.13.4 Codage du champ FrameDetails

Ce champ doit être codé conformément à 3.4.2.3 et la signification des bits individuels doit être la suivante:

Bit 0 – 1: FrameDetails.SyncFrame

Ce champ doit être codé avec les valeurs conformes au Tableau 544 et au Tableau 545.

**Tableau 544 – FrameDetails.SyncFrame conjointement avec
FrameDataProperties.ForwardingMode:= "Absolute mode"**

Valeur (hexadécimale)	Signification	Description
0x00	Aucune trame de synchronisation	Obligatoire
0x01	Trame de synchronisation primaire	Legacy (version originelle)
0x02	Trame de synchronisation secondaire	Les trames de synchronisation primaire et secondaire utilisent le même FrameID. Une seule trame doit être valide à un instant donné.
0x03	Réservé	—

Tableau 545 – FrameDetails.SyncFrame conjointement avec FrameDataProperties.ForwardingMode:= "Relative mode"

Valeur (hexadécimale)	Signification	Description
0x00	Aucune trame de synchronisation	Obligatoire
Autre	Réservé	—

Bit 2 – 3: FrameDetails.MeaningFrameSendOffset

Ce champ doit être codé avec les valeurs conformes au Tableau 546.

Tableau 546 – FrameDetails.MeaningFrameSendOffset

Valeur (hexadécimale)	Signification	Utilisation
0x00	Le contenu du champ FrameSendOffset spécifie le point dans le temps pour la transmission d'une trame. Pour le destinataire, le contenu du champ spécifie le point dans le temps pour la réception d'une trame.	Obligatoire
autre	Réservé	—

Bit 4 – 6: FrameDetails.reserved

Ce champ doit être mis à zéro.

Bit 7: FrameDetails.MediaRedundancyWatchDog

Ce champ doit être mis conformément au Tableau 547.

Tableau 547 – FrameDetails.MediaRedundancyWatchDog

Valeur (hexadécimale)	Signification	Signification
0x00	Désactivation	Ne surveille pas la qualité d'une connexion MRPD
0x01	Activation	Surveille la qualité d'une connexion MRPD et signale un changement

5.2.13.5 Codage du champ FrameDataProperties

Ce codage de ce champ doit être conforme à 3.4.2.5 et la signification des bits individuels doit être la suivante:

Bit 0: FrameDataProperties.ForwardingMode

Ce champ doit être mis conformément au Tableau 548.

Tableau 548 – FrameDataProperties.ForwardingMode

Valeur (hexadécimale)	Signification
0x00	Mode absolu Le FrameSendOffset contient l'heure d'envoi pour une trame transférée
0x01	Mode relatif Le FrameSendOffset doit contenir l'heure d'envoi d'une trame transférée ou la valeur "Best effort".

Bit 1 – 2: FrameDataProperties.FastForwardingMulticastMACAdd

Ce champ doit être mis conformément au Tableau 549.

Tableau 549 – FrameDataProperties.FastForwardingMulticastMACAdd

Valeur (hexadécimale)	Signification
0x00	Legacy (version originelle) Utilise l'adresse de diffusion unique de destination de MAC de l'interface
0x01	Utilise l'adresse de multidiffusion de destination RT_CLASS_3
0x02	Utilise FastForwardingMulticastMACAdd
0x03	Réservé

Bit 3 – 4: FrameDataProperties.FragmentationMode

Ce champ doit être codé avec les valeurs conformes au Tableau 550.

Tableau 550 – FrameDataProperties.FragmentationMode

Valeur (hexadécimale)	Signification	Utilisation
0x00	Aucune fragmentation	Obligatoire
0x01	Fragmentation activée Taille de fragment maximale pour une fragmentation statique 128 octets	Facultatif
0x02	Fragmentation activée Taille de fragment maximale pour une fragmentation statique 256 octets	Facultatif
0x03	Réservé	—

Bit 5 – 15: FrameDataProperties.reserved_1

Ce champ doit être mis à zéro.

Bit 16 – 31: FrameDataProperties.reserved_2

Ce champ doit être mis conformément à 3.4.2.2.

5.2.13.6 Codage du champ AdjustProperties

Ce champ doit être codé en type de données Unsigned16 avec la valeur zéro.

5.2.13.7 Codage du champ MAUType

Ce champ doit être codé en type de données Unsigned16 avec les valeurs conformes à RFC 4836, au Tableau 551 et au Tableau 552.

Tableau 551 – MAUType

Valeur (hexadécimale)	Signification	Utilisation
0x0000	Réservé à MediaType "Câble en cuivre" et "Câble de Fibre Optique"	Par défaut pour LinkState.Link != "UP"
	Utilisé pour MediaType "Communication Radio"	Par défaut pour la communication radio
0x0005	10BaseT	PDPortDataReal
0x000A	10BaseTXHD	PDPortDataReal
0x000B	10BaseTXFD	PDPortDataReal
0x000C	10BaseFLHD	PDPortDataReal
0x000D	10BaseFLFD	PDPortDataReal

Valeur (hexadécimale)	Signification	Utilisation
0x000F	100BaseTXHD	PDPortDataReal
0x0010	Default (MediaType Copper) 100BaseTXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0011	100BaseFXHD	PDPortDataReal
0x0012	Default (MediaType Fiber optic) 100BaseFXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0015	1000BaseXHD	PDPortDataReal
0x0016	1000BaseXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0017	1000BaseLXHD	PDPortDataReal
0x0018	1000BaseLXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0019	1000BaseSXHD	PDPortDataReal
0x001A	1000BaseSXFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x001D	1000BaseTHD	PDPortDataReal
0x001E	1000BaseTFD	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x001F	10GbaseFX	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0020	10GbaseLX4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0021	10GbaseR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0022	10GbaseER	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0023	10GbaseLR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0024	10GbaseSR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0025	10GbaseW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0026	10GbaseEW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0027	10GbaseLW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0028	10GbaseSW	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0029	10GbaseCX4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002A	2BaseTL	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002B	10PassTS	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002C	100BaseBX10D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002D	100BaseBX10U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002E	100BaseLX10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x002F	1000BaseBX10D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck

Valeur (hexadécimale)	Signification	Utilisation
0x0030	1000BaseBX10U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0031	1000BaseLX10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0032	1000BasePX10D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0033	1000BasePX10U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0034	1000BasePX20D	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0035	1000BasePX20U	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0036	10GBaseT ou 100BasePXF	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0037	10GBaseLRM	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0038	1000BaseKX	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0039	1000BaseKX4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003A	1000BaseKR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003B	10G1GBasePRXD1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003C	10G1GBasePRXD2	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003D	10G1GBasePRXD3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003E	10G1GBasePRXU1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x003F	10G1GBasePRXU2	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0040	10G1GBasePRXU3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0041	10GBasePRD1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0042	10GBasePRD2	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0043	10GBasePRD3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0044	10GBasePRU1	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0045	10GBasePRU3	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0046	40GbaseKR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0047	40GbaseCR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0048	40GbaseSR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x0049	40GbaseFR	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck

Valeur (hexadécimale)	Signification	Utilisation
0x004A	40GbaseLR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004B	100GbaseCR10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004C	100GbaseSR10	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004D	100GbaseLR4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
0x004E	100GbaseER4	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
IANA dependent	100BasePXF	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
Numéros connus utilisables	Vitesse de 100 Mbit/s (et plus) et duplicité complète	PDPortDataReal, PDPortDataAdjust, PDPortDataCheck
Numéros connus inutilisables	Vitesse inférieure à 100 Mbit/s ou duplicité à 50 %.	PDPortDataReal
autre	Réservé	—

NOTE Le MAUType 100BasePXF a deux codages pour des raisons d'héritage.

Tableau 552 – Valid combinations between MAUType and LinkState

LinkState.Port	LinkState.Link	MAUType	Utilisation
Unknown Disabled / Discarding Blocking Listening Learning Forwarding Broken	Up (des paquets)	10BaseT	PDPortDataReal
		10BaseTXHD	
		10BaseTXFD	
		10BaseFLHD	
		10BaseFLFD	
		100BaseTXHD	
		100BaseTXFD (Default)	
		100BaseFXHD	
		100BaseFXFD	
		1000BaseXHD	
		1000BaseXFD	
		1000BaseLXHD	
		1000BaseLXFD	
		1000BaseSXHD	
		1000BaseSXFD	
		1000BaseTHD	
		1000BaseTFD	
		10GigBaseFX	
		100BaseLX10	
		100BasePXF	
		Numéros connus utilisables	
		Numéros connus inutilisables	
	Down	Réservé	

LinkState.Port	LinkState.Link	MAUType	Utilisation
	Testing (dans certains modes d'essai)		
	Inconnu (l'état ne peut pas être déterminé)		
Réservé	Réservé		

5.2.13.8 Codage du champ CheckSyncMode

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0: CheckSyncMode.CableDelay

Ce champ doit être mis conformément au Tableau 553.

Tableau 553 – CheckSyncMode.CableDelay

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Pas de vérification
0x01	ON	Vérification de la différence entre le délai de câble mesuré distant et local par rapport à 50 ns.

Bit 1: CheckSyncMode.SyncMaster

Ce champ doit être mis conformément au Tableau 554.

Tableau 554 – CheckSyncMode.SyncMaster

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Pas de vérification
0x01	ON	Vérification de PTCP_MasterSourceAddress entre le local et le distant à l'aide de LLDP_PNIO_PTCPSTATUS.

Bit 2 – 15: CheckSyncMode.reserved

Ce champ doit être mis conformément à 3.4.2.2.

5.2.13.9 Codage du champ MAUTypeMode

Ce codage de ce champ doit être conforme à 3.4.2.4 et la signification des bits individuels doit être la suivante:

Bit 0: MAUTypeMode.Check

Ce champ doit être mis conformément au Tableau 555.

Tableau 555 – MAUTypeMode.Check

Valeur (hexadécimale)	Signification	Utilisation
0x00	OFF	Pas de vérification
0x01	ON	Vérification de la différence de type MAU entre la valeur détectée locale et distante.