

**PUBLICLY
AVAILABLE
SPECIFICATION**

**IEC
PAS 62411**

First edition
2005-06

Real-time Ethernet PROFINET IO



Reference number
IEC/PAS 62411:2005(E)

Publication numbering

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

Consolidated editions

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

Further information on IEC publications

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

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

- **Catalogue of IEC publications**

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 On-line information is also available on recently issued publications, withdrawn and replaced publications, as well as corrigenda.

- **IEC Just Published**

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

- **Customer Service Centre**

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

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

PUBLICLY AVAILABLE SPECIFICATION

IEC
PAS 62411

First edition
2005-06

Real-time Ethernet PROFINET IO

© IEC 2005 — Copyright - all rights reserved

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

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



Commission Electrotechnique Internationale
International Electrotechnical Commission
Международная Электротехническая Комиссия

PRICE CODE **XS**

For price, see current catalogue

LICENSED TO MECON Limited, - RANCHI/BANGALORE
FOR INTERNAL USE AT THIS LOCATION ONLY, SUPPLIED BY BOOK SUPPLY BUREAU.

CONTENTS

FOREWORD.....	16
INTRODUCTION.....	18
1 Scope	19
2 Normative references	20
3 Terms and definitions	21
3.1 Summary	21
3.2 Terms and definitions from other ISO/IEC standards	21
3.3 Terms and definitions from IEC 61158-5.....	22
3.4 ISO/IEC 8802-3 and IEEE 802.1Q terms	22
3.5 IEC 61588 terms.....	22
3.6 ISO/IEC 7498-1 terms.....	23
3.7 ISO/IEC 8822 terms.....	23
3.8 ISO/IEC 9545 terms.....	23
3.9 ISO/IEC 8824 terms.....	23
3.10 ISO/IEC 8802-3 and IEEE 802.1Q terms	23
3.11 Fieldbus Application Layer specific definitions	23
3.12 Abbreviations and symbols	31
3.13 Conventions for Part 5 of IEC 61158	33
3.14 Conventions for Part 6 of IEC 61158	36
3.15 Conventions used in state machines	39
4 Part 5: Application Layer Service definition of Type 10 for decentralized periphery	42
4.1 Concepts.....	42
4.2 Data type ASE.....	42
4.3 Communication model specification	52
4.4 DCP service specification	269
5 Part 6: Application Layer protocol specification of Type 10 for decentralized periphery	277
5.1 FAL syntax description.....	277
5.2 Transfer syntax.....	286
5.3 FAL protocol state machines	353
5.4 AP-Context state machine.....	359
5.5 FAL Service Protocol Machines (FSPMs)	359
5.6 Application Relationship Protocol Machines (ARPMs).....	409
5.7 RPC	510
5.8 DLL Mapping Protocol Machines (DMPMs).....	511
5.9 Parameters for an IO Device	593
5.10 DCP protocol	593
Annex A (informative) Device Instances	620
BIBLIOGRAPHY	622

Figures

Figure 1 — Common structure of specific fields	37
Figure 2 — Common structure of specific fields for Octet 1 (High).....	38
Figure 3 — Common structure of specific fields for Octet 2 (Low)	38
Figure 4 — Common structure of specific fields for Octet 1 (High).....	38
Figure 5 — Common structure of specific fields for Octet 2	39
Figure 6 — Common structure of specific fields for Octet 3	39
Figure 7 — Common structure of specific fields for Octet 4 (Low)	39
Figure 8 — Data type class hierarchy	42
Figure 9 — Example of communication between controlling devices and field devices	54
Figure 10 — Example of communication between an engineering station and several controlling and field devices.....	54
Figure 11 — Example of communication between field devices and a server station	55
Figure 12 — Example of communication between field devices	55
Figure 13 — Structural units of one arbitrary API of an IO device (General)	57
Figure 14 — Example 1 structural units for interfaces and ports within API 0.....	58
Figure 15 — Example 2 structural units for interfaces and ports within API 0.....	59
Figure 16 — Overview of application processes.....	61
Figure 17 — IO device with APs, slots and subslots.....	61
Figure 18 — Application Process with application objects (APOs)	64
Figure 19 — Access to a remote APO	65
Figure 20 — Access to a remote APO for provider/consumer association	66
Figure 21 — Example of one AR with two AREPs	67
Figure 22 — Relation of a record data object to one real object	69
Figure 23 — Relation of a record data object to two real objects	69
Figure 24 — Overview IO ASE service interactions.....	75
Figure 25 — Example of a resource model at the alarm source	130
Figure 26 — General isochronous application model (example)	157
Figure 27 — ASE relations in an IO device operating in isochronous mode	162
Figure 28 — State machine relations in an IO device operating in isochronous mode	163
Figure 29 — SyncCtl state diagram	166
Figure 30 — OUTPUT state diagram	168
Figure 31 — INPUT state diagram.....	172
Figure 32 — Assignment of communication relationship to application relationship.....	227
Figure 33 — Implicit application relationship.....	230
Figure 34 — Example IO application relationship (one-to-one)	232
Figure 35 — Example IO application relationship one-to-many	233
Figure 36 — Overview ASE state machines for IO device	246
Figure 37 — State diagram application startup IO device	247
Figure 38 — State diagram for a submodule	254
Figure 39 — State diagram client during startup	264
Figure 40 — Example of RT Class 1 behavior at the local interface.....	268
Figure 41 — Example of RT Class 1 behavior at the local interface.....	268
Figure 42 — Example of dropping RT Class 1 frames because of local overload	269

Figure 43 — Example of network topology including slower wireless segments	269
Figure 44 — Sequence diagram for DCP (example)	270
Figure 45 — Coding of the data type BinaryDate	287
Figure 46 — Encoding of Time Of Day value	288
Figure 47 — Encoding of Time Difference value	288
Figure 48 — Encoding of Network Time value	288
Figure 49 — Encoding of Network Time Difference value	289
Figure 50 — Relationship among Protocol Machines	353
Figure 51 — Structuring of the protocol machines and adjacent layers in a IO controller	356
Figure 52 — Structuring of the protocol machines and adjacent layers in a IO device	357
Figure 53 — Structuring of the protocol machines within the DMPM (single port)	511
Figure 54 — Structuring of the protocol machines within the DMPM (bridge)	512
Figure 55 — Line delay measurement	513
Figure 56 — Synchronization and line delay measurement	514
Figure 57 — Delay accumulation	517
Figure 58 — Worst case Time deviation of Synchronization	517
Figure 59 — Structure of a Time Frame	518
Figure 60 — Hardware Arrangement for Processing Sync PDU	519
Figure 61 — Start up sequence	520
Figure 62 — Green and Red intervals and interval transitions	557
Figure 63 — Possible Time Inaccuracies	560
Figure 64 — Using Medium Redundancy	561
Figure 65 — Locating the Destination for redundant RT Frames	561
Figure A.1 — Instance model of PROFINET IO	620

Tables

Table 1 — State machine description elements	40
Table 2 — Description of state machine elements.....	40
Table 3 — Conventions used in state machines.....	40
Table 4 — PROFINET IO UUID	51
Table 5 — Requirements and features of PROFINET IO	53
Table 6 — Read.....	71
Table 7 — Write.....	73
Table 8 — Set Input.....	82
Table 9 — Set Input IOCS.....	83
Table 10 — Get Input.....	84
Table 11 — Get Input IOCS	85
Table 12 — New Input	86
Table 13 — Set Input APDU Data Status	86
Table 14 — New Input APDU Data Status	87
Table 15 — Read Input Data	89
Table 16 — Set Output	91
Table 17 — Set Output IOCS	92
Table 18 — Get Output	93
Table 19 — Get Output IOCS	94
Table 20 — New Output.....	94
Table 21 — Set Output APDU Data Status	95
Table 22 — New Output APDU Data Status.....	96
Table 23 — Read Output Data	97
Table 24 — Write Output Substitute Data	100
Table 25 — Read Logbook.....	103
Table 26 — Logbook Event	105
Table 27 — Ext Channel Error Type	109
Table 28 — Read Device Diagnosis	111
Table 29 — Diagnosis Event	114
Table 30 — Alarm Type	119
Table 31 — Channel Diagnosis	120
Table 32 — Manufacturer Specific Diagnosis.....	120
Table 33 — Submodule Diagnosis State.....	121
Table 34 — AR Diagnosis State	121
Table 35 — User Structure Identifier	121
Table 36 — Alarm Notification	125
Table 37 — Alarm Ack	128
Table 38 — Module State.....	133
Table 39 — Usage with respect to CR Type.....	135
Table 40 — Detail.....	135
Table 41 — ARInfo	136
Table 42 — Ident Info	136
Table 43 — Connect	137

Table 44 — Release	143
Table 45 — Abort.....	144
Table 46 — End Of Parameter	144
Table 47 — Application Ready	145
Table 48 — Read Expected Identification	147
Table 49 — Read Real Identification	149
Table 50 — Read Identification Difference.....	152
Table 51 — Write IsoM Data	158
Table 52 — Read IsoM Data	160
Table 53 — SYNCH Event	162
Table 54 — Primitives issued by the AL to the SyncCtl state machine	164
Table 55 — Primitives issued by the user to the SyncCtl state machine	164
Table 56 — Primitives issued by the user to the input state machine.....	164
Table 57 — Primitives issued by the user to the output state machine.....	164
Table 58 — Primitives issued by the SyncCtl to the output state machine.....	165
Table 59 — Primitives issued by the output to the SyncCtl state machine.....	165
Table 60 — Primitives issued by the SyncCtl to the input state machine.....	165
Table 61 — Primitives issued by the output to the input state machine	165
Table 62 — Primitives issued by the output state machine to the AL	165
Table 63 — Primitives issued by the AL to the output state machine	165
Table 64 — Primitives issued by the input state machine to the AL	166
Table 65 — Primitives issued by the AL to the input state machine	166
Table 66 — SyncCtl state table	167
Table 67 — OUTPUT state table	169
Table 68 — INPUT state table.....	172
Table 69 — Subslot Number for Interface Submodules	180
Table 70 — Subslot Number for Port Submodules	180
Table 71 — System Capabilities.....	182
Table 72 — Auto Negotiation Support And Status	183
Table 73 — MDI Power Support	183
Table 74 — Link Aggregation Status	184
Table 75 — Multiple Peers	184
Table 76 — Subslot Number for Interface Submodules	186
Table 77 — Frame IDs for RT Class 3	187
Table 78 — Sync Frame	187
Table 79 —FrameSendOffset	187
Table 80 — Tx Port Entry.....	188
Table 81 — Subslot Number for Sync Interface Submodules.....	189
Table 82 — Sync Properties Role.....	190
Table 83 — Sync Class.....	190
Table 84 — Write Expected Port Data	191
Table 85 — Write Adjusted Port Data	193
Table 86 — Read Real Port Data	195
Table 87 — Read Expected Port Data	198

Table 88 — Read Adjusted Port Data	200
Table 89 — Write IR Data	202
Table 90 — Read IR Data	205
Table 91 — Write Sync Data	208
Table 92 — Read Real Sync Data	210
Table 93 — Read Expected Sync Data	213
Table 94 — Read PDev Data	215
Table 95 — Sync State Info	220
Table 96 — CS status	222
Table 97 — Summertime	223
Table 98 — Synchronization Active	224
Table 99 — Announcement hour	224
Table 100 — Accuracy	224
Table 101 — Set time	225
Table 102 — Sync interval violation	226
Table 103 — MProvider Data Status	238
Table 104 — Frame ID	239
Table 105 — Read AR Data	243
Table 106 — State table application startup IO device (RT class 1 and 2)	248
Table 107 — State table for a submodule	255
Table 108 — State table client during startup	265
Table 109 — Device Conformance	266
Table 110 — Device Conformance Version 2	267
Table 111 — Timeout values for name resolution	267
Table 112 — DCP Get	271
Table 113 — Option	271
Table 114 — Suboptions for IP option	271
Table 115 — Suboptions for control option	272
Table 116 — Suboptions for DeviceProperties options	272
Table 117 — Suboption for DHCP	272
Table 118 — DCP Set	273
Table 119 — DCP Identify	274
Table 120 — DCP Identify Q	276
Table 121 — DLPDU syntax	277
Table 122 — APDU syntax	277
Table 123 — Substitutions	278
Table 124 — LT	289
Table 125 — TagControlInformation.Priority	290
Table 126 — FrameID	290
Table 127 — FrameID for PTP sync	291
Table 128 — FrameID for PTP delay request	291
Table 129 — FrameID for PTP additional delay request	291
Table 130 — FrameID for PTP additional delay response	291
Table 131 — FrameID for PTP sync for RT class 3	291

Table 132 — FrameID for PTP follow up.....	292
Table 133 — FrameID for PTP delay response	292
Table 134 — FrameID for PTP additional delay followup request	292
Table 135 — PTP_RTAFIags.LocalReceiveExtensions	293
Table 136 — PTP_RTAFIags.RemoteSendExtensions	293
Table 137 — PTP_RTAFIags.DelayExtensions	293
Table 138 — PTP_RTAFIags.FollowUp	293
Table 139 — PTP_RTAFIags.DelayMeasure.....	293
Table 140 — PTP TypeLength.Type	294
Table 141 — PTP_SubType	294
Table 142 — IOxS.Extension	295
Table 143 — IOCS.Instance.....	295
Table 144 — IOxS.DataState	295
Table 145 — CycleCounter Difference.....	296
Table 146 — DataStatus.State	296
Table 147 — DataStatus.DataValid	296
Table 148 — DataStatus.ProviderState	296
Table 149 — DataStatus.StationProblemIndicator.....	296
Table 150 — The bits in the TransferStatus in a RT frame (RT class 3).....	297
Table 151 — AlarmType	299
Table 152 — AlarmSpecifier.ChannelDiagnosis	299
Table 153 — AlarmSpecifier.ManufacturerSpecificDiagnosis.....	300
Table 154 — AlarmSpecifier.SubmoduleDiagnosisState.....	300
Table 155 — AlarmSpecifier.ARDiagnosticsState	300
Table 156 — RPCPacketType	301
Table 157 — RPCFlags	301
Table 158 — RPCFlags2.....	301
Table 159 — RPCDRep.Character- and IntegerEncoding.....	302
Table 160 — RPCDRep Octet 2 – Floating Point Representation	302
Table 161 — RPCObjectUUID.Data4.....	302
Table 162 — RPCObjectUUID – defined values	303
Table 163 — RPCInterfaceUUID – defined values	303
Table 164 — RPCOperationNmb (IO device, controller and supervisor).....	304
Table 165 — RPCOperationNmb for endpoint mapper	304
Table 166 — RPCDataRepresentationUUID – defined values	305
Table 167 — BlockType	306
Table 168 — SlotNumber	308
Table 169 — SubslotNumber.....	308
Table 170 — Index (user specific)	308
Table 171 — Index (subslot specific).....	309
Table 172 — Index (slot specific)	309
Table 173 — Index (AR specific)	310
Table 174 — Index (API specific)	310
Table 175 — Index (device specific).....	311

Table 176 — UDP_SrcPort.....	311
Table 177 — UDP_DstPort.....	312
Table 178 — IP_DstIPAddress	312
Table 179 — RPCInquiryType	313
Table 180 — RPCEPMapStatus	314
Table 181 — ARTYPE.....	315
Table 182 — IOCRMulticastMACAdd.....	316
Table 183 — PTP sync multicast address.....	316
Table 184 — PTP follow up multicast address	316
Table 185 — PROFINET OUI	316
Table 186 — ARProperties.State.....	317
Table 187 — ARProperties.SupervisorTakeoverAllowed	317
Table 188 — ARProperties. ParametrizationServer.....	317
Table 189 — ARProperties.DataRate	317
Table 190 — ARProperties.DeviceAccess	318
Table 191 — IOCRProperties.RTClass	318
Table 192 — IOCRProperties. MProviderDataStatus.....	318
Table 193 — IOCRTagHeader.IOCRVLANID	319
Table 194 — IOCRTagHeader.IOUserPriority	319
Table 195 — IOCRType	319
Table 196 — CMInitiatorActivityTimeoutFactor	319
Table 197 — LengthIOCS	321
Table 198 — LengthIOPS	321
Table 199 — AlarmCRProperties.Priority	322
Table 200 — AlarmCRProperties.Transport	322
Table 201 — AlarmCRTagHeaderHigh.AlarmCRVLANID.....	322
Table 202 — AlarmCRTagHeaderHigh.AlarmUserPriority.....	322
Table 203 — AlarmCRTagHeaderLow.AlarmCRVLANID	323
Table 204 — AlarmCRTagHeaderLow.AlarmUserPriority	323
Table 205 — AlarmSequenceNumber	323
Table 206 — AlarmCRType.....	323
Table 207 — RTATimeoutFactor	324
Table 208 — AddressResolutionProperties.Protocol	324
Table 209 — AddressResolutionProperties.Factor	324
Table 210 — ModuleIdentNumber	325
Table 211 — SubmoduleIdentNumber	325
Table 212 — ControlCommand.PrmEnd	327
Table 213 — ControlCommand.ApplicationReady	327
Table 214 — ControlCommand.Release	327
Table 215 — ControlCommand.Done	327
Table 216 — DataDescription.Type	327
Table 217 — Values of ReductionRatio	328
Table 218 — Values of Phase	329
Table 219 — Values of Sequence	329

Table 220 — DataHoldFactor	330
Table 221 — WatchdogFactor	330
Table 222 — Values of FrameSendOffset	330
Table 223 — Values of ErrorCode for negative responses	331
Table 224 — Values of ErrorDecode	331
Table 225 — Coding of ErrorCode1 with ErrorDecode PNORW	331
Table 226 — Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO	332
Table 227 — Values of ErrorCode2 for ErrorCode1=RPC	334
Table 228 — ModuleState	334
Table 229 — SubmoduleState.AddInfo	335
Table 230 — SubmoduleState.DiagInfo	335
Table 231 — SubmoduleState.ARInfo	335
Table 232 — SubmoduleState.IdentInfo	335
Table 233 — SubmoduleState.FormatIndicator	335
Table 234 — SubmoduleState.Detail	336
Table 235 — SubmoduleState.FormatIndicator	336
Table 236 — SubmoduleProperties.Type	336
Table 237 — SubmoduleProperties.SharedInput	337
Table 238 — SubmoduleProperties.ReduceInputSubmoduleDataLength	337
Table 239 — SubmoduleProperties.ReduceOutputSubmoduleDataLength	337
Table 240 — SubstitutionMode	337
Table 241 — SubstituteActiveFlag	338
Table 242 — IM_Hardware_Revision	338
Table 243 — IM_SWRevision_Functional_Enhancement	338
Table 244 — IM_SWRevision_Bug_Fix	338
Table 245 — IM_SWRevision_Internal_Change	339
Table 246 — IM_Revision_Counter	339
Table 247 — IM_Profile_ID	339
Table 248 — IM_Profile_Specific_Type	339
Table 249 — IM_Version_Major	339
Table 250 — IM_Version_Minor	339
Table 251 — Values of NCAFaultStatus	341
Table 252 — Values of NCARectStatus	341
Table 253 — UserStructureIdentifier	342
Table 254 — ChannelErrorType	342
Table 255 — ChannelNumber	343
Table 256 — ChannelProperties.Type	343
Table 257 — ChannelProperties.Specifier	344
Table 258 — ChannelProperties.Direction	344
Table 259 — ExtChannelErrorType	344
Table 260 — RxPort	345
Table 261 — TxPortEntry	346
Table 262 — FrameDetails.SyncFrame	347

Table 263 — FrameDetails. MeaningFrameSendOffset	347
Table 264 — AdjustProperties.StorageMode	347
Table 265 — MAUType	347
Table 266 — DomainBoundary	348
Table 267 — SyncProperties.Role	348
Table 268 — SyncProperties.SyncClass	348
Table 269 — MRP_Type	349
Table 270 — MRP_Command	350
Table 271 — MRP_Port	350
Table 272 — MRP_Info	350
Table 273 — MRP_Counter	350
Table 274 — MRP_Transition	351
Table 275 — MRP_TimeStamp	351
Table 276 — ArgsLength check	351
Table 277 — ARBlockReq check	352
Table 278 — Assignment of state machines	355
Table 279 — Primitives issued by AP-Context (FAL user) to FSPMDEV	360
Table 280 — Primitives issued by FSPMDEV to AP-Context (FAL user)	366
Table 281 — FSPMDEV protocol machine for multicast communication	372
Table 282 — Functions used by AP-Context (FAL user) to FSPMDEV	376
Table 283 — Function used by FSPMDEV to AP-Context (FAL user)	380
Table 284 — Primitives issued by AP-Context (FAL user) to FSPMCTL	385
Table 285 — Primitives issued by FSPMCTL to AP-Context (FAL user)	389
Table 286 — Function used by AP-Context (FAL user) to FSPMCTL	396
Table 287 — Functions used by FSPMCTL to AP-Context (FAL user)	403
Table 288 — Primitives issued by FSPMDEV or FSPMCTL to PPM	409
Table 289 — Primitives issued by PPM to FSPMDEV or FSPMCTL	410
Table 290 — Primitives issued by CMDEV or CMCTL to PPM	410
Table 291 — Primitives issued by PPM to CMDEV or CMCTL	410
Table 292 — Primitives issued by LMPM to PPM	411
Table 293 — Primitives issued by PPM to LMPM	411
Table 294 — PPM state table	412
Table 295 — Functions used by the PPM	414
Table 296 — Primitives issued by FSPMDEV or FSPMCTL to CPM	415
Table 297 — Primitives issued by CPM to FSPM	415
Table 298 — Primitives issued by CMDEV or CMCTL to CPM	416
Table 299 — Primitives issued by CPM to CMCTL or CMDEV	416
Table 300 — Primitives issued by LMPM to CPM	416
Table 301 — Primitives issued by CPM to LMPM	416
Table 302 — CPM state table	417
Table 303 — Functions used by the CPM	420
Table 304 — Primitives issued by FSPMDEV or FSPMCTL to ALPMI	421
Table 305 — Primitives issued by ALPMI to FSPMDEV or FSPMCTL	421
Table 306 — Primitives issued by CMDEV or CMCTL to ALPMI	421

Table 307 — Primitives issued by ALPMI to CMCTL or CMDEV	422
Table 308 — Primitives issued by APMR to ALPMI	422
Table 309 — Primitives issued by ALPMI to APMR	422
Table 310 — Primitives issued by APMS to ALPMI	422
Table 311 — Primitives issued by ALPMI to APMS	423
Table 312 — ALPMI state table	423
Table 313 — Primitives issued by FSPMDEV or FSPMCTL to ALPMR.....	426
Table 314 — Primitives issued by ALPMR to FSPMDEV or FSPMCTL.....	426
Table 315 — Primitives issued by CMDEV or CMCTL to ALPMR.....	427
Table 316 — Primitives issued by ALPMR to CMCTL or CMDEV.....	427
Table 317 — Primitives issued by APMR to ALPMR.....	427
Table 318 — Primitives issued by ALPMR to APMR.....	428
Table 319 — Primitives issued by APMS to ALPMR.....	428
Table 320 — Primitives issued by ALPMR to APMS.....	428
Table 321 — ALPMR state table	429
Table 322 — Primitives issued by ALPMI/ALPMR to APMS.....	432
Table 323 — Primitives issued by APMS to ALPMI/ALPMR.....	432
Table 324 — Primitives issued by LMPM to APMS.....	432
Table 325 — Primitives issued by APMS to LMPM.....	432
Table 326 — APMS state table.....	433
Table 327 — Functions used by the APMS and APMR.....	437
Table 328 — Primitives issued by ALPMI/ALPMR to APMR.....	437
Table 329 — Primitives issued by APMR to ALPMI/ALPMR.....	437
Table 330 — APMR state table	438
Table 331 — Primitives issued by CMCTL to NRPM	441
Table 332 — Primitives issued by NRPM to CMCTL	442
Table 333 — Primitives issued by other machines to NRPM.....	443
Table 334 — Primitives issued by NRPM to other machines.....	444
Table 335 — NRPM state table	444
Table 336 — Functions used by the NRPM and RMPM.....	449
Table 337 — Primitives issued by CMDEV to RMPM.....	449
Table 338 — Primitives issued by RMPM to CMDEV.....	450
Table 339 — Primitives issued by RPC to RMPM.....	451
Table 340 — Primitives issued by RMPM to RPC.....	451
Table 341 — Primitives issued by other machines to RMPM	451
Table 342 — Primitives issued by RMPM to other machines	452
Table 343 — RMPM state table.....	452
Table 344 — Primitives issued by FSPMDEV to CMDEV.....	460
Table 345 — Primitives issued by CMDEV to FSPMDEV.....	461
Table 346 — CMDEV state table	462
Table 347 — Primitives issued by CMDEV to NRMC.....	483
Table 348 — Primitives issued by NRMC to CMDEV	483
Table 349 — Primitives issued by CPM to NRMC	483
Table 350 — Primitives issued by NRMC to CPM	483

Table 351 — Primitives issued by other machines to NRM C	484
Table 352 — NRM C state table	484
Table 353 — Primitives issued by FSPMCTL to CMCTL.....	487
Table 354 — Primitives issued by CMCTL to FSPMCTL.....	488
Table 355 — CMCTL state table	489
Table 356 — Primitives issued by NRPM to RPC	510
Table 357 — Primitives issued by RPC to NRPM	510
Table 358 — Procedures of clock synchronization master.....	514
Table 359 — Example Procedures of clock synchronization slave	514
Table 360 — Primitives issued by FSPM to TMM	521
Table 361 — Primitives issued by TMM to the FSPM	521
Table 362 — Primitives issued by RCTL to TMM	522
Table 363 — Primitives issued by TMM to the RCTL.....	522
Table 364 — Primitives issued by other machines to TMM	522
Table 365 — Primitives issued by TMM to other machines	522
Table 366 — Parameters used with primitives exchanged between FSPM, RCTL and TMM.....	522
Table 367 — TMM state table	524
Table 368 — Primitives issued by TMM to RCTL	538
Table 369 — Primitives issued by RCTL to the TMM.....	538
Table 370 — Primitives issued by SYN to RCTL	538
Table 371 — Primitives issued by RCTL to SYN	539
Table 372 — Parameters used with primitives exchanged between TMM, SYN, FW and RCTL	539
Table 373 — RCTL state table	539
Table 374 — Data resource	548
Table 375 — Primitives issued by LMPM-User to LMPM	550
Table 376 — Primitives issued by LMPM to the LMPM-User	550
Table 377 — Parameters used with primitives exchanged between LMPM-User and LMPM	551
Table 378 — LMPM state table	552
Table 379 — LMPM function table.....	554
Table 380 — Primitives issued by MMAC to FW	564
Table 381 — Primitives issued by FW to MMAC	564
Table 382 — Parameters used with primitives exchanged between MMAC and FW	564
Table 383 — FW state table.....	565
Table 384 — Primitives issued by FSPM to IFW	570
Table 385 — Primitives issued by IFW to FSPM	570
Table 386 — Parameters used with primitives exchanged between LMPM and IFW.....	570
Table 387 — IFW state table.....	570
Table 388 — IFW function table	573
Table 389 — Primitives issued by LMPM to MMAC	574
Table 390 — Primitives issued by MMAC to LMPM	574
Table 391 — Primitives issued by MMAC to MAC	574
Table 392 — Primitives issued by MAC to MMAC	575

Table 393 — Primitives issued by TMM to MMAC	575
Table 394 — Primitives issued by MMAC to TMM	575
Table 395 — Primitives issued by MMAC to RCTL	575
Table 396 — Primitives issued by RCTL to MMAC	576
Table 397 — Primitives issued by MMAC to FW	576
Table 398 — Primitives issued by FW to MMAC	577
Table 399 — MMAC state table	578
Table 400 — MMAC function table	591
Table 401 — Bus parameter/reaction times for an IO device	593
Table 402 — APDU syntax	593
Table 403 — Substitutions	594
Table 404 — Service-ID	595
Table 405 — Service-Type	595
Table 406 — Option	596
Table 407 — Status (Request)	597
Table 408 — Result	597
Table 409 — suboptions for control option	597
Table 410 — suboption Signal	597
Table 411 — Status (suboption Response)	598
Table 412 — suboptions for IP option	598
Table 413 — ResponseStatus (suboption IP-parameter)	599
Table 414 — suboptions for DeviceProperties options	599
Table 415 — suboption Device-role-details	600
Table 416 — suboption for DHCP	601
Table 417 — Suboptions for AllSelector option	602
Table 418 — Suboptions for Identify response content	602
Table 419 — Primitives issued by NRPM to DCPUCS	603
Table 420 — Primitives issued by DCPUCS to NRPM	603
Table 421 — Primitives issued by DCPUCS to LMPM	604
Table 422 — Primitives issued by LMPM to DCPUCS	604
Table 423 — Parameters used with primitives exchanged between FSPM and DCPUCS	604
Table 424 — DCPUCS state table	605
Table 425 — Primitives issued by RMPM to DCPUCR	608
Table 426 — Primitives issued by DCPUCR to RMPM	608
Table 427 — Primitives issued by DCPUCR to LMPM	609
Table 428 — Primitives issued by LMPM to DCPUCR	609
Table 429 — Parameters used with primitives exchanged between FSPM and DCPUCR	609
Table 430 — DCPUCR state table	610
Table 431 — Primitives issued by NRMC to DCPMCS	612
Table 432 — Primitives issued by DCPMCS to NRMC	612
Table 433 — Primitives issued by DCPMCS to LMPM	613
Table 434 — Primitives issued by LMPM to DCPMCS	613

Table 435 — Parameters used with primitives exchanged between FSPM and DCPMCS	613
Table 436 — DCPMCS state table.....	614
Table 437 — Primitives issued by RMPM to DCPMCR	616
Table 438 — Primitives issued by DCPMCR to RMPM	616
Table 439 — Primitives issued by DCPMCR to LMPM.....	616
Table 440 — Primitives issued by LMPM to DCPMCR.....	617
Table 441 — Parameters used with primitives exchanged between FSPM and DCPMCR	617
Table 442 — DCPMCR state table	618

INTERNATIONAL ELECTROTECHNICAL COMMISSION

Real-time Ethernet PROFINET¹ IO

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 provides no marking procedure to indicate its approval and cannot be rendered responsible for any equipment declared to be in conformity with an IEC Publication.
- 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.

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 patents concerning Real-Time Ethernet PROFINET IO.

IEC takes no position concerning the evidence, validity and scope of this patent right.

WO publication	Title (WO)	Derwent accession	Derwent publication
WO 02/043336 A1	System and method for the parallel transmission of real-time critical and non real-time critical data via switched data networks especially ethernet	2003-066596	CN1476702, 2004-02-18; DE10058524, 2002-06-13; EP1388238, 2004-02-11; JP2004515122, 2004-05-20; US2002064157, 2002-05-30; WO200243336, 2002-05-30
WO 02/076033 A1	Synchronous, clocked communication system with local input/output components and method for integrating local input/output components into such a system	2003-090512	DE10113261, 2002-09-26; DE10113261, 2003-07-10; EP1368935, 2003-12-10; US2002131454, 2002-09-19; WO200276033, 2002-09-26
WO 03/028258 A1	Method for synchronising nodes of a communication system	2003-333488	DE10229110, 2003-04-24; EP1430627, 2004-06-23; WO2003028258, 2003-04-03
WO 03/028259 A1	Communications system and method for synchronising a communications cycle	2003-333489	DE10147422, 2003-04-24; EP1430628, 2004-06-23; WO2003028259, 2003-04-03

The holder of this patent right has assured the IEC that he is willing to negotiate licenses under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holder of this patent right is registered with IEC. Information may be obtained from:

Siemens AG
A&D PT2
Nuremberg
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. The IEC shall not be held responsible for identifying any or all such patent rights.

¹ PROFINET is a trade name of PROFIBUS International (PI). This information is given for the convenience of users of this IEC/PAS and does not constitute an endorsement by IEC of the trademark holder or any of its products. Compliance to this standard does not require use of the trade name PROFINET. Use of the trade name PROFINET requires permission of PROFIBUS International.

A PAS is a technical specification not fulfilling the requirements for a standard but made available to the public .

IEC-PAS 62411 has been processed by subcommittee 65C: Digital communications, of IEC technical committee 65: Industrial-process measurement and control.

The text of this PAS is based on the following document:

This PAS was approved for publication by the P-members of the committee concerned as indicated in the following document

Draft PAS	Report on voting
65C/359/NP	65C/375/RVN

Following publication of this PAS, the technical committee or subcommittee concerned will transform it into an International Standard.

It is intended that the content of this PAS will be incorporated in the future new edition of the various parts of IEC 61158 series according to the structure of this series.

This PAS shall remain valid for an initial maximum period of three years starting from 2005-06. The validity may be extended for a single three-year period, following which it shall be revised to become another type of normative document or shall be withdrawn.

INTRODUCTION

This PAS provides an additional communication network with services and protocols to the existing IEC 61158 series. These additional communication network meet the industrial automation market objective of identifying Real-Time Ethernet (RTE) communication networks coexisting with ISO/IEC 8802-3 – commonly known as Ethernet. These RTE communication network use provisions from ISO/IEC 8802-3 for the lower communication stack layers and additionally provide more predictable and reliable real-time data transfer and means for support of precise synchronization of automation equipment. This includes a range of selectable and configurable options within their detailed specifications. In general, only certain restricted combinations of options will interwork correctly (see IEC 61158-1). The recommended combinations of options will be collected in another future International Standard (IEC 61784-2) dealing with RTE communication profiles (CP). IEC 61784-2 will provide users and implementers with details of different CP that is needed to define compliant and interoperable devices. Adoption of Ethernet technology for industrial communication between controllers and even for communication with field devices promotes use of Internet technologies in the field area. This availability would be unacceptable if it causes the loss of features required in the field area for industrial communication automation networks, such as:

- real-time,
- synchronized actions between field devices like drives,
- efficient, frequent exchange of very small data records.

Features of the current fieldbus systems, see IEC 61784-1, should be improved the same way as the properties of Ethernet networks in terms of transmission bandwidth and network span.

Another implicit but essential requirement is that the typical Ethernet communication capabilities as used in the office world are fully retained, so that the software involved remains applicable.

The market is in need of several network solutions, each with different performance characteristics and functional capabilities, matching the diverse application requirements. IEC 61784-2 will deal with a description of RTE performance indicators of a communication profile will enable the user to match compliant network devices to application dependant performance requirements of an RTE network.

This PAS follows the general structure and terms of IEC 61158 in order to support a future migration to IEC 61158 (new edition). It specifies the model and services of the fieldbus Application Layer of PROFINET IO. The part 6 and part 5 related protocol and services are merged in this PAS using a clause number instead a part number as in IEC 61158. This approach causes to shift the clauses one level and merge the common Clauses 1 to 4.

Real-time Ethernet PROFINET IO

1 Scope

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.”

The FAL is an Application Layer Communication Standard designed to support the conveyance of time-critical and non-time-critical application requests and responses among devices in an automation environment. The term “time-critical” is used to represent the presence of an application 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 PAS complies with the structure and services of the IEC fieldbus Application Layer. It is specified in conformance with the OSI Basic Reference Model (ISO/IEC 7498) and the OSI Application Layer Structure (ISO/IEC 9545).

FAL services and protocols are provided by FAL application-entities (AE) contained within the application processes. The FAL AE is composed of a set of object-oriented Application Service Elements (ASEs) and a Layer Management Entity (LME) that manages the AE. The ASEs provide communication services that operate on a set of related application process object (APO) classes. One of the FAL ASEs is a management ASE that provides a common set of services for the management of the instances of FAL classes.

The AL services within this PAS specify interactions between remote applications in terms of:

- an abstract model for defining application resources (objects) capable of being manipulated by users via the use of FAL Services,
- the primitives (interactions between the FAL and the FAL user) associated with each FAL Service;
- the parameters associated with each primitive;
- the interrelationship between and the valid sequences of the primitives for each service.

Although these services specify, from the perspective of applications, how request and responses are issued and delivered, they do not include a specification of what the requesting and responding applications are to do with them. That is, the behavioral aspects of the applications are not specified; only a definition of what requests and responses they can send/receive is specified. This permits greater flexibility to the FAL users in standardizing such object behavior. In addition to these services, some supporting services are also defined in this PAS provide access to the FAL to control certain aspects of its operation.

Isochronous Mode Application class specification containing IsoM Data, and IO Device to IO Device provider consumer relationship.

The AL protocols within this PAS specify interactions between remote applications in terms of:

- the encoding rules that are applied to all the Application Layer Protocol Data Units (APDUs);
- the formal Abstract Syntax definitions of such APDUs;
- the protocol state machine descriptions that handle the APDUs and the primitives in the correct sequences;
- the mappings of the APDUs to and from the Data Link Layer services.

The FAL encoding rules are designed assuming that both the encoder (sender) and the decoder (receiver) have the common knowledge of the abstract syntax. Wherever possible, data type identifiers are not encoded and transferred over the network.

NOTE This is why the Abstract Syntax Notation One / Basic Encoding Rule is not practical for the FAL.

The purpose of this PAS is to define the protocol provided to the Data Link Layer at the boundary between the Application and Data Link Layers.

The communication network profile template GSDML (see ISO 15745-4:2003/Amd 1) shall be usable to describe the selected features of a device for integration in a system.

2 Normative references

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

IEC 60559, *Binary floating-point arithmetic for microprocessor systems*

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

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

IEC 61588:2004, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61784-1:2003, *Digital data communications for measurement and control – Part 1: Profile sets for continuous and discrete manufacturing relative to fieldbus use in industrial control systems*

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

ISO 15745-4:2003/Amd 1, *Industrial automation systems and integration – Open systems application integration framework – Part 4: Reference description for Ethernet-based control systems*,
Amendment 1: *PROFINET profiles*²

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

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 – Part 1: The Basic Model*

ISO/IEC 8802-3:2000, *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*

ISO/IEC 8822:1994, *Information technology – Open Systems Interconnection – Presentation service definition*

ISO/IEC 8824 (all parts), *Information technology – Abstract Syntax Notation One (ASN.1)*

ISO/IEC 8825-1, *Information technology – Encoding rules – Part 1: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)*

ISO/IEC 8859-1:1998, *Information technology – 8-bit single-byte coded graphic character sets – Part 1: Latin alphabet No. 1*

² To be published

ISO/IEC 8877, *Information technology – Telecommunications and information exchange between systems – Interface connector and contact assignments for ISDN Basic Access Interface located at reference points S and T*

ISO/IEC 9545:1994, *Information technology – Open Systems Interconnection – Application Layer structure*

IEEE 802.1D:1998, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common specifications – Part 3: Media Access Control (MAC) bridges*

IEEE 802.1Q: 1998 *IEEE standards for local and metropolitan area networks – Virtual bridged local area networks*

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 3636, *Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)*, available at <http://www.ietf.org>

3 Terms and definitions

3.1 Summary

For the purpose of this document, the following definitions apply:

3.2 Terms and definitions from other ISO/IEC standards

3.2.1 Terms and definitions from ISO/IEC 7498-1

- 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.

3.2.2 Terms and definitions from ISO/IEC 9545

- 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.

3.2.3 Terms and definitions from ISO/IEC 8824

- a) object identifier;
- b) type;
- c) value;
- d) simple type;

- e) structured type;
- f) component type;
- g) tag;
- h) Boolean type;
- i) true;
- j) false;
- k) integer type;
- l) bitstring type;
- m) octetstring type;
- n) null type;
- o) sequence type;
- p) sequence of type;
- q) choice type;
- r) tagged type;
- s) any type;
- t) module;
- u) production.

3.2.4 Terms and definitions from ISO/IEC 8825-1

- a) encoding (of a data value);
- b) data value;
- c) Identifier Octets (the singular form is used in this PAS);
- d) Length Octet(s) (both singular and plural forms are used in this PAS);
- e) Contents Octets.

3.3 Terms and definitions from IEC 61158-5

- a) application relationship;
- b) conveyance path;
- c) client;
- d) error class;
- e) error code;
- f) name;
- g) numeric identifier;
- h) peer;
- i) pre-defined AR endpoint;
- j) pre-established AR endpoint;
- k) publisher;
- l) subscriber;
- m) server.

3.4 ISO/IEC 8802-3 and IEEE 802.1Q terms

For the purposes of this PROFINET IO, the following terms as defined in ISO/IEC 8802-3 apply.

- a) Canonical Format Indicator;
- b) Destination address;
- c) Frame Check Sequence;
- d) Length/Type;
- e) MAC Client Data;
- f) MAC Frame;
- g) Pad;
- h) Preamble;
- i) Source address;
- j) Start Frame Delimiter;
- k) Tagged MAC frame;
- l) Tag Control Information;
- m) User Priority field;
- n) VLAN Identifier.

3.5 IEC 61588 terms

For the purposes of this PROFINET IO, the following terms as defined in IEC 61588 series apply.

- a) Clock;
- b) Clock Stratum;

3.6 ISO/IEC 7498-1 terms

- 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;

3.7 ISO/IEC 8822 terms

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

- a) abstract syntax;
- b) presentation context;

3.8 ISO/IEC 9545 terms

For the purposes of this PROFINET IO, 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;

3.9 ISO/IEC 8824 terms

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

- a) object identifier;
- b) type;

3.10 ISO/IEC 8802-3 and IEEE 802.1Q terms

For the purposes of PROFINET IO, the following terms as defined in ISO/IEC 8802-3 and IEEE 802.1Q apply.

- a) Canonical Format Indicator;
- b) Destination address;
- c) Frame Check Sequence;
- d) Length/Type;
- e) MAC Client Data;
- f) MAC Frame;
- g) Pad;
- h) Preamble;
- i) Source address;
- j) Start Frame Delimiter;
- k) Tagged MAC frame;
- l) Tag Control Information;
- m) User Priority field;
- n) VLAN Identifier;

3.11 Fieldbus Application Layer specific definitions

3.11.1

General

For the purposes of this PROFINET IO, the following definitions apply.

3.11.2
access protection

limitation of the usage of an application object to one client

3.11.3
alarm

activation of an event that shows a critical state

3.11.4
alarm ack

acknowledgment of an event that shows a critical state

3.11.5
alarm data object

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

3.11.6
allocate

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

3.11.7
application

function or data structure for which data is consumed or produced

3.11.8
application layer interoperability

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

3.11.9
application objects

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

3.11.10
application process

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

3.11.11
application process identifier

distinguishes multiple application processes used in a device

NOTE Application process identifier is assigned by PROFIBUS International (PI).

3.11.12
application process object

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

NOTE 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.

3.11.13
application process object class

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

3.11.14**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

3.11.15**application relationship application service element**

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

3.11.16**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 Each application process involved in the application relationship maintains its own application relationship endpoint.

3.11.17**attribute**

description of an externally visible characteristic or feature of an object

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

3.11.18**backup**

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

3.11.19**behavior**

indication of how an object responds to particular events

3.11.20**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 The channel typically represents a single connector or clamp as a real interface of a module or sub-module. This reference is used to identify points of failure within diagnosis PDUs.

3.11.21**channel related diagnosis**

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

EXAMPLE open loop

3.11.22**class**

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

NOTE 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.

3.11.23**class attributes**

attribute that is shared by all objects within the same class

3.11.24
class code

unique identifier assigned to each object class

3.11.25
class specific service

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

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

3.11.26
clear

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

3.11.27
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

3.11.28
communication data object

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

3.11.29
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

3.11.30
configuration fault

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

3.11.31
configuration identifier

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

3.11.32
consume

act of receiving data from a provider

3.11.33
consumer

node or sink receiving data from a provider

3.11.34
context management

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

NOTE Managing includes functions such as controlling, monitoring, and diagnosing.

3.11.35
conveyance path

unidirectional flow of APDUs across an application relationship

3.11.36**cyclic**

repetitive in a regular manner

3.11.37**data consistency**

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

3.11.38**device**

physical hardware connected to the link

NOTE A device may contain more than one node.

3.11.39**device ID**

a vendor assigned device type identification

3.11.40**device profile**

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

3.11.41**diagnosis data object**

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

3.11.42**diagnosis information**

all data available at the server for maintenance purposes

3.11.43**dynamic reconfiguration**

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

3.11.44**endpoint**

one of the communicating entities involved in a connection

3.11.45**engineering**

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

3.11.46**error**

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

3.11.47**error class**

general grouping for related error definitions and corresponding error codes

3.11.48**error code**

identification of a specific type of error within an error class

3.11.49

event

instance of a change of conditions

3.11.50

frame

unit of data transfer on a link

3.11.51

identification data object

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

3.11.52

implicit AR endpoint

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

3.11.53

index

address of a record data object within an application process

3.11.54

instance

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

3.11.55

instance attributes

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

3.11.56

instantiated

object that has been created in a device

3.11.57

invocation

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

NOTE 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.

3.11.58

IO controller

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

NOTE This is usually a programmable controller or a distributed control system.

3.11.59

IO data object

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

3.11.60

IO device

field device which acts as server for IO operation

3.11.61**IO parameter server**

server for application parameter of IO devices (client)

NOTE This is usually a device to backup parameter data and to log online changes of device parameter.

3.11.62**IO subsystem**

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

3.11.63**IO supervisor**

engineering device which manages commissioning and diagnosis of an IO system

3.11.64**IO system**

system composed of all its IO subsystems

NOTE As an example a PLC with more than one IO controller (network interface) controls one IO system composed of an IO subsystems for each IO controller.

3.11.65**Isochronous mode**

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

3.11.66**member**

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

3.11.67**method**

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

3.11.68**module**

hardware or logical component of a physical device

3.11.69**network**

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

3.11.70**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

3.11.71**object specific service**

service unique to the object class which defines it

3.11.72**operate**

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

3.11.73**peer**

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

3.11.74
physical device

an automation or other network device

3.11.75
point-to-point connection

connection that exists between exactly two application objects

3.11.76
primary

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

NOTE Besides a primary IO AR a backup IO AR may exist. In example used for redundancy and dynamic reconfiguration of IO data.

3.11.77
provider

node or source sending data to one or many consumer

3.11.78
PTP domain

a certain number of PTP subdomains in one IP subnet

3.11.79
PTP subdomain

a certain amount of DTEs with synchronized clocks

3.11.80
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/slot/subslot/index

3.11.81
resource

processing or information capability

3.11.82
run

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

3.11.83
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

3.11.84
service

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

3.11.85
slot

address of a structural unit within an IO device

NOTE Within a modular device, a slot typically addresses a physical module. Within compact devices, a slot typically addresses a logical function or virtual module.

3.11.86**stop**

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

3.11.87**submodule**

hardware or logical component of a module

3.11.88**subslot**

address of a structural unit within a slot

NOTE 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.

3.11.89**vendor ID**

central administrative number to unambiguously distinguish between device manufacturers

NOTE The vendor ID is assigned by PROFIBUS International (PI).

3.12**Abbreviations and symbols**

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
APO	Application Object
AR	Application Relationship
AREP	Application Relationship End Point
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
BMC	Best Master Clock
CM	Context Management
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
DCE	OSF Distributed Computing Environment
DCP	Discovery and basic Configuration Protocol

DCPMCR	DCP Multicast Receiver
DCPMCS	DCP Multicast Sender
DCPUCR	DCP Unicast Receiver
DCPUCS	DCP Unicast Sender
DHCP	Dynamic Host Configuration Protocol
DIM	Device Interface Module
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLL	Data Link Layer
DLPDU	Data Link-Protocol Data Unit
DLSDU	DL-service-data-unit
DNS	Domain Name Service
DTE	Date Terminal Equipment
FAL	Fieldbus Application Layer
FIFO	First In First Out
GSDML	Generic 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
Ind	Indication
IOCS	Input Output Object Consumer Status
IOPS	Input Output Object Provider Status
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
Isb	Least Significant Bit
LT	Length/Type

MAC	Medium Access Control
msb	Most Significant Bit
NCA	Network Computing Architecture
OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PL	Physical Layer
PTP	Precision Time Protocol
QoS	Quality of Service
Req	Request
RPC	Remote Procedure Call
Rsp	Response
RT	Real Time Protocol
RTA	Real Time Protocol Acyclic
RTC	Real Time Protocol Cyclic
RTE	Real Time Ethernet
SDU	Service Data Unit
TLV	Type Length Value (coding rule)
UDP	User Datagram Protocol
UUID	Universal Unique Identifier
VLAN	Virtual Local Area Network

3.13 Conventions for Part 5 of IEC 61158

3.13.1 Overview

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of two parts, its class specification, and its service specification.

The class specification defines the attributes of the class. The service specification defines the services that are provided by the ASE.

3.13.2 General conventions

This PAS uses the descriptive conventions given in ISO/IEC 10731

3.13.3 Conventions for class definitions

Class definitions are described using templates. Each template consists of a list of attributes for the class. The general form of the template is shown below:

FAL ASE:	ASE Name
CLASS:	Class Name
CLASS ID:	#
PARENT CLASS:	Parent Class Name

ATTRIBUTES:

- 1 (o) Key Attribute: numeric identifier
- 2 (o) Key Attribute: name
- 3 (m) Attribute: attribute name(values)
- 4 (m) Attribute: attribute name(values)
- 4.1 (s) Attribute: attribute name(values)
- 4.2 (s) Attribute: attribute name(values)
- 4.3 (s) Attribute: attribute name(values)
- 5. (c) Constraint: constraint expression
- 5.1 (m) Attribute: attribute name(values)
- 5.2 (o) Attribute: attribute name(values)
- 6 (m) Attribute: attribute name(values)
- 6.1 (s) Attribute: attribute name(values)
- 6.2 (s) Attribute: attribute name(values)

SERVICES:

- 1 (o) OpsService: service name
- 2 (c) Constraint: constraint expression
- 2.1 (o) OpsService: service name
- 3 (m) MgtService: service name

- (1) The "FAL ASE:" entry is the name of the FAL ASE that provides the services for the class being specified.
- (2) The "CLASS:" entry is the name of the class being specified. All objects defined using this template will be an instance of this class. The class may be specified by this PAS, or by a user of this PAS.
- (3) The "CLASS ID:" entry is a number that identifies the class being specified. This number is unique within the FAL ASE that will provide the services for this class. When qualified by the identity of its FAL ASE, it unambiguously identifies the class within the scope of the FAL. The value "NULL" indicates that the class cannot be instantiated. Class IDs between 1 and 255 are reserved by this PAS to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2048 are allocated for identifying user defined classes.
- (4) The "PARENT CLASS:" entry is the name of the parent class for the class being specified. All attributes defined for the parent class and inherited by it are inherited for the class being defined, and therefore do not have to be redefined in the template for this class.

NOTE The parent-class "TOP" indicates that the class being defined is an initial class definition. The parent class TOP is used as a starting point from which all other classes are defined. The use of TOP is reserved for classes defined by this PAS.

- (5) The "ATTRIBUTES" label indicate that the following entries are attributes defined for the class.
 - a) Each of the attribute entries contains a line number in column 1, a mandatory (m) / optional (o) / conditional (c) / selector (s) indicator in column 2, an attribute type label in column 3, a name or a conditional expression in column 4, and optionally a list of enumerated values in column 5. In the column following the list of values, the default value for the attribute may be specified.

- b) Objects are normally identified by a numeric identifier or by an object name, or by both. In the class templates, these key attributes are defined under the key attribute.
 - c) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting is used to specify
 - i) fields of a structured attribute (4.1, 4.2, 4.3),
 - ii) attributes conditional on a constraint statement (5). Attributes may be mandatory (5.1) or optional (5.2) if the constraint is true. Not all optional attributes require constraint statements as does the attribute defined in (5.2).
 - iii) the selection fields of a choice type attribute (6.1 and 6.2).
- (6) The "SERVICES" label indicates that the following entries are services defined for the class.
- a) An (m) in column 2 indicates that the service is mandatory for the class, while an (o) indicates that it is optional. A (c) in this column indicates that the service is conditional. When all services defined for a class are defined as optional, at least one has to be selected when an instance of the class is defined.
 - b) The label "OpsService" designates an operational service (1).
 - c) The label "MgtService" designates an management service (2).
 - d) The line number defines the sequence and the level of nesting of the line. Each nesting level is identified by period. Nesting within the list of services is used to specify services conditional on a constraint statement.

3.13.4 Conventions for service definitions

3.13.4.1 General

The service model, service primitives, and time-sequence diagrams used are entirely abstract descriptions; they do not represent a specification for implementation.

3.13.4.2 Service parameters

Service primitives are used to represent service user/service provider interactions (ISO/IEC 10731). They convey parameters which indicate information available in the user/provider interaction. In any particular interface, not all parameters need be explicitly stated.

The service specification of this PAS uses a tabular format to describe the component parameters of the ASE service primitives. The parameters which apply to each group of service primitives are set out in tables. Each table consists of up to five columns for the

- 1) Parameter name,
- 2) request primitive,
- 3) indication primitive,
- 4) response primitive, and
- 5) confirm primitive.

One parameter (or component of it) is listed in each row of each table. Under the appropriate service primitive columns, a code is used to specify the type of usage of the parameter on the primitive specified in the column:

- M parameter is mandatory for the primitive
- U parameter is a User option, and may or may not be provided depending on dynamic usage of the service user. When not provided, a default value for the parameter is assumed.
- C parameter is conditional upon other parameters or upon the environment of the service user.
- (blank) parameter is never present.
- S parameter is a selected item.

Some entries are further qualified by items in brackets. These may be

- a) a parameter-specific constraint:

“(=)” indicates that the parameter is semantically equivalent to the parameter in the service primitive to its immediate left in the table.

- b) an indication that some note applies to the entry:

“(n)” indicates that the following note "n" contains additional information pertaining to the parameter and its use.

3.13.4.3 Service procedures

The procedures are defined in terms of

- the interactions between application entities through the exchange of Fieldbus Application Protocol Data Units, and
- the interactions between an application layer service provider and an application layer service user in the same system through the invocation of application layer service primitives.

These procedures are applicable to instances of communication between systems which support time-constrained communications services within the Fieldbus Application Layer.

3.14 Conventions for Part 6 of IEC 61158

3.14.1 General concept

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. 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. The protocol specification for each of the ASEs is defined in this PAS.

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. The service specification defines the services that are provided by the ASE.

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

3.14.2 Conventions for PROFINET IO

3.14.2.1 Abstract syntax conventions

3.14.2.1.1 PDUs are described as octets or groups of octets

- a) 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
- b) If octets or groups of octets are grouped within “{ }” the order is arbitrary
- c) 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.
- d) Octets can be grouped or values can be assigned within “()”
- e) If octets or groups of octets are grouped within “[]” the group can be omitted
- f) Complex APDUs may be built by means of substitutions (sub-structures)
- g) 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 1440 octets in total.

NOTE 4 The APDU syntax for CL RPC implies that an IO controller supports a minimal ASDU size of 4096 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.

3.14.2.2 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.

3.14.2.3 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.

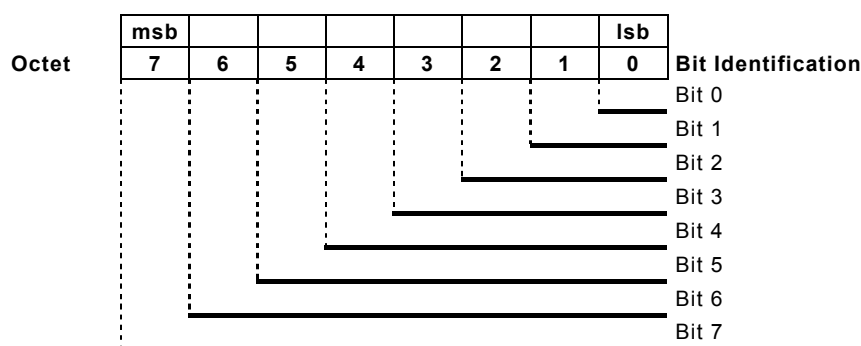


Figure 1 — Common structure of specific fields

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.

3.14.2.4 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.

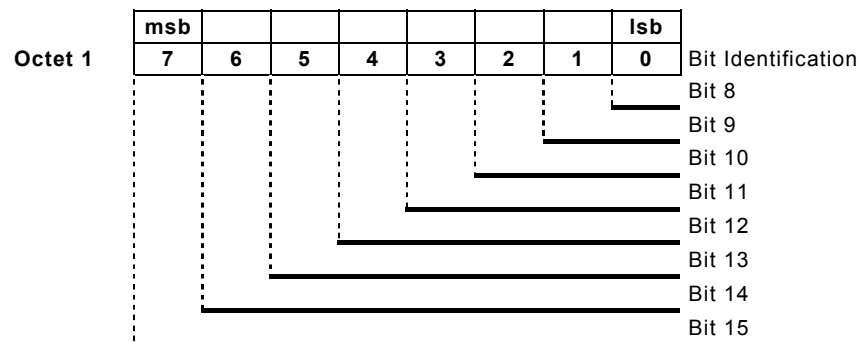


Figure 2 — Common structure of specific fields for Octet 1 (High)

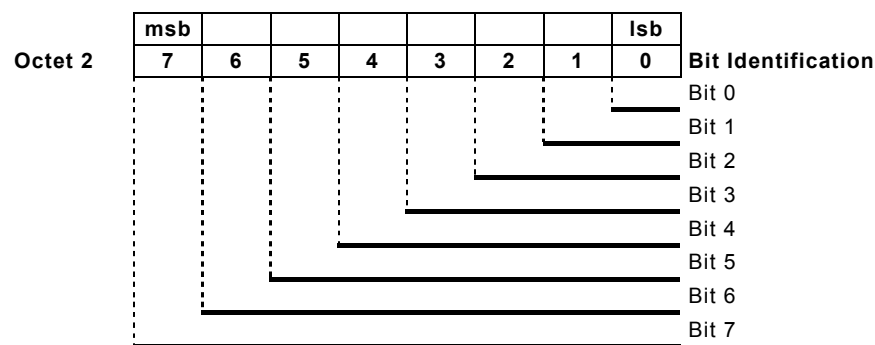


Figure 3 — Common structure of specific fields for Octet 2 (Low)

Several 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 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.

3.14.2.5 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.

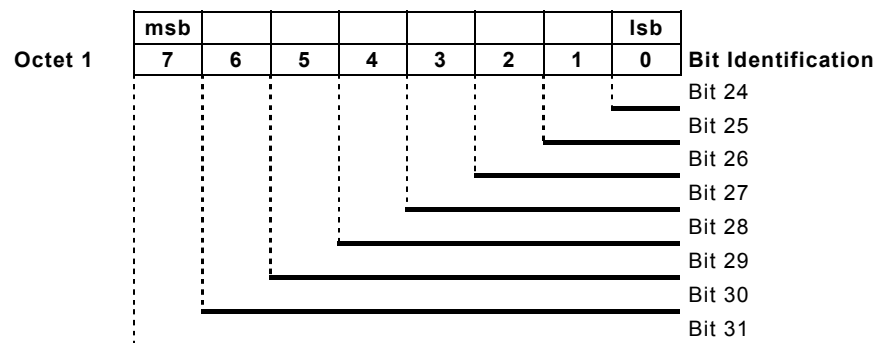


Figure 4 — Common structure of specific fields for Octet 1 (High)

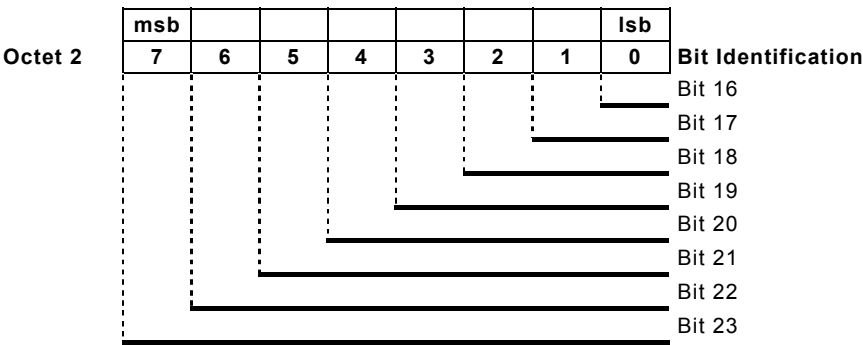


Figure 5 — Common structure of specific fields for Octet 2

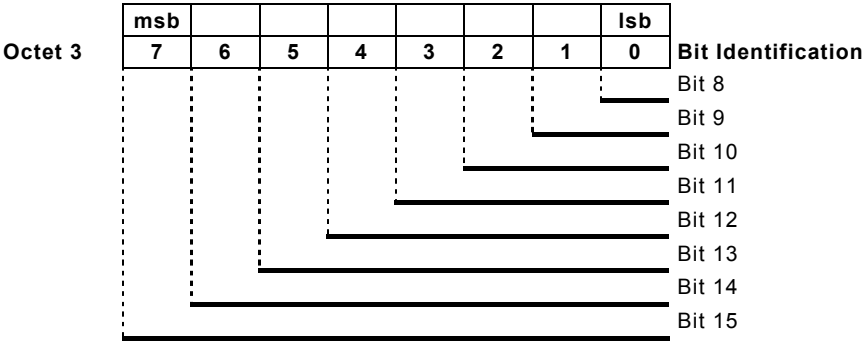


Figure 6 — Common structure of specific fields for Octet 3

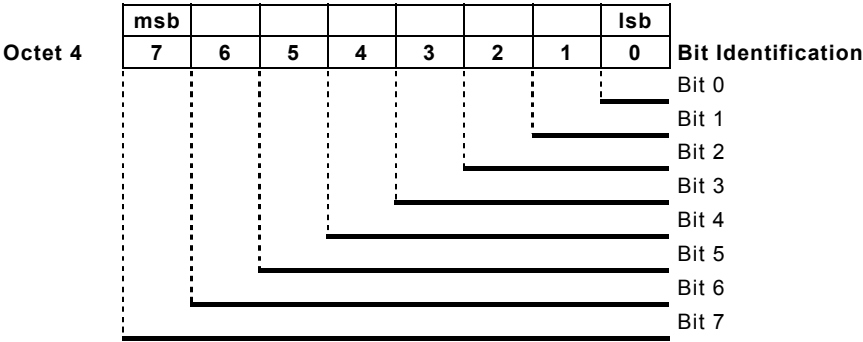


Figure 7 — Common structure of specific fields for Octet 4 (Low)

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.

3.15 Conventions used in state machines

3.15.1 State machine conventions for PROFINET IO

The protocol sequences are described by means of State Machines.

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

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

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

If the event occurs and the conditions are fulfilled the transition fires, i.e. 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 are 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.'
=	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.
<>	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”
<<	A semantic condition with the meaning “older”
&&	Logical “AND”
	Logical “OR”
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.

Readers are strongly recommended to refer to the subclauses 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”.

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.

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 Part 5: Application Layer Service definition of Type 10 for decentralized periphery

4.1 Concepts

The concepts of IEC 61158-5 are applied.

4.2 Data type ASE

4.2.1 General

4.2.1.1 Overview

Fieldbus data types specify the machine independent syntax for application data conveyed by FAL services. The fieldbus application layer supports the definition and transfer of both basic and constructed data types. Encoding rules for the data types specified in this Clause are provided in PROFINET IO Protocol Specification.

Basic types are atomic types that cannot be decomposed into more elemental types. Constructed types are types composed of basic types and other constructed types. Their complexity and depth of nesting is not constrained by this PAS.

Data types are defined as instances of the data type class, as shown in Figure 8. Only a subset of the data types defined in this Clause are shown in this figure. Defining new types is accomplished by providing a numeric Id and supplying values for the attributes defined for the data type class.

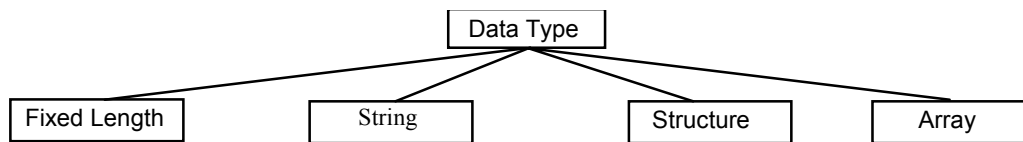


Figure 8 — Data type class hierarchy

The data type definitions in Figure 8 are represented as a class/format/instance structure beginning with data type class entitled "Data Type". The formats for data types are defined by the data type class and are represented in Figure 8.

The basic data classes are used to define fixed length and bitstring data types. Standard types taken from ISO/IEC 8824 are referred to as *simple* data types. Other standard basic data types are defined specifically for fieldbus applications and are referred to as *specific types*.

The constructed types specified in this PAS are strings, arrays and structures. There are no standard types defined for arrays and structures.

4.2.1.2 Basic type overview

Most basic types are defined from a set of ISO/IEC 8824 types (simple types). Some ISO/IEC 8824 types have been extended for fieldbus specific use (specific types).

Simple types are ISO/IEC 8824 universal types. They are defined in this PAS to provide them with fieldbus class identifiers.

Specific types are basic types defined specifically for use in the fieldbus environment. They are defined as simple class subtypes.

Basic types have a constant length. Two variations are defined, one for defining data types whose length is an integral number of octets, and one for defining data types whose length is bits.

NOTE Boolean, Integer, OctetString, VisibleString, and UniversalTime are defined in this PAS for the purpose of assigning fieldbus class identifiers to them. This PAS does not change their definitions as specified in ISO/IEC 8824.

4.2.1.3 Fixed length type overview

The length of Fixed length types is an integral number of octets.

4.2.1.4 Constructed type overview

4.2.1.4.1 Strings

A string is composed of an ordered set, variable in number, of homogeneously typed fixed-length elements.

4.2.1.4.2 Arrays

An array is composed of an ordered set of homogeneously typed elements. This PAS places no restriction on the data type of array elements, but it does require that each element be of the same type. Once defined, the number of elements in an array may not be changed.

4.2.1.4.3 Structures

A structure is made of an ordered set of heterogeneously typed elements called fields. Like arrays, this PAS does not restrict the data type of fields. However, the fields within a structure do not have to be of the same type.

4.2.1.4.4 Nesting level

This PAS permits arrays and structures to contain arrays and structures. It places no restriction on the number of nesting levels allowed. However, the FAL services defined to access data provide for partial access to the level negotiated by the Initiate service. The default number of levels for partial access is one.

When an array or structure contains constructed elements, access to a single element in its entirety is always provided. Access to subelements of the constructed element is also provided, but only when explicitly negotiated during AR establishment, or when explicitly preconfigured on pre-established ARs.

NOTE For example, suppose that a data type named "employee" is defined to contain the structure "employee name", and "employee name" is defined to contain "last name" and "first name". To access the "employee" structure, the FAL permits independent access to the entire structure and to the first level field "employee name". Without explicitly negotiating partial access to more than one level, independent access to "last name" or "first name" would not be possible; their values could only be accessed together as a unit through access to "employee" or "employee name".

4.2.1.5 Specification of user defined data types

Users may find it necessary to define custom data types for their own applications. User defined types are supported by this PAS as instances of data type classes.

User defined types are specified in the same manner that all FAL objects are specified. They are defined by providing values for the attributes specified for their class.

4.2.1.6 Transfer of user data

User data is transferred between applications by the FAL protocol. All encoding and decoding are performed by the FAL user.

The rules for encoding user data in FAL protocol data units is data type dependent. These rules are defined in Part 6 of IEC 61558. User-defined data types for which there are no encoding rules are transferred as a variable-length sequence of octets. The format of the data within the octet string is defined by the user.

4.2.2 Formal definition of data type objects

4.2.2.1 Data type class

4.2.2.1.1 Template

The data type class specifies the root of the data type class tree. Its parent class "top" indicates the top of the FAL class tree.

FAL ASE:	DATA TYPE ASE
CLASS:	DATA TYPE
CLASS ID:	5 (FIXED LENGTH & STRING), 6 (STRUCTURE), 12 (ARRAY)
PARENT CLASS:	TOP

ATTRIBUTES:

1	(m)	Key Attribute:	Data Type Numeric Identifier
2	(o)	Key Attribute:	Data Type Name
3	(m)	Attribute:	Format (FIXED LENGTH, STRING, STRUCTURE, ARRAY)
4	(c)	Constraint:	Format = FIXED LENGTH STRING
4.1	(m)	Attribute:	Octet Length
5	(c)	Constraint:	Format = STRUCTURE
5.1	(m)	Attribute:	Number of Fields
5.2	(m)	Attribute:	List of Fields
5.2.1	(o)	Attribute:	Field Name
5.2.2	(m)	Attribute:	Field Data Type
6	(c)	Constraint:	Format = ARRAY
6.1	(m)	Attribute:	Number of Array Elements
6.2	(m)	Attribute :	Array Element Data Type

4.2.2.1.2 Attributes

Data Type Numeric Identifier

This optional attribute identifies the numeric identifier of the related data type.

Data Type Name

This optional attribute identifies the name of the related data type.

Format

This attribute identifies the data type as a fixed-length, string, array, or data structure.

Octet Length

This attribute defines the representation of the dimensions of the associated type object. It is present when the value of the format attribute is "FIXED LENGTH" or "STRING". For FIXED LENGTH data types, it represents the length in octets. For STRING data types, it represents the length in octets for a single element of a string.

Number of Fields

This attribute defines the number of fields in a structure. It is present when the value of the format attribute is "STRUCTURE".

List of Fields

This attribute is an ordered list of fields contained in the structure. Each field is specified by its number and its type. Fields are numbered sequentially from 0 (zero) in the order in which they occur. Partial access to fields within a structure is supported by identifying the field by number. This attribute is present when the value of the format attribute is "STRUCTURE".

Field Name

This attribute specifies the name of the field. It may be present when the value of the format attribute is "STRUCTURE".

Field Data Type

This attribute specifies the data type of the field. It is present when the value of the format attribute is "STRUCTURE". This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data Type description shown below is used.

Number of Array Elements

This attribute defines the number of elements for the array type. Array elements are indexed starting at “0” through “n-1” where the size of the array is “n” elements. This attribute is present when the value of the format attribute is "ARRAY".

Array Element Data Type

This attribute specifies the data type for the elements of an array. All elements of the array have the same data type. It is present when the value of the format attribute is "ARRAY". This attribute may itself specify a constructed data type either by referencing a constructed data type definition by its numeric id, or by embedding a constructed data type definition here. When embedding a description, the Embedded Data Type description shown below is used.

Embedded Data Type Description

This attribute is used to recursively define embedded data types within a structure or array. The template below defines its contents. The attributes shown in the template are defined above in the data type class, except for the Embedded Data Type attribute, which is a recursive reference to this attribute. It is used to define nested elements.

ATTRIBUTES:

1	(m)	Attribute:	Format(FIXED LENGTH, STRING, STRUCTURE, ARRAY)
2	(c)	Constraint:	Format = FIXED LENGTH STRING
2.1	(m)	Attribute:	Data Type Numeric ID value
2.2	(m)	Attribute:	Octet Length
3	(c)	Constraint:	Format = STRUCTURE
3.1	(m)	Attribute:	Number of Fields
3.2	(m)	Attribute:	List of Fields
3.2.1	(m)	Attribute:	Embedded Data Type Description
4	(c)	Constraint:	Format = ARRAY
4.1	(m)	Attribute:	Number of Array Elements
4.2	(m)	Attribute:	Embedded Data Type Description

4.2.3 FAL defined data types

4.2.3.1 Fixed length types

4.2.3.1.1 Boolean types

4.2.3.1.1.1 Boolean

CLASS: **Data Type**

ATTRIBUTES:

1	Data Type Numeric Identifier	=	1
2	Data Type Name	=	Boolean
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This data type expresses a Boolean data type with the values TRUE and FALSE.

4.2.3.1.2 Date types

4.2.3.1.2.1 Date

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	50
2	Data Type Name	=	Date
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	7

This data type is composed of six elements of unsigned values and expresses calendar date and time. The first element is an Unsigned16 data type and gives the fraction of a minute in milliseconds. The second element is an Unsigned8 data type and gives the fraction of an hour in minutes. The third element is an Unsigned8 data type and gives the fraction of a day in hours with the most significant bit indicating Standard Time or Daylight Saving Time. The fourth element is an Unsigned8 data type. Its upper three (3) bits give the day of the week and its lower five (5) bits give the day of the month. The fifth element is an Unsigned8 data type and gives the month. The last element is Unsigned8 data type and gives the year. The values 0 ... 50 correspond to the years 2000 to 2050, the values 51 ... 99 correspond to the years 1951 to 1999.

4.2.3.1.2.2 TimeOfDay

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	12
2	Data Type Name	=	TimeOfDay
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values and expresses the time of day and the date. The first element is an Unsigned32 data type and gives the time after the midnight in milliseconds. The second element is an Unsigned16 data type and gives the date counting the days from January 1, 1984.

4.2.3.1.2.3 TimeOfDay with date indication

This data type is the same as the TimeOfDay data type defined above.

4.2.3.1.2.4 TimeOfDay without date indication

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	52
2	Data Type Name	=	TimeOfDay without date indication
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type is composed of one element of an unsigned value and expresses the time of day. The element is an Unsigned32 data type and gives the time after the midnight in milliseconds.

4.2.3.1.2.5 TimeDifference**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	13
2	Data Type Name	=	TimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4 or 6

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The optional second element is an Unsigned16 data type that provides the difference in days.

4.2.3.1.2.6 TimeDifference with date indication**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	53
2	Data Type Name	=	TimeDifference with date indication
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	6

This data type is composed of two elements of unsigned values that express the difference in time. The first element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds. The second element is an Unsigned16 data type that provides the difference in days.

4.2.3.1.2.7 TimeDifference without date indication**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	54
2	Data Type Name	=	TimeDifference without date indication
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This data type is composed of one element of an unsigned value that express the difference in time. The element is an Unsigned32 data type that provides the fractional portion of one day in milliseconds.

4.2.3.1.3 Numeric types**4.2.3.1.3.1 Floating Point types****4.2.3.1.3.1.1 Float32****CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	8
2	Data Type Name	=	Float32
3	Format	=	FIXED LENGTH

This type has a length of four octets. The format for float32 is that defined by IEC 60559 as single precision.

4.2.3.1.3.1.2 Float64

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	15
2	Data Type Name	=	Float64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type has a length of eight octets. The format for float64 is that defined by IEC 60559 as double precision.

4.2.3.1.3.2 Integer types

4.2.3.1.3.2.1 Integer8

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	2
2	Data Type Name	=	Integer8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This integer type is a two's complement binary number with a length of one octet.

4.2.3.1.3.2.2 Integer16

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	3
2	Data Type Name	=	Integer16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is a two's complement binary number with a length of two octets.

4.2.3.1.3.2.3 Integer32

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	4
2	Data Type Name	=	Integer32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is a two's complement binary number with a length of four octets.

4.2.3.1.3.2.4 Integer64**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	55
2	Data Type Name	=	Integer64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This integer type is a two's complement binary number with a length of eight octets.

4.2.3.1.3.3 Unsigned types**4.2.3.1.3.3.1 Unsigned8****CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	5
2	Data Type Name	=	Unsigned8
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	1

This type is a binary number. The most significant bit of the most significant byte is always used as the most significant bit of the binary number; no sign bit is included. This type has a length of one octet.

4.2.3.1.3.3.2 Unsigned16**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	6
2	Data Type Name	=	Unsigned16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This type is a binary number. The most significant bit of the most significant byte is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of two octets.

4.2.3.1.3.3.3 Unsigned32**CLASS:** **Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	7
2	Data Type Name	=	Unsigned32
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This type is a binary number. The most significant bit of the most significant byte is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of four octets.

4.2.3.1.3.3.4 Unsigned64

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	56
2	Data Type Name	=	Unsigned64
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This type is a binary number. The most significant bit of the most significant byte is always used as the most significant bit of the binary number; no sign bit is included. This unsigned type has a length of eight octets.

4.2.3.1.4 OctetString character types

4.2.3.1.4.1 UUID

CLASS: Data Type

ATTRIBUTES:

2	Data Type Name	=	UUID
3	Format	=	STRUCTURE
5.1	Number of Fields	=	4
5.2.1	Field Name	=	Data1
5.2.2	Field Data Type	=	Unsigned32
5.2.3	Field Name	=	Data2
5.2.4	Field Data Type	=	Unsigned16
5.2.5	Field Name	=	Data3
5.2.6	Field Data Type	=	Unsigned16
5.2.7	Field Name	=	Data4
5.2.8.1	Format	=	ARRAY
5.2.8.4.1	Number of Array Elements	=	8
5.2.8.4.2	Array Element Data Type	=	Unsigned8

This data type defines a 16 octet fixed length data type. The semantic is specified by the OSF C706 DCE RPC model.

This data type is structured as follows.

Data1

This field contains the first eight hexadecimal digits of the UUID representing time low.

Data2

This field contains the first group of four hexadecimal digits of the UUID representing time middle.

Data3

This field contains the second group of four hexadecimal digits of the UUID representing time high and version information.

NOTE For UUID version 1, the Time Value is represented by Coordinated Universal Time (UTC) as a count of 100 ns intervals since 00:00:00.00, 15 October 1582 (the date of Gregorian reform to the Christian calendar).

Data4

This field contains an array of eight elements. The first two elements contain the third group of four hexadecimal digits of the UUID representing clock sequence. The remaining six elements contain the final 12 hexadecimal digits of the UUID representing the node.

Specific values for PROFINET IO items are shown in Table 4.

Table 4 — PROFINET IO UUID

Value	Description
UUID_NIL 00000000-0000-0000-0000-000000000000	Default AR in a device
UUID_IO_ObjectInstance_XYZ DEA00000-6C97-11D1-8271-{xxxxyyyyzzzz}	Identifies a special object instance within a physical device in case there are more than one where - xxxx represents the instance or node number - yyyy identify the Device ID as a vendor specific number for the device class and - zzzz represents the Vendor ID as a central administrative number.
UUID_IO_DeviceInterface DEA00001-6C97-11D1-8271-00A02442DF7D	Identifies the IO device interface uniquely.
UUID_IO_ControllerInterface DEA00002-6C97-11D1-8271-00A02442DF7D	Identifies the IO controller interface uniquely.
UUID_IO_SupervisorInterface DEA00003-6C97-11D1-8271-00A02442DF7D	Identifies the IO supervisor interface uniquely.
UUID_IO_ParameterServerInterface DEA00004-6C97-11D1-8271-00A02442DF7D	Identifies the IO parameter server interface uniquely.

4.2.3.1.5 Time types**4.2.3.1.5.1 NetworkTime****CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	58
2	Data Type Name	=	NetworkTime
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This data type is composed of an integer value and of an unsigned value that express the network time.

The first element is an Unsigned32 data type that provides the network time in seconds since 1900.01.01 00:00:00(UTC) for network time greater/equal 1984.01.01 00:00:00 (UTC) and less than or 2036.07.02 06:28:16(UTC), or in seconds since 2036.07.02 06:28:16(UTC) for Network time greater or equal 2036.07.02 06:28:16(UTC).

The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s).

4.2.3.1.5.2 NetworkTimeDifference**CLASS:****Data Type****ATTRIBUTES:**

1	Data Type Numeric Identifier	=	59
2	Data Type Name	=	NetworkTimeDifference
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	8

This data type is composed of an integer value and of an unsigned value that express the difference in network time. The first element is an Integer32 data type that provides the network time difference in seconds. The second element is an Unsigned32 data type that provides the fractional portion of seconds in $1/2^{32}$ s.

4.2.3.2 String types

4.2.3.2.1 OctetString

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	10
2	Data Type Name	=	OctetString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

An OctetString is an ordered sequence of octets, numbered from 1 to n. For the purposes of discussion, octet 1 of the sequence is referred to as the first octet. PNO Guideline PROFINET IO-6 defines the order of transmission.

4.2.3.2.2 VisibleString

CLASS: Data Type

ATTRIBUTES:

1	Data Type Numeric Identifier	=	9
2	Data Type Name	=	VisibleString
3	Format	=	STRING
4.1	Octet Length	=	1 to n

This type is defined as the ISO/IEC 646 string type.

4.2.4 Data type ASE service specification

There are no operational services defined for the type object.

4.3 Communication model specification

4.3.1 PROFINET IO concepts

4.3.1.1 User requirements of the PROFINET IO system

The typical automation system consists of one or several programmable controllers connected with IO-systems to the machine/process. Today, an IO-system is mostly implemented as a hierarchical decentralized system or a set of single point connections with a well defined electrical interface such as 24V digital signals or 4 to 20mA analogue signals.

PROFINET IO defines a decentralized system and the connection is based on ISO/IEC 8802-3 information and telecommunication technology. The defined object model is an enhancement of the IEC61158-5 Type 3 object model.

4.3.1.2 Features of PROFINET IO

Combining the structure defined in IEC 61131-1 with PROFINET IO the following advantages are achieved:

- Changes in the plant cause only small changes in wiring
- Critical signals can be transferred over short distances to the remote field device (sensor/actuator) and to/from the programmable controller over long distances via PROFINET IO with high accuracy
- Parameterization and diagnosis can be done without additional wiring
- Commissioning of subparts without additional installations

Table 5 gives an overview of the requirements and features of a PROFINET IO system.

Table 5 — Requirements and features of PROFINET IO

Requirement	Feature of PROFINET IO
Short reaction time	Cyclic exchange of more than 1000 inputs and outputs with 32 field devices in less than 1 ms
Automation system consists of one or several programmable controller	Mono-controller or Multi-controller operation, more than one IO controller can be associated to one IO device
Simple field device	Simple protocol, low cost communication interface
Adaptation of application model of IEC61158-5/-6 Type 3	Similar application process object model with compatibility modus
Unified PROFINET protocols	Use of the same real-time transport protocol mechanism for different PROFINET application models
Excellent diagnosis	Diagnosis and alarm object model for IO devices
Smart field devices	Acyclic communication which provides a flexible and enhanced addressing scheme of data within IO devices. The possibility to transfer an alarm from the IO device to the IO controller and vice versa with an explicit acknowledgement.
Efficient communication between IO devices and between IO controller	Provider/Consumer mechanism
Synchronization of applications	Isochronous mode, jitter less than 1 µs, signaling to the application
Synchronization of clocks	High precision time synchronization
Redundancy	IO device and IO controller redundancy, support for medium redundancy
Interoperability	Precise and complete definitions including the definition of the system behavior
Modification during operation	Dynamic reconfiguration

4.3.1.3 PROFINET IO communication model

4.3.1.3.1 Associations

The PROFINET IO communication model supports three types of associations which are characterized by device types involved in the association.

NOTE Additional associations e.g. for upload or download may be possible but are not part of the PROFINET IO communication model.

The PROFINET IO communication model supports the communication of field devices with one or more controlling devices (e.g. programmable controller or distributed control system) via associations (see Figure 9). It includes cyclic data exchange of IO data and transmission of alarms, transmission of parameter data, configuration data, identification data, logbook data, diagnosis data, and record data.

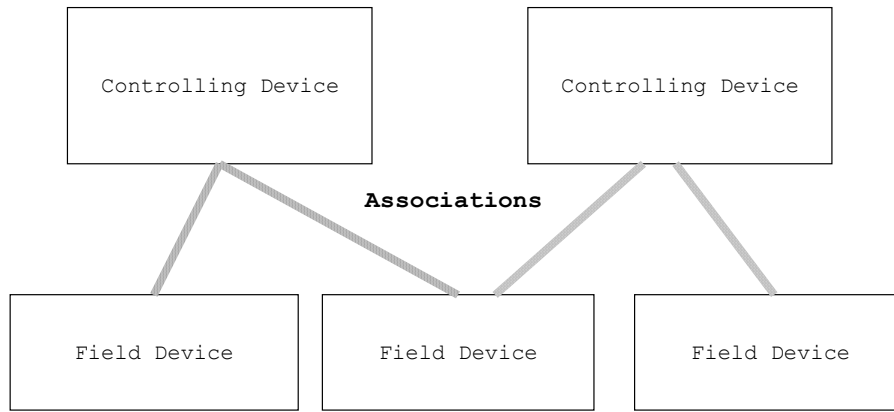


Figure 9 — Example of communication between controlling devices and field devices

Furthermore, the communication between an engineering device and several controlling or field devices is also supported (see Figure 10). The association between engineering and field devices includes transmission of parameter data, configuration data, identification data, logbook data, diagnosis data, and record data via connectionless RPC protocol.

NOTE The association between engineering and controlling device includes transmission of controller diagnosis, domains and commands for device activation via connectionless RPC protocol (for further studies).

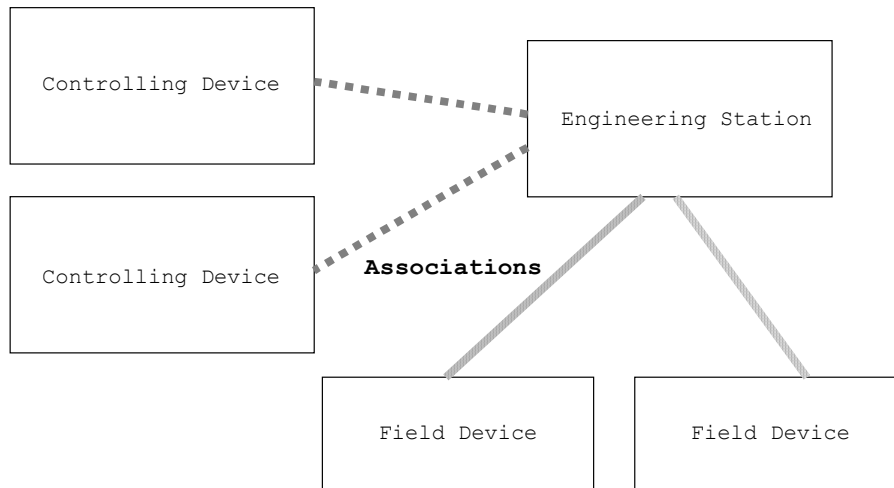


Figure 10 — Example of communication between an engineering station and several controlling and field devices

In addition to these communication models, PROFINET IO supports also the communication between a server station and field devices (see Figure 11). The association between a server station and field device includes transmission of record data via connectionless RPC protocol.

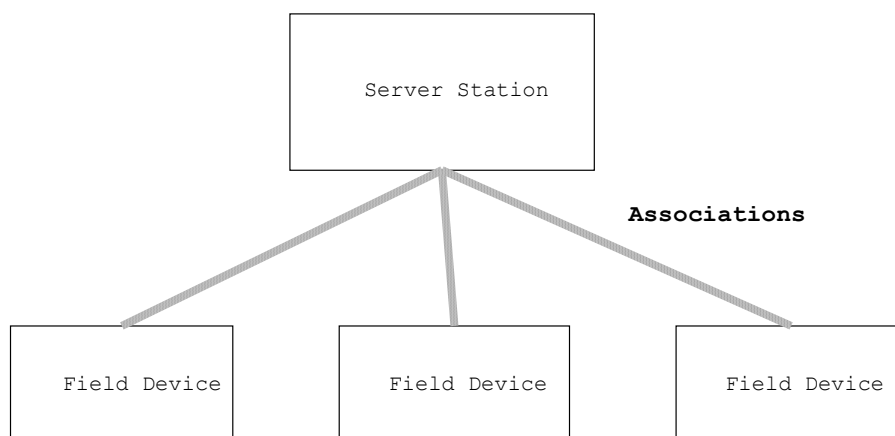


Figure 11 — Example of communication between field devices and a server station

In addition to these communication models, PROFINET IO supports also the communication between field devices (see Figure 12). This communication model includes a controlling device to setup and control the association between the field devices.

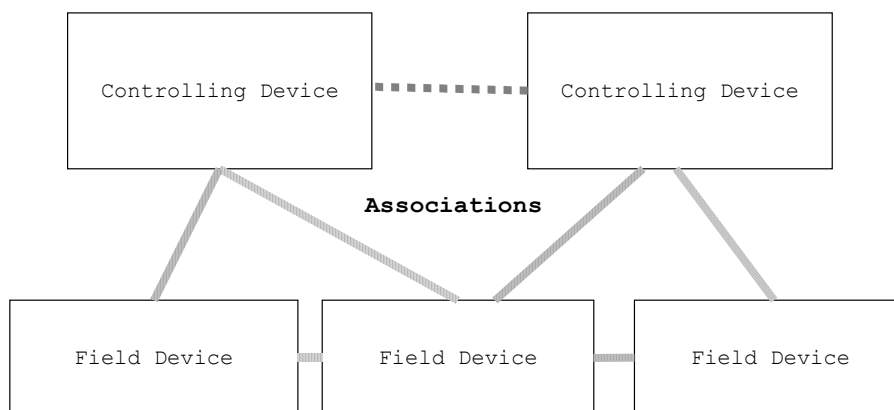


Figure 12 — Example of communication between field devices

NOTE The association between controlling devices may include e.g. the transmission of IO data, diagnosis, alarms and parameter.

4.3.1.3.2 Device types

4.3.1.3.2.1 General

A physical automation device may host one or more of the same or different PROFINET IO device types with appropriate associations.

4.3.1.3.2.2 IO controller

The IO controller is a controlling device which is associated with one or more IO devices (field devices). The IO controller performs one or more of the following functionalities:

- a) cyclic functionalities
 - exchange of IO data with related IO devices

- b) acyclic functionalities
 - read diagnosis from IO devices
 - configuration of IO devices
 - writes parameter data to IO devices (startup or application parameter)
 - treatment of configuration and diagnosis requests of an engineering device
 - establish context to IO devices by means of context management
 - acyclic access to record data of IO devices
 - treatment of alarms from IO devices
 - sending of alarms to IO devices
- c) general functionalities
 - redundancy
 - dynamic reconfiguration
 - isochronous operation
 - provider/consumer communication between IO devices and IO controllers
 - clock synchronization

IO controller conformance classes (given in 4.3.5) are defined to support common functions.

4.3.1.3.2.3 IO supervisor

The IO supervisor is an engineering device which manages provision of configuration data (parameter sets) and collection of diagnosis data for/from a IO controllers and/or IO devices.

4.3.1.3.2.4 IO parameter server

The IO parameter server is a server station which serves to store and to load application configuration data (record data objects) from IO devices (client).

4.3.1.3.2.5 IO device

4.3.1.3.2.5.1 General

The IO device is a field device and performs the following activities depending on the functionality.

The functionalities are:

- a) cyclic functionalities
 - exchange of IO data with one ore more assigned IO controller
 - exchange of IO data with related IO devices
- b) acyclic functionalities
 - provisions for diagnosis data supply
 - treatment of configuration requests of an IO controller
 - provision of acyclic access to record data for an IO controller
 - provision of alarms to the assigned IO controller
 - treatment of parameters (startup or application parameter)
 - treatment of configuration and diagnosis requests of an engineering device
 - treatment of alarms from IO controllers
 - sending of alarms to IO controllers

c) general functionalities

- provider/consumer communication between IO devices and IO controllers
- redundancy
- dynamic reconfiguration
- isochronous operation
- clock synchronization

IO device conformance classes are defined to support common functions.

In general, an IO device is hierarchical composed including the following elements:

- one or more IO device instances (see clause Annex A)
- each IO device instance includes one or more application processes referenced by its identifier (API)
- each API includes one or more slots
- each slot includes one or more subslots
- each subslots may include one or more channels

The IO device is composed of different structural units to group application objects and provides a certain level of abstraction. These structural units may reflect hardware components or virtual functional units of the field device. The aim is to provide a suitable set of address parameter. These structural units are referred to as slots, subslots, and channels which may reflect also physical units, sub-units, or a single connection point of an IO device. Figure 13 shows the structure for one arbitrary API. The structural units instance, api, slot and subslot are accessed by means of Object UUID, API Number, Slot Number and Subslot Number within the PROFINET IO address model.

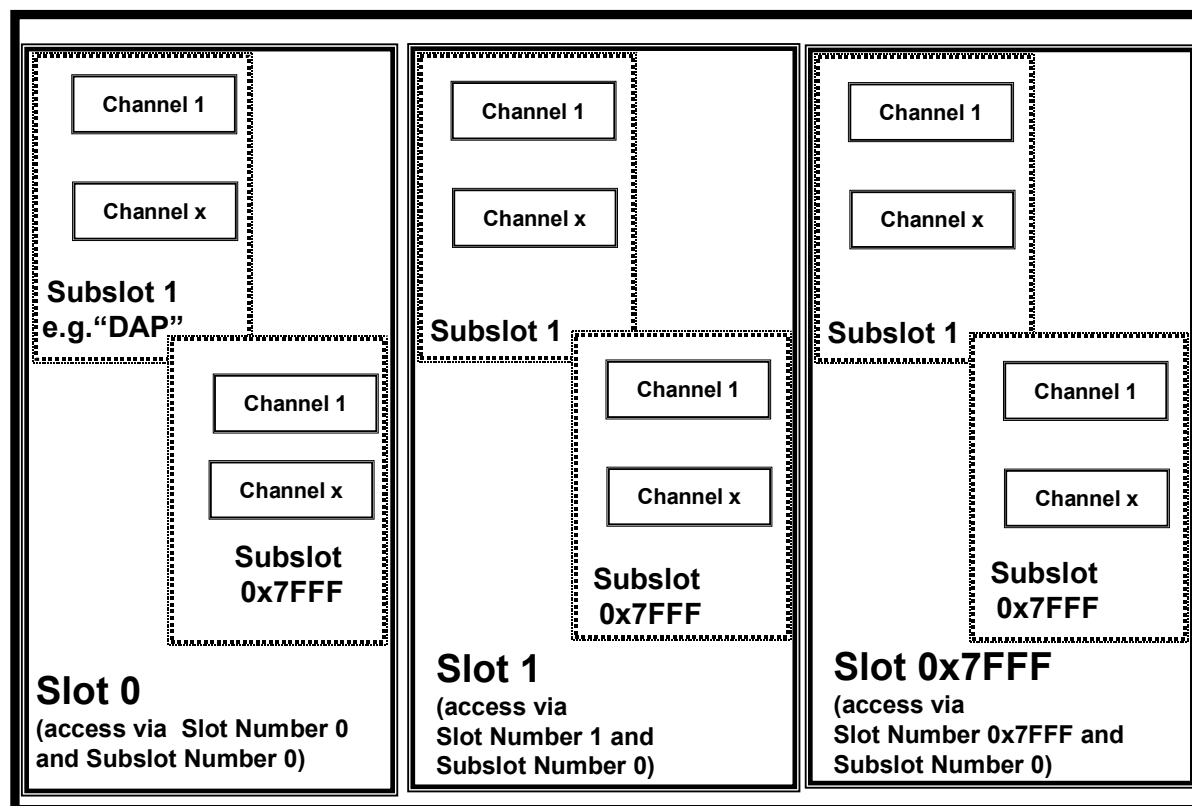


Figure 13 — Structural units of one arbitrary API of an IO device (General)

Furthermore, subslots 1 to 0x7FFF may be used in all APIs of an IO device in parallel. A particular application relationship may be established to more than one application processes. An IO device instance with more than one Application Process Identifier referred to as Multi-API-Instance.

NOTE An IO device instance supporting a particular profile API is a Multi-API-Instance because the API 0 is always mandatory.

A device manufacturer may define a particular slot/subslot combination to represent the IO device itself. It is referred to as Device Access Point (DAP). The DAP submodule shall be described within the GSDML.

Furthermore, the Subslot Number 0 shall be used to represent a special case. Subslot Number 0 in combination with an arbitrary Slot Number shall represent the dedicated module itself. This definition implies that a “real” Subslot 0 does not exist and shall not contain IO channels.

Additionally, special subslots in the range from 0x8000 to 0x8FFF are defined for API 0. An IO device shall contain at least one interface with ports. Therefore, at least one Slot (e.g. Slot Number 0), shall contain up to 16 further special Subslots from 0x8000 (Interface 1), 0x8100 (Interface 2), to 0x8F00 (Interface 16) referred to as Interface Subslots. Interface Subslots define the remote access to IP address parameters and the name of the interface (DNS name). Furthermore, each Interface Subslot shall have at least one and up to 255 assigned ports. These subslots from 0x8x01(Port 1) to 0x8xFF (Port 255) referred to as Port Subslots. The subslot defines the remote access to port specific parameter e.g. Own Port ID. Figure 14 depicts an example with several ports on the same slot belonging to one interface. All these special subslots shall only be used within API 0.

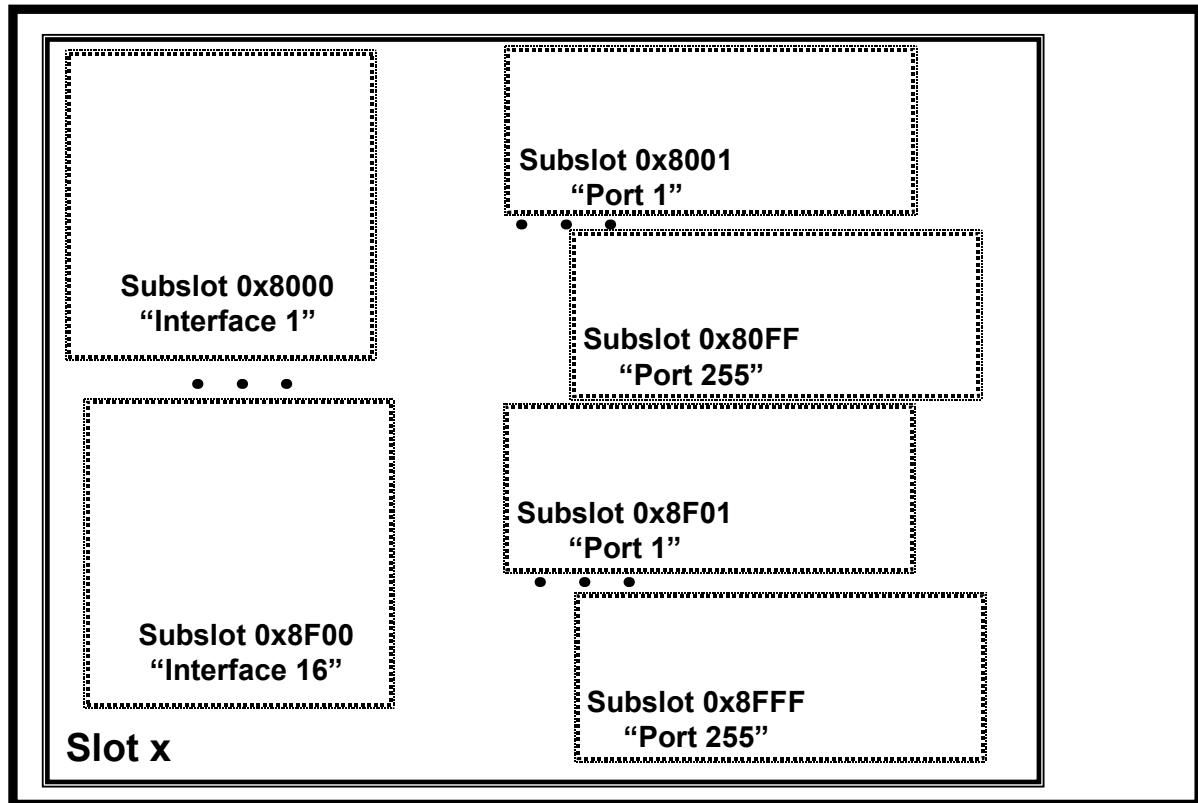


Figure 14 — Example 1 structural units for interfaces and ports within API 0

Figure 15 depicts an example with several ports on several slots belonging to one interface.

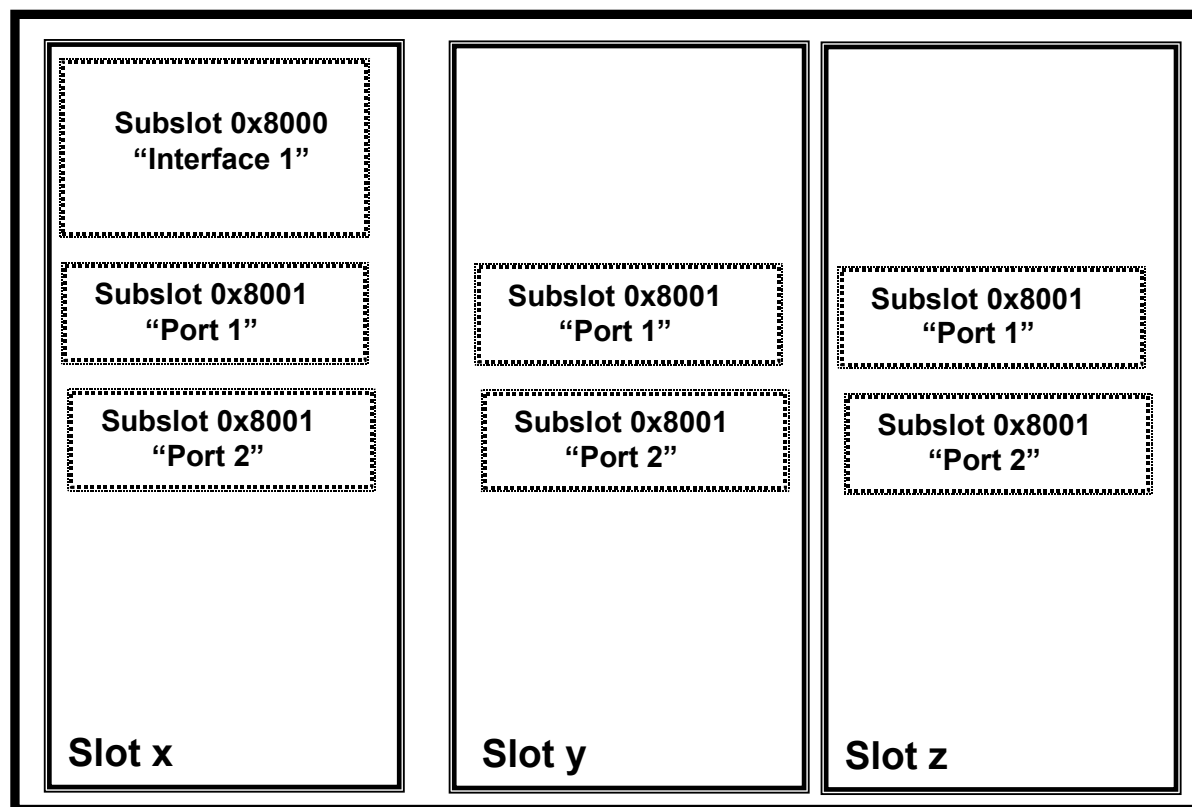


Figure 15 — Example 2 structural units for interfaces and ports within API 0

All possible configurations of an IO device shall be described in the GSDML file of the device.

4.3.1.3.2.5.2 Slot

The PROFINET IO Application Layer uses slots to reflect the structuring of functions or components (e.g. hardware modules, logical units) inside a IO device. A slot shall have one or more subslots which represent the further level to structure the data.

Each slot shall be addressed by a slot number starting with one. The last possible Slot Number is 0x7FFF. Numbering may contain gaps.

NOTE The slot model is an abstract address model via numeric references. Real hardware modules or virtual functional units may even occupy more than one slot. The device characteristics concerning use of slots are defined in a generic station description for configuration purpose.

Furthermore, each module owns configuration data. Within the connection establishment phase the requested configuration shall be compared with the real configuration data. If a slot is not occupied with a physical or virtual module then it is not part of the configuration.

In modular IO devices slots determine the actual configuration of the field device. The number of slots is fixed according to the device specification.

A compact IO device is a special case of a modular IO device with one or more fixed configurations.

4.3.1.3.2.5.3 Subslot

The PROFINET IO Application Layer uses subslots to reflect the structuring of functions or further components (e.g. hardware units, logical units) inside a slot of an IO device. A subslot may have one or more channels which represent the real structure of the Input and/or Output Data. These channels may be a further subdivision of the Input and/or Output Data object.

Each subslot will be addressed by a subslot number 1 to 0x7FFF. Furthermore, within a slot special subslots 0x8000 to 0x8FFF for interfaces and related ports may exist. The subslot

number 0 shall only be used to address a slot itself. Therefore, it shall not contain channels and IO data. On the other hand, real subslots may contain channels and IO data. The actual use of subslot number and index to address data within the device is manufacturer specific.

Within Slot 1 to 0x7FFF, a subslot provides one or several of the following objects:

- Record Data (data with open semantics, e.g. parameter, addressed by slot number (0 to 0x7FFF), subslot number (1 to 0x7FFF), index (0 to 0x7FFF))
- IO Data (addressed by slot number (0 to 0x7FFF), subslot number (1 to 0x7FFF) as an IO combined data structure, the structure is depending on generic station description, each IO data structure includes status information)
- Diagnosis (addressed by slot number (0 to 0x7FFF), subslot number (1 to 0x7FFF), index (implicitly set by the diagnosis service), contains the channel or generic diagnosis for further identification)
- Alarm (addressed by slot number (0 to 0x7FFF), subslot number (1 to 0x7FFF), alarm type).

NOTE Subslot number 0 is used as magic number to reduce the count of alarms in case of pulling out a module. In this case subslot number 0 is used as "all submodules pulled out".

Additionally, a subslot provides the above-mentioned objects also for subslot number 0x8000 to 0x8FFF.

4.3.1.3.2.5.4 Channel

The PROFINET IO Application Layer uses slots and subslots to reflect the structuring of functions and components (e.g. hardware units, logical units) inside a IO device. A slot/subslot may have one or more channels which represent the real structure of the Input and/or Output Data. These channels may be a further subdivision of the Input and/or Output Data object.

Channels are identified in diagnosis or alarm information.

4.3.1.3.3 Instance model and device addresses

The design and construction of physical automation devices is beyond the scope of this specification and therefore vendor specific. However, Annex A provides a model for physical devices to host one or more IO controller or IO devices.

4.3.1.3.4 Application process

4.3.1.3.4.1 Overview

In the PROFINET IO environment, an application may be partitioned and distributed to a number of devices on the network. Each of these partitions is referred to as an application process (AP). A device may have several APs as shown in Figure 16. In this case each individual AP is uniquely identified by an AP Identifier (API).

Each IO device shall have a Default AP with API = 0 for unrestricted.

The Default AP's shall provide device related information. Other AP's are reserved to be uniquely assigned to PROFINET IO device profiles and further standardized use.

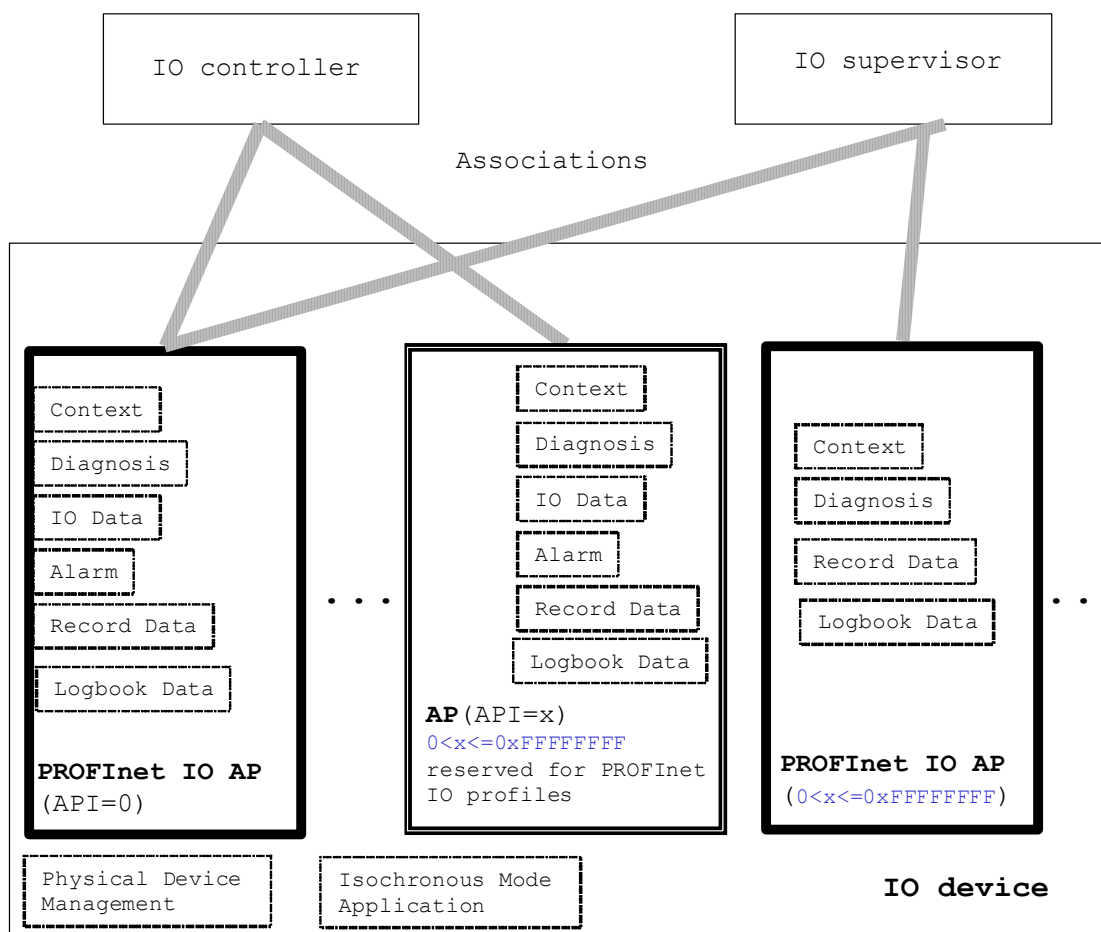


Figure 16 — Overview of application processes

An AP may be distributed to several slots and subslots (see 4.3.1.3.2.5.1). Figure 17 shows the relationship between the APs, the data elements, the slots, and subslots of an IO device. The gray boxes illustrate in Figure 17 that there exist no “real” subslot 0. These values are address parameter and reserved to address a slot itself. Subslot 0 shall also not contain e.g. IO data, as shown in Figure 17.

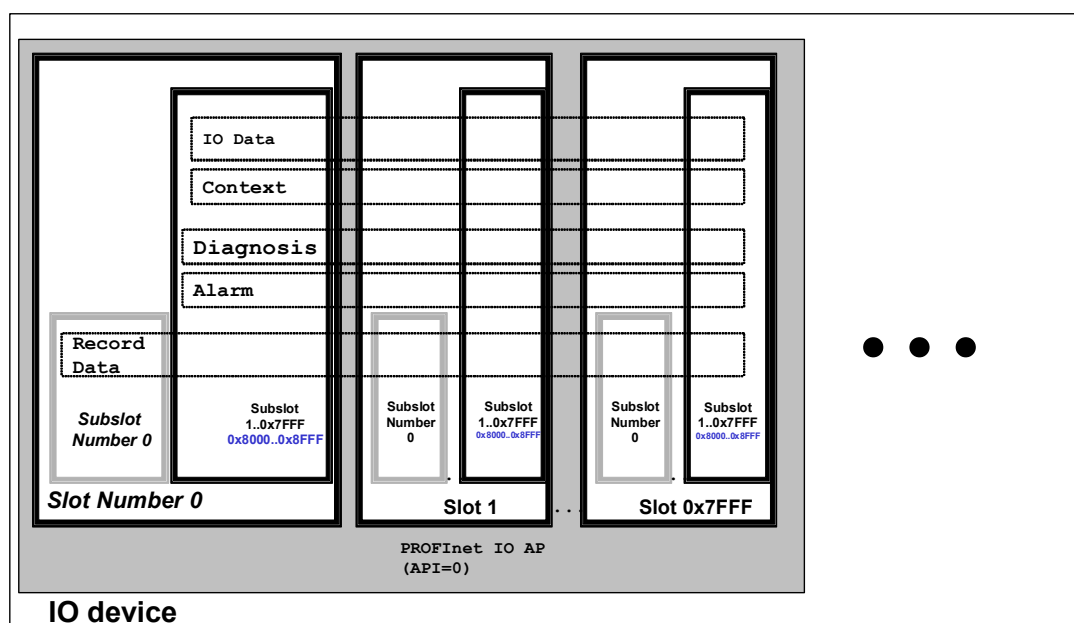


Figure 17 — IO device with APs, slots and subslots

The Subslot 0x8000 – 0x8FFF shall exist only in API 0.

NOTE The mapping of a real object to the PROFINET IO addressing model is a local matter. It means that one real object can be mapped or grouped in different APs and subslots. However, from the network perspective all address information API/slot/subslot are unique. Each API/slot/subslot can provide a different but consistent view on network visible objects. However, the IO device avoids conflicts concerning multiple use of output data referred to as overlapping outputs. An application association from a supervisor is always able to take control over output data and the controller device is informed about this by means of a special alarm.

The following rules shall be applied for the AP:

- Device instance identification data as part of the Context ASE shall always be identical regardless of the addressed AP
- API 0 addresses the PROFINET IO application process
- API 1 to 0xFFFFFFFF is reserved for IO profiles which shall be unique.

NOTE The API numbers for IO profiles are assigned by PROFIBUS International (PI).

Besides the vendor and profile specific use of the address model the following rules shall be applied for the Index:

- Index 0-0x7FFF shall be used by the device manufacturer for user specific Record Data Objects within the scope of the addressed AP. For API 0 indexes are vendor specific. For other APs further definitions specified in the related profile shall be applied.

NOTE Index 0-0xFF can be used for migration of IEC 61784-1 CP3/1 conform devices to access the same Process Data Objects with the same index.

- Index 0x8000 to 0xFFFF are reserved for protocol internal use or further extensions and shall not be used by the application regardless of the AP.

Besides the vendor and profile specific use of the address model the following rules shall be applied for the slot:

- Slot 0 to 0x7FFF is free usable within the scope of the addressed AP. For API 0 slots are vendor specific. For other APs further definitions specified in the related profile shall be applied.
- Slot 0x8000 to 0xFFFF are reserved for future use and shall not be used regardless of the AP.

Besides the vendor and profile specific use of the address model the following rules shall be applied for the Subslot:

- Subslot 0 is addressing the module within the scope of the addressed AP.
- Subslot 1 to 0x7FFF is free usable within the scope of the addressed AP. For API 0 subslots are vendor specific. For other APIs further definitions specified in the related profile shall be applied.
- For Slot 0 to 0x7FFF, Subslot 0x9000 to 0xFFFF are reserved for future use and shall not be used regardless of the AP.
- Subslot 0x8000 to 0x8FFF shall used to address the interface and port submodule. They may distributed over all slots but there shall no double use of one Subslot number across all slots. They shall only used in API 0.

Each AP shall be addressed by a APIs (x) with $0 < x < 0xFFFFFFFF$. The usage of additional APs shall be restricted to official profile definitions.

4.3.1.3.4.2 Application service element

An application service element (ASE), as defined in ISO/IEC 9545, is a set of application functions that provide a capability for the interworking of application processes for a specific purpose. ASEs provide a set of services for conveying requests and responses to and from application processes and their objects.

The PROFINET IO Application Layer offers the following ASEs:

IO Data ASE

The IO Data ASE provides a set of services to convey IO data cyclically. These data always belong to those slots/subslots that have been configured in terms of the Context ASE. IO Data Objects contain a status for transmission. Optionally the IO Data ASE offers the possibility to share the Input Data of one IO device with other IO devices. IO Data may also be read and written by a IO supervisor acyclically.

Record Data ASE

The Record Data ASE provides a set of services to convey acyclically data. The application of the IO controller or IO supervisor requests each transmission individually. The Record Data ASE can be related to all APs of an IO device.

Logbook Data ASE

The Logbook Data ASE provides a set of services to read logbook data. The application of the IO controller or IO supervisor requests each transmission individually. The Logbook Data ASE can be related to all APs of an IO device.

Diagnosis ASE

The Diagnosis ASE provides services for the IO controller or IO supervisor to read Diagnosis information from an IO device.

Alarm ASE

The Alarm ASE provides a set of services to convey alarms issued by the IO device or IO controller. The assigned IO controller or IO device acknowledges the alarm.

Context ASE

The Context ASE provides a set of services to

- convey device parameter and configuration according device description
- identify AR endpoints
- maintain AR and CR parameter (timeouts, modes, ..)
- and to establish or release an association between individual APs.

Time ASE

The Time ASE provides a set of services to synchronize the clocks of several or all devices on a PROFINET IO subnet.

Isochronous Mode Application ASE

The Isochronous Mode Application ASE provides a set of services to parameterize and synchronize isochronous application processes.

Application Relationship ASE

The Application Relationship ASE (AR ASE) provides a description model for the separate AR types. This includes their transfer characteristics as well as their current communication states.

Physical Device Management ASE

The Physical Device Management ASE provides a set of services to manage the start-up of the physical device. It includes the assignment of a Station Name and further IP address parameter.

NOTE The term physical device implies that there is only one instance of this ASE for a “real” automation device regardless of the number of instances of PROFINET IO device types within the automation device. Anyway, it will be only one instance for most devices.

4.3.1.3.4.3 Application objects

An application object (APO) is a network representation of a specific aspect of an AP. Each APO represents a set of information and processing capabilities of an AP that are accessible through services of the PROFINET IO Application Layer. APOs are used to represent these capabilities to other APs in a PROFINET IO system.

In order to permit an AP to communicate with an AP of another device, APOs have to be available. Application objects (virtual objects) represent existing process objects (real objects), that an application process has made visible and accessible to the communication (see Figure 18).

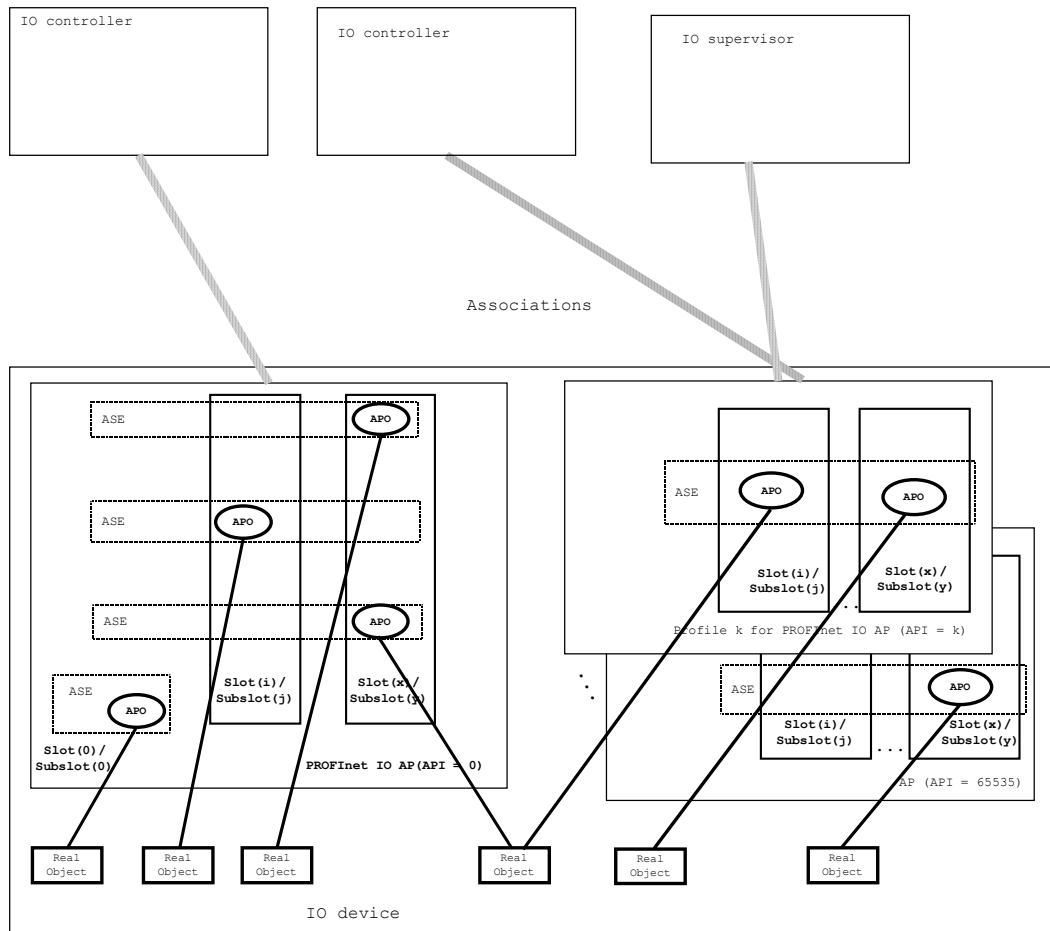


Figure 18 — Application Process with application objects (APOs)

In Figure 19, a remote AP acting as Client may access the real object by sending requests through the APO that represents the real object. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

Within the AP an APO is identified by slot, subslot and index. The address space which is defined by slot, subslot and index can be used by several APs.

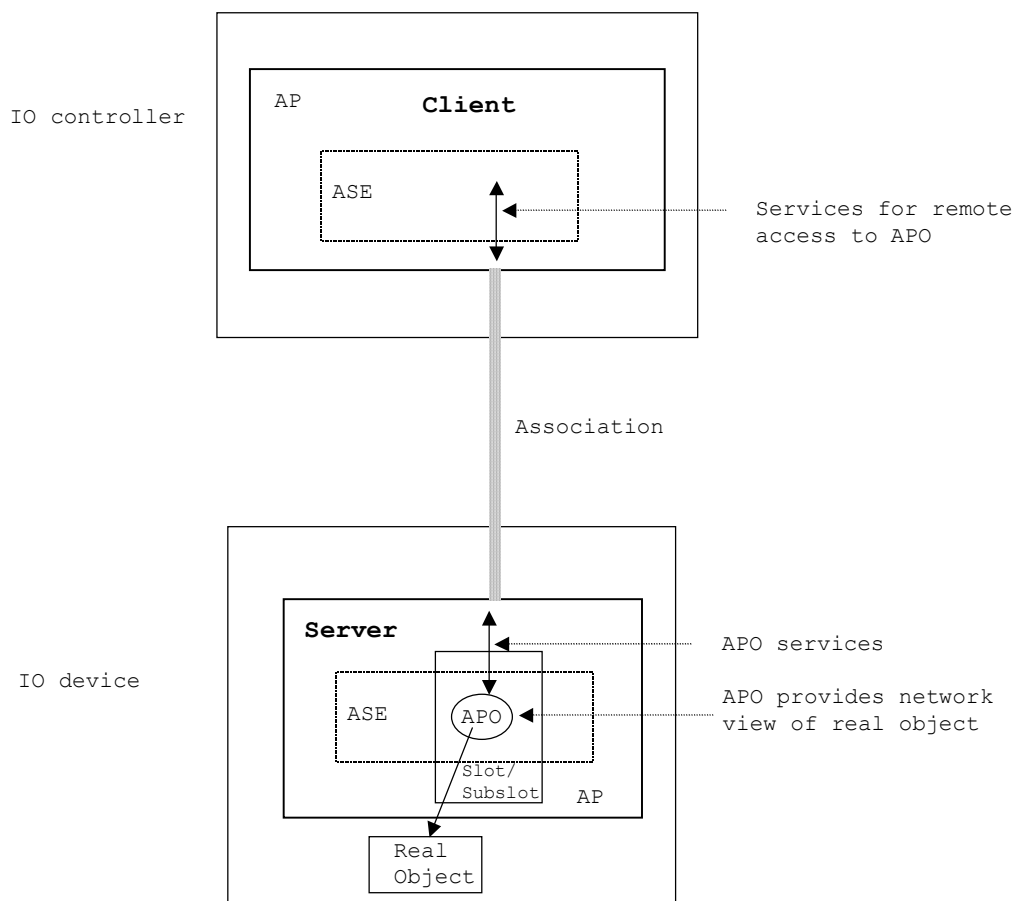


Figure 19 — Access to a remote APO

In Figure 20, a Client/Server and a Provider/Consumer association is shown. The Client may access the real object by sending requests through the APO that represents the real object. The Consumer may subscribe to the whole or to a part of the remote APO. The Provider/Consumer association is always related to the IO Data ASE. Local aspects of the AP convert between the network view (the APO) of the real object and the internal AP view of the real object.

NOTE The mapping of real data to network visible APOs is a local matter. However, the device vendor has to provide local means to avoid conflicts with overlapping outputs and writeable variables.

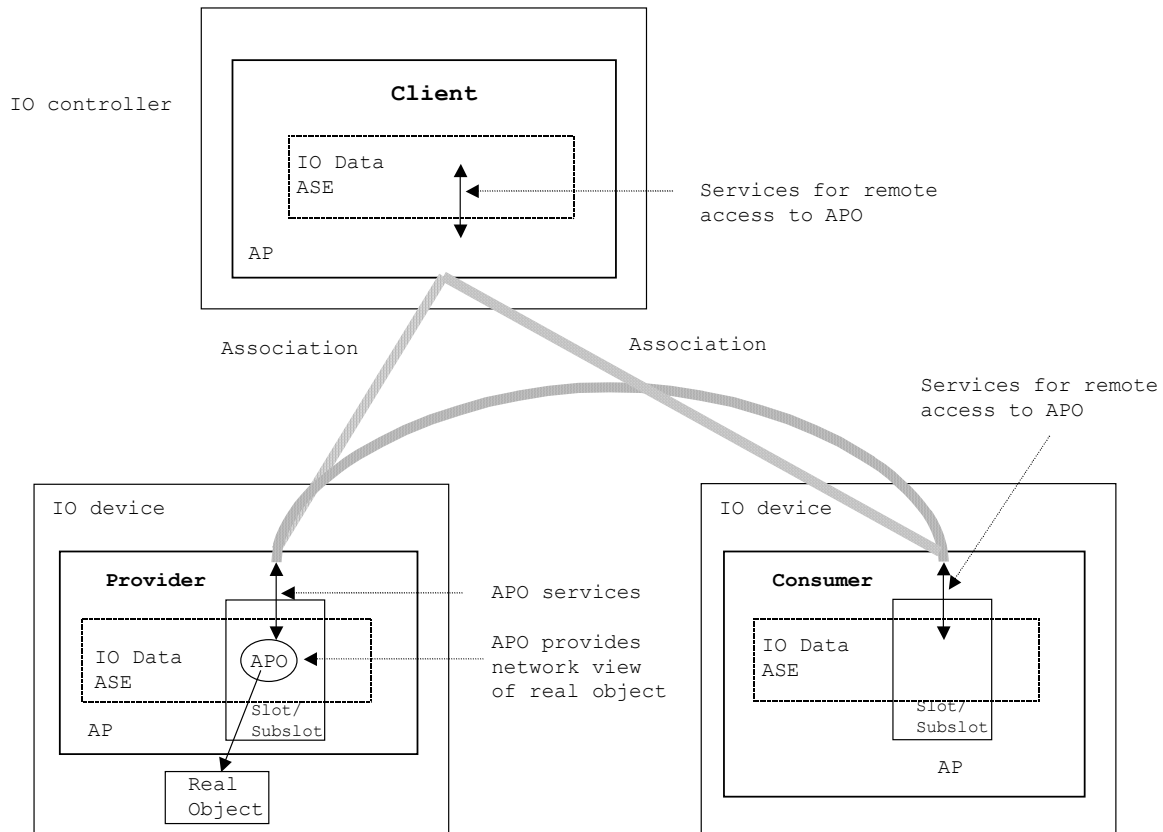


Figure 20 — Access to a remote APO for provider/consumer association

4.3.1.3.5 Application relationship

4.3.1.3.5.1 General

An Application Relationship (AR) is a co-operative relationship between two or more APs for the purpose of exchange of information and co-ordination of their joint operation (see Figure 21). This relationship is activated by the exchange of Application Protocol Data Units (APDU). PROFINET IO uses different types of ARs which distinguish in their conveyance characteristic.

4.3.1.3.5.2 Application relationship endpoint

An AP has access to the communication using application relationship endpoints (AREP, see ISO 7498). One or more AREPs are fixed and uniquely assigned to an AP. These endpoints are addressed by the AP through an identifier (AREP ID). These identifiers are device specific and not defined by the communication itself. ARs are defined as a set of co-operating AREPs. Between two APs one or more ARs may exist, each one having unique AREPs. Figure 21 shows an example of one AR with two AREPs.

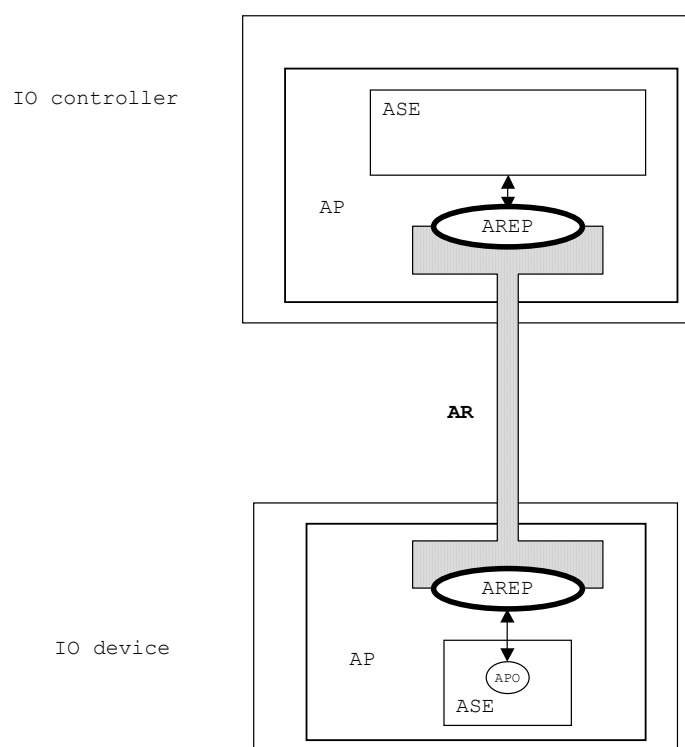


Figure 21 — Example of one AR with two AREPs

4.3.1.3.5.3 Overview of application relationships

The PROFINET IO Application Layer offers the following AR types:

NOTE The AR can be identified with a unique UUID referred to as AR UUID.

IO AR: application relationship between the application process of one IO controller and the related application process of an IO device for the following purposes:

- cyclic exchange of the Input data with the IO controller by means of Input CR (BBUU CR)
- cyclic exchange of the Output data with the IO controller by means of Output CR (BBUU CR)
- cyclic exchange (multicast) of the Input data or Output data by means of M CR (BBUU-OM CR)
- acyclic data transfer for e.g. parameterization, configuration, IO data and diagnosis (Record Data objects) by means of RDCM CR (QQCB-CO CR)
- transmission of alarm data in both directions by means of Alarm CR (QQCB-CO CR)

The IO AR shall only be used together with API 0 to 0xFFFFFFFF.

NOTE The CR specific terms e.g. BBUU are described in clause 4.3.2.8.3.5.

Supervisor AR: application relationship between:

- the application processes of one IO supervisor and one IO device,
- the application processes of one IO parameter server and one related IO device,
- the application processes of one IO controller and one IO device,

for the following purposes:

- same functions as provided by IO AR with API 0 to 0xFFFFFFFF,
- additional take over of an IO AR (controlled by supervisor)

Implicit AR: (AR UUID = UUID_NIL) application relationship between:

- the application processes of IO controller, IO devices, IO supervisor to Read ASE object values

The Implicit AR can be used together with all APIs.

4.3.2 ASEs

4.3.2.1 Record data ASE

4.3.2.1.1 Overview

In the PROFINET IO Application Layer environment, application processes contain data that remote applications are able to read and write. The Record Data ASE defines attributes of general purpose Record Data objects and provides a set of services used to read, and write their values. Furthermore, the client may cancel a requested outstanding service due to special application conditions such as dynamic reconfiguration. A Record Data ASE is specific for an AP and each AP shall contain one Record Data ASE. Record Data objects within an AP are addressed by Slot Numbers, Subslot Numbers, and Index. The Record Data object can be read and written partially or entirely. The local connection between a Record Data object and a real world or physical object is not in the scope of this definition.

NOTE A Record Data object may be connected to one real world object for read and write, or, it can be assigned to one real object for reading and for another one for writing the value.

The IO controller and IO supervisor are able to read or write the value of the Record Data object within the AP of an IO device. Furthermore, an IO device is able to read or write the value of a Record Data object within the AP of an IO parameter server. The IO supervisor is able to read or write the value of a Record Data object within the AP of an IO controller.

The client application process has to use the Context ASE to establish an association to get access to the Record Data objects of a server. Read access is also possible without a connection via the implicit AR.

The access to Record Data objects is performed according to the Client/Server access model. The Client/Server model is characterized by a Client application sending a read or write request to a Server application that responds accordingly.

The formal model of the Record Data ASE is presented by the Record Data class specification, containing a description of its attributes, services as well as invocations, followed by a detailed service specification.

Special Record Data objects shall be reserved for identification and maintenance functions. The reserved indices are 0xAFF0 to 0xAFFF.

NOTE The content of these Record Data Objects is specified in the document PROFIBUS 3.502. The PROFIBUS IM (see PROFIBUS 3.502) functions are mapped as follows:

- 0xAFF0 IM0,
- 0xAFF1 IM1,
- 0xAFF2 IM2,
- ...
- 0xAFFF IM15

Furthermore, additional reservations are defined for further Profile definitions.

4.3.2.1.2 Record data class specification

4.3.2.1.2.1 General

The Record Data ASE defines one Record Data object type:

4.3.2.1.2.2 Record data class specification

4.3.2.1.2.2.1 Template

A Record Data object may be related to real object as shown in Figure 22.

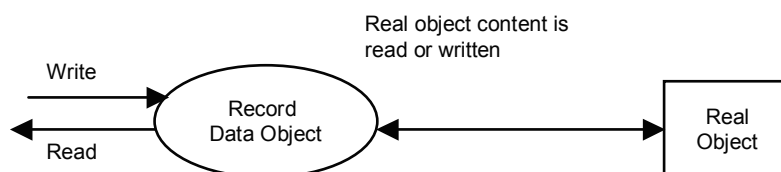


Figure 22 — Relation of a record data object to one real object

A Record Data object may be related to two real objects as shown in Figure 23.

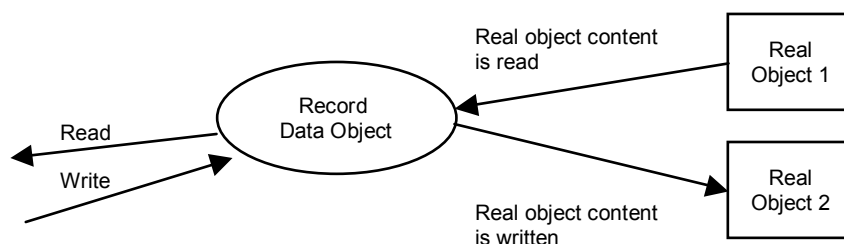


Figure 23 — Relation of a record data object to two real objects

A Record Data object is described by the following template:

DP ASE: Record Data ASE

CLASS: Record Data

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Key Attribute: Identifier
2	(m)	Attribute: Read Record Data Description
3	(m)	Attribute: Read Partial Access
4	(m)	Attribute: Write Record Data Description
5	(m)	Attribute: Write Partial Access

SERVICES:

1	(o)	OpsService: Read
2	(o)	OpsService: Write

4.3.2.1.2.2.2 Attributes

Identifier

This key attribute is composed of API, Slot Number, Subslot Number, and Index to define to which Application Process Identifier, structural elements slot (module), subslot (submodule), and to which index the Record Data object belongs. This Identifier is unique in the IO device and shall not be used by another object.

Attribute Type: Unsigned16, Unsigned16, Unsigned16, Unsigned16

Allowed values for API: 0, all others are reserved for Profile definitions

Allowed values for Slot Number and Subslot Number: 0 to 0x7FFF, additionally Subslot Number 0x8000 to 0x8FFF are allowed

Allowed values for Index: 0 to 0x7FFF, 0xAFF0 to 0xAFFF for I&M Record Data objects, reserved for profile definitions are: 0xB000-0xBFFF, 0xD000-0xDFFF, 0xEC00-0xEFFF, 0xF400-0xF7FF, 0xFC00-0xFFFF.

Other values are reserved for further use as defined in 4.3.1.3.4.

Read Record Data Description

This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in 4.2.1 is used.

Read Partial Access

This attribute defines whether a partial read access to the Record Data object is supported or not. This attribute controls the behavior of the Read service. Partial access equal TRUE means that the content of the Record Data object may be read from the first octet up to the octet defined by the length given in the service request even if the length is shorter than the length of the Record Data object. The entire object will be read if the length given in the service is greater or equal than the length of the object in any case. Partial access equal FALSE means that the content of the Record Data object can only be read entirely. That implies that the length given in the service shall be equal or greater to the length of the object. Otherwise it shall return a negative response with the error code “Invalid Type”.

Attribute Type: Boolean

Write Record Data Description

This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in 4.2.1 is used.

Write Partial Access

This attribute defines whether a partial write access to the Record Data object is supported or not. This attribute controls the behavior of the Write service. Partial access equal TRUE means that the content of the Record Data object shall be written from the first octet up to the octet defined by the length given in the service request even if the length is shorter than the length of the Record Data object. The entire object will be written if the length given in the service is equal to the length of the object in any case. Partial access equal FALSE means that the content of the Record Data object can only be written entirely. That implies that the length given in the service shall be equal to the length of the object. Otherwise it shall return a negative response with the error code “Write Length Error”. If the length in the Write service exceeds the length of the Record Data object a negative response with the error code “Write Length Error” shall be returned.

Attribute Type: Boolean

4.3.2.1.2.3 Invocation of the record data object

For the invocation of the Record Data object the following rules apply:

- Record Data objects shall not exceed the total length of 232-65 octets. The attribute Record Data description has to be set accordingly.
- At least one service shall be allowed for the access of the object.
- At least one AR shall have access to the object.
- The access rights have to be set accordingly to the allowed service(s).

4.3.2.1.3 Record data service specification

4.3.2.1.3.1 Read

This confirmed service may be used to read the value of a Record Data object. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 6 shows the parameters of the service.

The attribute Read Partial Access controls the behavior of the response in case the requested length is shorter than the length of the Record Data object (see attribute description).

Table 6 — Read

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)
Length			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Record Data object in a specific slot (typically a module).

Subslot Number

The parameter Subslot Number is used in the destination device for addressing the desired Record Data object in a specific subslot (typically a submodule).

Index

The parameter Index is used in the destination device for addressing the desired Record Data object.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of a Record Data object that has to be read. The allowed length is from 0 to $2^{32} - 65$.

Result(+)

This parameter indicates that the service request succeeded.

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 can be used by profile specifications to transmit specific messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 can be used by device vendors to transmit specific additional messages.

Data

The parameter Data contains the value of the object which has been read and consists of the number of octets indicated in the Length of the response. This parameter has to be composed of the data types defined in 4.2.2.1.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 can be used by device vendors to transmit specific error messages.

4.3.2.1.3.2 Write

This confirmed service may be used to write the value of a Record Data object. This service shall only be used in conjunction with the IO AR or Supervisor AR. Table 7 shows the parameters of the service.

The attribute Write Partial Access controls the behavior of the response in case the requested length is shorter than the length of the Record Data object (see attribute description).

The service parameter "Multiple" may be used to convey multiple Record Data objects within one APDU. The server shall mirror the value of "Multiple" within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE) in one APDU.

Table 7 — Write

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Data	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Record Data object in a specific slot (typically a module).

Subslot Number

The parameter Subslot Number is used in the destination device for addressing the desired Record Data object in a specific sub-element (typically a submodule).

Index

The parameter Index is used in the destination device for addressing the desired Record Data object.

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service

request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of a Record Data object which has to be written. The allowed length is from 0 to $2^{32}-65$.

Data

The parameter Data contains the value of the Record Data object which has to be written and consists of the number of octets indicated in the Length of the request. This parameter has to be composed of the data types defined in 4.2.2.1.

Result(+)

This parameter indicates that the service request succeeded.

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 can be used by profile specifications to transmit specific response messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 can be used by device vendors to transmit specific response messages.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: write error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, write constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid AR.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 can be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 can be used by device vendors to transmit specific error messages.

4.3.2.2 IO Data ASE

4.3.2.2.1 Overview

In the PROFINET IO Application Layer environment, each application process of an IO device shall contain exactly one IO Data object for each application process instance. It is structured by means of slots and subslots. Subslots represent IO data that shall be cyclically conveyed over the network. Therefore the IO Data ASE defines attributes of the IO Data objects and provides a set of services used to get and set the buffer to transport their values. The IO Data objects are composed of an Input Data object and an Output Data object.

An Input Data object or an Output Data object shall be partitioned in Input Data or Output Data Elements which represent the IO Data of the slot/subslot.

Additional services are provided to read acyclically the values of the IO Data objects and to indicate new values for the Input Data object and the Output Data object. Furthermore services are provided to indicate the presence and to get the content of new consumed Input Data.

The IO Data objects are implicitly addressed through the related services. The granularity of input or output data in a Server/Provider is according to the correspondent configuration attributes.

The I/O Data ASE uses the Client/Server and Provider/Consumer access model. The Client/Server model is characterized by a Client application conveying the value for the Output Data object to the Server's buffer. The Server application fetches this value by a Get Output service. The receipt of a new value is indicated by the New Output service. The Server application conveys the value of its Input Data object via the Set Input service to the Server's buffer as soon as one of the real object values changes. The Client application gets this value by the Get Input service. The transmission of the Server buffer to the Client buffer and vice versa is done asynchronously to the Get Output and Set Input services.

The formal model of the IO Data ASE is described next, followed by a description of its services. Furthermore, the IO Data ASE represents the real input and output structure of an IO device. The relations between real IO data structure and conveyance of IO data via ARs are described within the Context ASE (see 4.3.2.5.4).

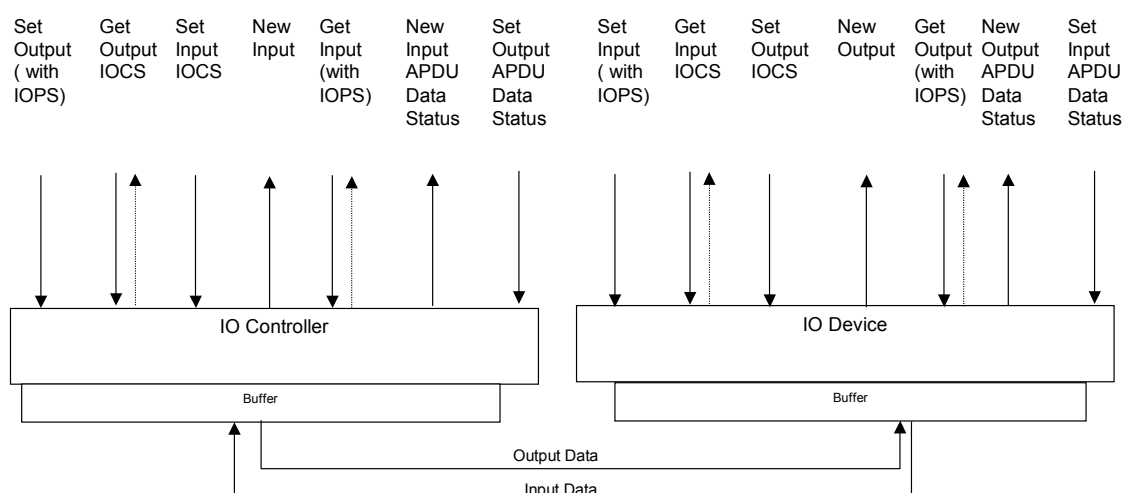


Figure 24 — Overview IO ASE service interactions

Figure 24 shows the IO service interactions for the IO controller and IO device. The services interact with local buffers maintained by the application layer protocol. This protocol provides mechanisms for cyclic exchange of buffers between the devices. Local services provide a means to get or set the values of IO data together with their status values. The status values are referred to as IOPS and IOCS. The IOPS (Input Output Provider Status) characterizes the status of the data source. It belongs to Output Data for an IO controller and to Input Data for an IO device. The IOCS (Input Output Consumer Status) characterizes the feedback of the data sink. It belongs to Input Data for an IO controller and to Output Data for an IO device.

Furthermore, services are provided to set and indicate a change of the application data status (APDU Data Status).

NOTE If a submodule does not contain any input or output data it is considered as an Input Data item with the length of Input Data attribute set to zero.

The transition of IOPS/IOCS from GOOD to BAD may be done without further notification (e.g. alarms). The transition of IOPS/IOCS from BAD to GOOD shall be notified to and acknowledged from the IO Controller by means of an alarm after a change of IOXS from Bad->Good in data exchange state..

4.3.2.2.2 IO data class specification

4.3.2.2.2.1 Input data class specification

4.3.2.2.2.1.1 Template

The Input Data object is described by the following template:

DP ASE:	IO Data ASE
CLASS:	Input Data
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of APs
2.1	(m) Attribute: API
2.2	(m) Attribute: List of Input Data Elements
2.2.1	(m) Attribute: Slot Number
2.2.2	(m) Attribute: List of Subslots
2.2.2.1	(m) Attribute: Subslot Number
2.2.2.2	(m) Attribute: Input Data Object Format
2.2.2.2.1	(m) Attribute: Length Input Data
2.2.2.2.2	(m) Attribute: Input Data Structure
2.2.2.3	(m) Attribute: Length IOPS
2.2.2.4	(m) Attribute: IOPS
2.2.2.5	(m) Attribute: Length IOCS
2.2.2.6	(m) Attribute: IOCS
SERVICES:	
1	(m) OpsService: Set Input
2	(m) OpsService: Set Input IOCS
3	(m) OpsService: Get Input
4	(m) OpsService: Get Input IOCS
5	(o) OpsService: New Input
6	(m) OpsService: Set Input APDU Data Status
7	(m) OpsService: New Input APDU Data Status
8	(m) OpsService: Read Input Data

4.3.2.2.2.1.2 Attributes

Implicit

The attribute Implicit indicates that the Input Data object is implicitly addressed by the services.

List of APs

One AP is composed of the following list elements:

API

This attribute defines to which application process the Input Data Elements belongs.

Attribute Type: Unsigned32

Allowed values: 0 – default PROFINET IO AP, The values 1 to 0xFFFFFFFF are reserved for future use in Profile guidelines.

List of Input Data Elements

One Input Data Element is composed of the following list elements:

Slot Number

This attribute defines to which module the Input Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Subslots

This attribute consists of the following attributes.

Subslot Number

This attribute defines to which subslot the Input Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE Subslot Number 0 is used to address a certain module within the IO device. It is never a "real" submodule and it does not contain Input Data Elements.

Input Data Object Format

This attribute consists of the following attributes.

Length Input Data

The attribute Length Input Data defines the number of octets of the Input Data Element without IOPS.

Attribute Type: Unsigned16

Allowed values: 0 to 1439

Input Data Structure

This attribute shall contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in 4.2.2.1 is used.

The Input Data shall be transmitted consistent from the sending buffer to the receiving buffer completely. However, the application may get or set only parts of a single data object including the IOPS.

This field is part of the GSDML.

Furthermore, if the Input Data are the source of a Multicast Provider CR then the associated Output Data of the Multicast Consumer CR shall have exactly the same data structure.

Length IOPS

The attribute Length IOPS defines the number of octets of the provider status.

Attribute Type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOPS

The attribute defines the provider status for the Input Data Object Element. It defines the status of the Input Data Object Element which belongs to the subslot of the IO device. It provides a means to identify the first instance which detected the invalid content of the Input Data Object Element by local means. The detecting instance shall set the IOPS accordingly and immediately. Possible instances are the submodule, the module, the IO device, and the IO controller (locally, e.g. watchdog timeout). Only valid data with IOPS==GOOD shall be processed by the application. In all other cases the application shall use default values (e.g. zero, last valid value, substitute value) according to its configuration and parameterization.

Attribute Type: Unsigned8

Allowed values:

GOOD – content of Input Data Object Element is valid,

BAD_BY_SUBSLOT – content of Input Data Object Element is invalid, it was detected by the submodule,

BAD_BY_SLOT – content of Input Data Object Element is invalid, it was detected by the module,

BAD_BY_DEVICE – content of Input Data Object Element is invalid, it was detected by the IO device,

Special case: BAD_BY_CONTROLLER – content of Input Data Object Element is invalid, only locally detected by the IO controller itself e.g. through a communication timeout. This value shall only be used with the service parameter IOPS together with the service Get Input at the IO controller. It shall never be applied with the Set Input service at the IO device.

Length IOCS

The attribute Length IOCS defines the number of octets of the consumer status.

Attribute Type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOCS

The attribute defines the consumer status for the Input Data object element as an application feedback of the IO controller. It is not only a mean to detected communication problems. Moreover it is a mean to inform the IO device that the control application or parts of the control application are not processing the values. Therefore the application process shall set the IOCS to BAD_BY_CONTROLLER if the Input Data Object Element could not be processed e.g. if the application program stopped. The possible action of the IO device is beyond the scope of this service definition. The attribute IOCS shall only contain a valid value if the Input Data Object Element is part of an established Application Relationship (AR). In all other cases the value of this attribute has no meaning.

Attribute Type: Unsigned8

Allowed values:

GOOD – the Input Data Object Element could be successfully processed by the application process of the IO controller,

BAD_BY_CONTROLLER – the Input Data Object Element could not be successfully processed by the application of the IO controller e.g. in case of an error or as cause of the operation state („stop“) of the IO controller

Special case: BAD_BY_DEVICE – the IO device has locally detected problems to convey data. It shall never be applied with the Set Input IOCS service at the IO controller.

4.3.2.2.1.3 Invocation of the input data object

For the invocation of the Input Data object the following rules apply:

- Input Data object elements data shall not exceed the total length of 1439 octets. The attributes List of Input Data Elements, Input Data Object Format respectively Format and Length have to be set accordingly.
- The relation of an Input Data object element to a Input CR, Output CR (only IOCS), M CR will be handled in the Context ASE.
- The sum of all Input Data object elements related to one Input CR or M CR including IOPS and IOCS of the related Output Data object elements shall not exceed the total length of 1440 octets.
- One Input Data object shall be invoked in an application process of an IO device.

4.3.2.2.2.2 Output data class specification

4.3.2.2.2.2.1 Template

The Output Data object is described by the following template:

DP ASE:	IO Data ASE
CLASS:	Output Data
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of APs
2.1	(m) Attribute: API
2.2	(m) Attribute: List of Output Data Elements
2.2.1	(m) Attribute: Slot Number
2.2.2	(m) Attribute: List of Subslots
2.2.2.1	(m) Attribute: Subslot Number
2.2.2.2	(m) Attribute: Output Data Object Format
2.2.2.2.1	(m) Attribute: Length Output Data
2.2.2.2.2	(m) Attribute: Output Data Structure
2.2.2.3	(m) Attribute: Length IOPS
2.2.2.4	(m) Attribute: IOPS
2.2.2.5	(m) Attribute: Length IOCS
2.2.2.6	(m) Attribute: IOCS
2.2.2.7	(m) Attribute: Safe State Behavior Changeable
2.2.2.8	(m) Attribute: Substitute Mode
2.2.2.9	(m) Attribute: Output Substitute Data
SERVICES:	
1	(m) OpsService: Set Output
2	(m) OpsService: Set Output IOCS
3	(m) OpsService: Get Output
4	(m) OpsService: Get Output IOCS
5	(o) OpsService: New Output
6	(m) OpsService: Set Output APDU Data Status
7	(m) OpsService: New Output APDU Data Status
8	(m) OpsService: Read Output Data
9	(m) OpsService: Write Output Substitute Data

4.3.2.2.2.2.2 Attributes

Implicit

The attribute Implicit indicates that the Output Data object is implicitly addressed by the services.

List of APs

One AP is composed of the following list elements:

API

This attribute defines to which application process the Output Data Elements belongs.

Attribute Type: Unsigned32

Allowed values: 0 – default PROFINET IO AP, The values 1 to 0xFFFFFFFF are reserved for future use in Profile guidelines .

List of Output Data Elements

One Output Data Element is composed of the following list elements:

Slot Number

This attribute defines to which module the Output Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Subslots

This attribute consists of the following attributes.

Subslot Number

This attribute defines to which subslot the Output Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

NOTE Subslot Number 0 is a special meaning to address a certain module within the IO device. It is never a "real" submodule and it does not contain Output Data Elements.

Output Data Object Format

This attribute consists of the following attributes.

Length Output Data

The attribute Length Output Data defines the number of octets of the Output Data Element without IOPS.

Attribute Type: Unsigned16

Allowed values: 1 to 1439

Output Data Structure

This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description. For data description the template given in 4.2.2.1 is used.

The Output Data shall be transmitted consistent from the sending buffer to the receiving buffer completely. However, the application may get or set only parts of a single data object including the IOPS.

This field is part of the GSDML.

Furthermore, if the Output Data are the sink of a Multicast Consumer CR then the associated Input Data of the Multicast Provider CR shall have exactly the same data structure.

Length IOPS

The attribute Length IOPS defines the number of octets of the provider status.

Attribute Type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOPS

The attribute defines the provider status for the Output Data Object Element. It defines the status of the Output Data Object Element which belongs to the submodule of the IO device. Moreover it is a means to inform the IO device that the control application or parts of the control application are not processing. The application process shall set the IOPS to BAD_BY_CONTROLLER if the Output Data Object Element could not set e.g. if the application program stopped. The IO device shall use the default values (zero, last valid value, substitute value) for process output in such a case. The attribute IOPS shall only contain a valid value if the Output Data Object Element is part of an established Application Relationship (AR). In all other cases the value of this attribute has no meaning.

Attribute Type: Unsigned8

Allowed values:

GOOD – the Output Data Object Element was successfully processed by the application process of the IO controller,

BAD_BY_CONTROLLER – the Output Data Object Element could not be successfully processed by the application of the IO controller e.g. in case of an error or as cause of the operation state („stop“) of the IO controller

Special case: BAD_BY_DEVICE – the IO device has locally detected problems to convey data. It shall never be applied with the Set Input service at the IO controller.

Length IOCS

The attribute Length IOCS defines the number of octets of the consumer status.

Attribute Type: Unsigned8

Allowed values: 1, (2 to 255 are reserved for future use)

IOCS

The attribute defines the consumer status for the Output Data object element as an application feedback of the IO device. It is not only a means to detect communication problems. Moreover it is a means to inform the IO controller that the device application could not set the values to the “real” process for local reasons. The detecting instance shall set the IOCS accordingly and immediately. Possible instances are the submodule, the module, the IO device, and the IO controller (locally, e.g. watchdog timeout). The possible action of the IO controller is beyond the scope of this service definition. The attribute IOCS shall only contain a valid value if the Output Data Object Element is part of an established Application Relationship (AR). In all other cases the value of this attribute has no meaning.

Attribute Type: Unsigned8

Allowed values:

GOOD – the Output Data Object Element was successfully processed,

BAD_BY_SUBSLOT – the Output Data Object Element was not successfully processed, it was detected by the submodule,

BAD_BY_SLOT – the Output Data Object Element was not successfully processed, it was detected by the module,

BAD_BY_DEVICE – the Output Data Object Element was not successfully processed, it was detected by the IO device,

Special case: BAD_BY_CONTROLLER – only locally detected by the IO controller itself e.g. through a communication timeout. This value shall only be used with the service parameter IOCS together with the service Get Output IOCS at the IO controller. It shall never be applied with the Set Output IOCS service at the IO device.

Safe State Behavior Changeable

The Safe State Behavior Changeable attribute defines if this behavior is adjustable.

Attribute Type: Boolean

Substitute Mode

The Substitute Mode attribute defines the handling of the output data subslot in case of a failure. This behavior shall be applied if the IOPS value contains other values than GOOD.

Attribute Type: Unsigned16

Allowed Values:

ZERO – is a submodule specific function to set all output values of the Output Data element to zero or inactive,

LAST_VALUE – all output values of the Output Data element shall be set to the last valid (IOPS==GOOD) values of the Get Output confirmation service primitive,

REPLACEMENT_VALUE – all output values of the Output Data element shall be set to the value of the attribute Safe State Value

PROFILE_SPECIFIC1 – PROFILE_SPECIFIC255 reserved for profile specification

Output Substitute Data

The Output Substitute Data attribute defines the replacement value of the output data for the subslot in case of failure.

The attribute type shall be the same as defined in the Output Data Structure attribute.

4.3.2.2.2.3 Invocation of the output data object

For the invocation of the Output Data object the following rules apply:

- Output Data object elements (subslot) data shall not exceed the total length of 1439 octets. The attributes List of Output Data Elements, Output Data Object Format respectively Format and Length have to be set accordingly.
- The relation of an Output Data object element to a Output CR, Input CR, M CR will be handled in the Context ASE.
- The sum of all Output Data object elements including IOPS and IOCS of the related Input Data object elements related to one CR shall not exceed the total length of 1440 octets.
- One Output Data object shall be invoked in an application process of an IO device.

4.3.2.2.3 IO data service specification

4.3.2.2.3.1 Set Input

The Set Input service sets new input data of one slot/subslot together with its IOPS for next transmission for all related application relationships. The IOPS shall be at any time consistent with delivered Input Data values. The value of IOPS shall only be “GOOD” for valid input data of the whole subslot. In all other cases the value shall be set to “BAD_BY_xxx” and include the detecting instance e.g. subslot, slot or IO Device. This service is used by the application process of the IO device to update the addressed part of the protocol internal cyclic buffer. The actual transmission of the buffer is time triggered.

The Set Input service may set the IOPS only. It is the case if the slot/subslot does not contain Input Data.

NOTE The local implementation of the service interface may support access variants with other than subslot granularity e.g. for more than one subslot or only parts (variables) of the Input Data object element. However, the Input Data object element is always consistent with its IOPS.

This service shall only be used in conjunction with the IO AR and Supervisor AR.

According to the cyclic buffer to buffer transportation characteristics the following behavior is possible:

- The Set Input requests are issued faster than the contents of the buffer is transported over the network. In this case only the values provided with the latest Set Input service are conveyed over the network.
- The Set Input requests are issued slower than the contents of the buffer is transported over the network. In this case the values provided with each Set Input service are conveyed more than once over the network.
- If the Set Input requests are issued synchronous with the transport of the contents of the buffers the values provided with each Set Input service are conveyed once over the network.

Table 8 shows the parameters of the service.

Table 8 — Set Input

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOPS	M	
Input Data	U	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

NOTE The Set Input service sets the input data for all established ARs that belong to the submodule. In this case multiple calls for shared inputs are not necessary. Therefore, the AR and CR service parameter are omitted.

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

IOPS

This parameter shall contain the values GOOD if data values are valid, BAD_BY_DEVICE, BAD_BY_SLOT, or BAD_BY_SUBSLOT.

Input Data

This parameter contains the value of the Input Data object element.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.2 Set Input IOCS

This service shall be used by a consumer (IO controller) of inputs to pass the value of changed IOCS to the PROFINET IO Application Layer. The IOCS is conveyed back to the provider and provides a means to monitor the application. This service shall only be used in conjunction with the IO AR and Supervisor AR.

By this service the application process of the IO device updates the local buffer. According to the cyclic buffer to buffer transportation characteristics of the used application relationship the following behavior is possible:

- The Set Input IOCS requests are issued faster than the contents of the buffers are transported over the network to the Producer. In this case not each value of the IOCS is conveyed over the network. Only the latest value is transported, others are locally overwritten.
- The Set Input IOCS requests are issued slower than the contents of the buffers are transported over the network to the Producer. In this case each value of the IOCS provided with the service is conveyed more than once over the network.
- If the Set Input IOCS requests are issued synchronous with the transport of the contents of the buffers each value of the IOCS provided with the service is conveyed over the network.
- If there are several IOCS than only the IOCS of the non-shared CR will be indicated (see clause 4.3.2.5.4).

Table 9 shows the parameters of the service.

Table 9 — Set Input IOCS

Parameter name	Req	Cnf
Argument	M	M
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOCS	M	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

IOCS

This parameter shall contain the values GOOD or BAD_BY_CONTROLLER.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.3 Get Input

This service shall only be used in conjunction with the IO AR and Supervisor AR. It is used by an IO controller to read the values of an input data subslot with the related IOPS.

Table 10 shows the parameters of the service.

Table 10 — Get Input

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOPS		M
Input Data		C
New Flag		M
IOCS		C
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

Result(+)

This parameter indicates that the service request succeeded.

IOPS

This parameter contains the value of the corresponding attribute of the Input Data object.

Input Data

This parameter contains the value of the Input Data object.

New Flag

This parameter contains the value TRUE if new Input data with IOPS have been received since last time calling the service. If the service was called more frequently the New Flag is set to FALSE by the protocol machines.

IOCS

This parameter contains the value of the IOCS in case there is corresponding Output Data object.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.4 Get Input IOCS

This service shall only be used in conjunction with the IO AR and Supervisor AR. It is used by a provider to get the IOCS of the IO controller.

Table 11 shows the parameters of the service.

Table 11 — Get Input IOCS

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOCS		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

Result(+)

This parameter indicates that the service request succeeded.

IOCS

This parameter contains the value of the IOCS.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.5 New Input

The New Input service signals to the application of the IO Controller a reception of a valid APDU. This service shall also be conveyed if the IOPS has been changed. Furthermore, this service is used to indicate that the Watchdog expired with the parameter Watchdog Flag.

Table 12 shows the parameters of the service.

Table 12 — New Input

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Slot Number	M
Subslot Number	M
Watchdog Flag	C
InData Flag	C

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR passed with the indication.

CREP

This parameter is the local identifier for the desired CR passed with the indication.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

Watchdog Flag

This parameter contains the value WATCHDOG_EXPIRED if no APDU or no valid APDU Data Status has been received within the Watchdog Time Interval. The AR will be terminated in parallel if the Data Hold Time Interval contains the same value. Otherwise the application behavior within the IO controller is a local means.

InData Flag

This parameter indicates that the first valid frame was received. All consumer IOCRs of the AR have got data first time.

4.3.2.2.3.6 Set Input APDU Data Status

This service shall be used by the IO device to set the flags “Data Flag”, “AR State Flag”, “Provider State Flag”, and “Problem Indicator Flag”. These flags modify the field APDU Data Status of the next conveyed APDU. The flags “Data Valid Flag”, “AR State Flag”, and “Problem Indicator Flag” influence only the selected AR. The flag “Provider State Flag” can influence all established ARs of the IO device.

Table 13 shows the parameters of the service.

Table 13 — Set Input APDU Data Status

Parameter name	Req	Cnf
Argument	M	
AREP	U	
Data Valid Flag	U	
AR State Flag	U	
Provider State Flag	U	
Problem Indicator Flag	U	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR passed with the request. It is omitted if the service parameter Provider State Flag is used because this parameter influences all established ARs transmitting IO Data.

Data Valid Flag

This parameter indicates with the value “DataItem invalid” that all data (inclusive IOPS/IOCS) of the AR are not valid and shall not be processed by the consumer.

NOTE The Data Flag with “DataItem invalid” could be used if the application process of the IO device is not able to set Input Data and/or IOPS/IOCS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS/IOCS should be used to tell the consumer about the status of the input data. This flag is also used for redundancy in future versions.

The value “DataItem valid” shall be the default value for regular operation to indicate that the data values including the IOPS/IOCS are valid.

AR State Flag

This parameter indicates if the AR is operating with the value “primary” or standby with the value “backup”.

NOTE The value “backup” is reserved for further versions including system redundancy.

Provider State Flag

This parameter indicates if the application process is running with the value “run” or stopped with the value “stop”. This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS.

NOTE This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag

This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE. It is indicated by the value “Problem detected”. It shall be set to “Regular operation” as long as no diagnosis entry exists.

NOTE This flag could be used to trigger the reading of diagnosis data of the application within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to “BAD” for local reasons by means of the Set Input service.

The default values for the flags above are:

- Data Flag: “DataItem valid”
- AR State Flag: “primary”
- Provider State Flag: “run”
- Problem Indicator Flag: “Regular operation”

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.7 New Input APDU Data Status

The service signals a changed of the APDU Data Status value to the application of the IO controller. This service shall only be conveyed if the APDU Data Status value has been changed by the IO device and shall provide only the changed flags.

Table 14 shows the parameters of the service.

Table 14 — New Input APDU Data Status

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Data Valid Flag	C
AR State Flag	C
Provider State Flag	C
Problem Indicator Flag	C

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR passed with the indication.

CREP

This parameter is the local identifier for the desired CR passed with the indication.

Data Valid Flag

This parameter indicates with the value "DataItem invalid" that all data (inclusive IOPS) of the AR are not valid and shall not be processed by the consumer.

NOTE The Data Flag with "DataItem invalid" could be used if the application process of the IO device is not able to set Input Data and/or IOPS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS should be used to tell the consumer about the status of the input data.

The value "DataItem valid" indicate that the data values including the IOPS, IOCS are valid.

AR State Flag

This parameter indicates if the AR is operating with the value "primary" or standby with the value "backup".

NOTE The value "backup" is reserved for further versions including redundancy.

Provider State Flag

This parameter indicates if the application process is running with the value „run“ or stopped with the value „stop“. This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS.

NOTE This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag

This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE, which affects the related AR. It is indicated by the value „Problem detected“. It shall be set to „Regular operation“ as long as no diagnosis entry exists.

NOTE This flag could be used to trigger the reading of diagnosis data within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to "BAD" for local reasons by means of the Set Input service.

4.3.2.2.3.8 Read Input Data

This confirmed service may be used to read the value of an Input Data object element. This service can be used in conjunction with the IO AR, Supervisor AR, or implicit AR.

NOTE A client application may use the service Read AR Data to read possible AR UUIDs of established ARs that could be used in this context.

Implicit AR using Target AR UUID: This service does only contain an Input Data object if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service contains an Input Data object.

Table 15 shows the parameters of the service.

Table 15 — Read Input Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Length IOCS			M	M(=)
Length IOPS			M	M(=)
Length Input Data			M	M(=)
IOCS			M	M(=)
IOPS			M	M(=)
Input Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Input Data object in a specific slot (typically a module).

Subslot Number

The parameter Subslot Number is used in the destination device for addressing the desired Input Data object in a specific subslot (typically a submodule).

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of a Input Data object that has to be read. The allowed length is from 2^0 to $2^{32} - 256$. If the requested length exceeds the actual length of the Input Data object element then the actual length with data shall be

responded. If the requested length is shorter than the length of the Input Data object element then a parameter error shall be the result.

Result(+)

This parameter indicates that the service request succeeded.

Length IOPS

This parameter contains the value of the corresponding attribute of the Input Data object.

Length IOCS

This parameter contains the value of the corresponding attribute of the Input Data object.

Length Input Data

This parameter contains the data length of the object which has been read.

IOCS

This parameter contains the value of the corresponding attribute of the Input Data object.

IOPS

This parameter contains the value of the corresponding attribute of the Input Data object.

Input Data

This parameter contains the value of the object which has been read and consists of the number of octets indicated in the parameter Length Input Data. This parameter has to be composed of the data types defined in 4.2.2.1.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNIRW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.2.3.9 Set Output

The Set Output service sets new output data of one slot/subslot together with its IOPS for next transmission for all related application relationships. The IOPS shall be at any time consistent with delivered Output Data values. The value of IOPS shall only be "GOOD" for valid output data of the whole subslot. In all other cases the value shall be set to "BAD_BY_CONTROLLER". This service is used by the application process of the IO controller to update the addressed part of the protocol internal cyclic buffer. The actual transmission of the buffer is time triggered.

NOTE The local implementation of the service interface may support access variants with other than subslot granularity e.g. for more than one subslot or only parts (variables) of the Output Data object element. However, the Output Data object element is always consistent with its IOPS.

This service shall only be used in conjunction with the IO AR or Supervisor AR.

According to the cyclic buffer to buffer transportation characteristics the following behavior is possible:

- The Set Output requests issued faster than the contents of the buffers are transported over the network. In this case only the values provided with the latest Set Output service are conveyed over the network.
- The Set Output requests issued slower than the contents of the buffers are transported over the network. In this case the values provided with each Set Output service are conveyed more than once over the network.
- If the Set Output requests are issued synchronous with the transport of the contents of the buffers the values provided with each Set Output service are conveyed once over the network.

Table 16 shows the parameters of the service.

Table 16 — Set Output

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOPS	M	
Output Data	M	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the subslot number.

IOPS

This parameter shall contain the values GOOD if data values are valid or BAD_BY_CONTROLLER if this data are not valid.

Output Data

This parameter contains the value of the Output Data object element.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.10 Set Output IOCS

This service shall be used by a consumer (IO device) of outputs to pass the value of changed IOCS to the PROFINET IO Application Layer. The IOCS is conveyed back to the provider and

provides a means to monitor the application. This service shall only be used in conjunction with the IO AR and Supervisor AR.

By this service the application process of the IO device updates the local buffer. According to the cyclic buffer to buffer transportation characteristics of the used application relationship the following behavior is possible:

- The IO Status Change requests issued faster than the contents of the buffers are transported over the network to the Producer. In this case not each value of the IOCS is conveyed over the network. Only the latest value is transported, others are locally overwritten.
- The IO Status Change requests are issued slower than the contents of the buffers are transported over the network to the Producer. In this case each value of the IOCS provided with the service is conveyed more than once over the network.
- If the IO Status Change requests are issued synchronous with the transport of the contents of the buffers each value of the IOCS provided with the service is conveyed over the network.

Table 17 shows the parameters of the service.

Table 17 — Set Output IOCS

Parameter name	Req	Cnf
Argument	M	M
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
IOCS	M	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

IOCS

This parameter may contain the values GOOD, BAD_BY_DEVICE, BAD_BY_SLOT, or BAD_BY_SUBSLOT.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.11 Get Output

This service shall be used by an IO device to read the value for one slot/subslot of an Output Data object element together with its IOPS from the PROFINET IO Application Layer. This service shall only be used in conjunction with the IO AR or Supervisor AR.

The IO device shall only process the values of the Output Data object element if the IOPS contains the value GOOD (see Output Data ASE attribute description in 4.3.2.2.2.2)

Table 18 shows the parameters of the service.

Table 18 — Get Output

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOPS		M
Output Data		M
New Flag		M
IOCS		C
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR passed with the indication.

CREP

This parameter is the local identifier for the desired CR passed with the indication.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

Result(+)

This parameter indicates that the service request succeeded.

IOPS

This parameter contains the value of the corresponding attribute of the Output Data object element.

Output Data

This parameter contains the value of the Output Data object.

New Flag

This parameter contains the value TRUE if new Output data with IOPS have been received since last time calling the service. If the service was called more frequently the New Flag is set to FALSE by the protocol machines.

IOCS

This parameter contains the value of the IOCS in case there is a corresponding Input Data object.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.12 Get Output IOCS

This service shall be used by an IO device to read the IOCS from the PROFINET IO Application Layer. This service shall only be used in conjunction with the IO AR or Supervisor AR.

Table 19 shows the parameters of the service.

Table 19 — Get Output IOCS

Parameter name	Req	Cnf
Argument	M	
AREP	M	
CREP	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOCS		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR passed with the indication.

CREP

This parameter is the local identifier for the desired CR passed with the indication.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

Result(+)

This parameter indicates that the service request succeeded.

IOCS

This parameter contains the value of the IOCS.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.13 New Output

The New Output service signals to the application of the IO device a reception of a valid APDU. This service shall also be conveyed if the IOPS has been changed. Furthermore, this service is used to indicate that the Watchdog expired with the parameter Watchdog Flag.

Table 20 shows the parameters of the service.

Table 20 — New Output

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Slot Number	M
Subslot Number	M
Watchdog Flag	C
InData Flag	C

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR passed with the indication.

CREP

This parameter is the local identifier for the desired CR passed with the indication.

Slot Number

This parameter contains the local slot number.

Subslot Number

This parameter contains the local subslot number.

Watchdog Flag

This parameter contains the value WATCHDOG_EXPIRED if no APDU or no valid APDU Data Status has been received within the Watchdog Time Interval. The AR will be terminated in parallel if the Data Hold Time Interval contains the same value. Otherwise the application behavior within the IO device is to hold the last valid value until the communication continued or the AR is terminated by the Context Management ASE.

InData Flag

This parameter indicates that the first valid frame was received. All consumer IOCRs of the AR have got data first time.

4.3.2.2.3.14 Set Output APDU Data Status

This service shall be used by the IO controller to set the flags “Data Flag”, “AR State Flag”, “Provider State Flag”, and “Problem Indicator Flag”. These flags modify the field APDU Data Status of the next conveyed APDU. The flags “Data Flag”, “AR State Flag”, and “Problem Indicator Flag” influence only the selected AR. The flag “Provider State Flag” influences all established ARs of the IO device.

Table 21 shows the parameters of the service.

Table 21 — Set Output APDU Data Status

Parameter name	Req	Cnf
Argument	M	
AREP	U	
Data Flag	U	
AR State Flag	U	
Provider State Flag	U	
Problem Indicator Flag	U	
Result(+)		S
AREP		M
Result(-)		S
AREP		M

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR passed with the request. It is omitted if the service parameter Provider State Flag is used because this parameter influences all established ARs transmitting IO Data.

Data Flag

This parameter indicates with the value “DataItem invalid” that all data (inclusive IOPS) of the AR are not valid and shall not be processed by the consumer.

NOTE The Data Flag with “DataItem invalid” could be used if the application process of the IO controller is not able to set Input Data and/or IOPS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS should be used to tell the consumer about the status of the output data.

The value “DataItem valid” shall be the default value for regular operation to indicate that the data values including the IOPS are valid.

AR State Flag

This parameter indicates if the AR is operating with the value „primary“ or standby with the value „backup“. The default value is „primary“.

NOTE The value „backup“ is reserved for further versions including redundancy.

Provider State Flag

This parameter indicates if the application process is running with the value „run“ or stopped with the value „stop“. This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS. In case of „stop“ all IOPS shall also be set to the value „BAD_BY_CONTROLLER“ by means of the Set Input service request to all slots/subslots.

NOTE This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag

This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE, which affects the related AR. It is indicated by the value „Problem detected“. It shall be set to „Regular operation“ as long as no diagnosis entry exists. It is also the default value.

NOTE This flag could be used to trigger the reading of diagnosis data within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to „BAD“ for local reasons by means of the Set Input service.

In summary, initial or default values for the flags above are:

- Data Flag: „DataItem valid“
- AR State Flag: „primary“
- Provider State Flag: „run“
- Problem Indicator Flag: „Regular operation“

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.2.3.15 New Output APDU Data Status

The service signals a changed of the APDU Data Status value to the application of the IO device. This service shall only be conveyed if the APDU Data Status value has been changed by the IO controller and shall provide only the changed flags.

Table 22 shows the parameters of the service.

Table 22 — New Output APDU Data Status

Parameter name	Ind
Argument	M
AREP	M
CREP	M
Data Flag	C
AR State Flag	C
Provider State Flag	C
Problem Indicator Flag	C

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR passed with the indication.

CREP

This parameter is the local identifier for the desired CR passed with the indication.

Data Flag

This parameter indicates with the value DATA_INVALID that all data (inclusive IOPS) of the AR are not valid and shall not be processed by the consumer.

NOTE The Data Flag with DATA_INVALID could be used if the application process of the IO controller is not able to set Input Data and/or IOPS due to a fatal application error. However, it should be used only in such serious cases because the Data Hold Time times out at the consumer and terminates the connection. In all other cases the IOPS should be used to tell the consumer about the status of the input data. This flag is also used for redundancy in future versions.

The value „DataItem valid“ shall be the default value for regular operation to indicate that the data values including the IOPS are valid.

AR State Flag

This parameter indicates if the AR is operating with the value „primary“ or standby with the value „backup“. The default value is „primary“.

NOTE The value „backup“ is reserved for further versions including redundancy.

Provider State Flag

This parameter indicates if the application process is running with the value „run“ or stopped with the value „stop“. This value is only informative for the consumer application process, the actual validity of the data shall be according to IOPS. In case of „stop“ all IOPS shall also be set to the value „BAD_BY_CONTROLLER“ by means of the Set Input service request to all slots/subslots.

NOTE This flag could be used to control a global indicator to visualize the global remote application status.

Problem Indicator Flag

This parameter indicates if at least one diagnoses value is set within the Diagnoses ASE. It is indicated by the value „Problem detected“. It shall be set to „Regular operation“ as long as no diagnosis entry exists. It is also the default value.

NOTE This flag could be used to trigger the reading of diagnosis data within the IO controller. This flag does not say anything about the validity of input data that is only indicated by IOPS. A diagnosis entry may cause also to change the IOPS to „BAD“ for local reasons by means of the Set Input service.

4.3.2.2.3.16 Read Output Data

This confirmed service may be used to read the value of an Output Data object element or the substitute value. This service can be used in conjunction with the IO AR, Supervisor AR, or the implicit AR.

NOTE A client application may use the service Read AR Data to read possible AR UUIDs of established ARs that could be used in this context.

Implicit AR using Target AR UUID: This service does only contain an Output Data object if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service does only contain an Output Data object if the application supports reading of values of Output Data objects without an established AR. Otherwise a parameter error is responded.

Table 23 shows the parameters of the service.

Table 23 — Read Output Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Length IOCS			M	M(=)
Length IOPS			M	M(=)
Length Output Data			M	M(=)
IOCS			M	M(=)
IOPS			M	M(=)
Output Data			M	M(=)
Substitute Mode			M	M(=)
Substitute Active Flag			M	M(=)
Output Substitute Data			U	U(=)

Parameter name	Req	Ind	Rsp	Cnf
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall be used in conjunction with the implicit AR to address the desired AR within the IO device.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Output Data object in a specific slot (typically a module).

Subslot Number

The parameter Subslot Number is used in the destination device for addressing the desired Output Data object in a specific subslot (typically a submodule).

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of a Output Data object that has to be read. The allowed length is from 2^0 to $2^{32}-256$. If the requested length exceeds the actual length of the Output Data object element then the actual length with data shall be responded. If the requested length is shorter than the length of the Output Data object element then a parameter error shall be the result.

Result(+)

This parameter indicates that the service request succeeded.

Length IOPS

This parameter contains the value of the corresponding attribute of the Output Data object.

Length IOCS

This parameter contains the value of the corresponding attribute of the Output Data object.

Length Output Data

This parameter contains the data length of the object which has been read.

IOCS

This parameter contains the value of the corresponding attribute of the Output Data object.

IOPS

This parameter contains the value of the corresponding attribute of the Output Data object.

Output Data

This parameter contains the value of the object which has been read and consists of the number of octets indicated in the Length Output Data of the response. This parameter has to be composed of the data types defined in 4.2.2.1.

Substitute Mode

This parameter contains the value of the corresponding attribute of the Output Data object.

Substitute Active Flag

This parameter contains the value TRUE if substitute data are used in case of a failure.

Output Substitute Data

This parameter contains the substitute value of the object which has been read and consists of the number of octets indicated in the Length Output Data of the response. This parameter has to be composed of the data types defined in 4.2.2.1.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.2.3.17 Write Output Substitute Data

This confirmed service may be used to write the substitute value of an Output Data object element. This service can be used in conjunction with the IO AR and Supervisor AR. Table 24 shows the parameters of the service.

The service parameter “Multiple” may be used to convey substitute values within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

Table 24 — Write Output Substitute Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Substitute Mode	M	M(=)		
Length Output Data	M	M(=)		
Output Substitute Data	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used in the destination device for addressing the desired Output Data object in a specific slot (typically a module).

Subslot Number

The parameter Subslot Number is used in the destination device for addressing the desired Output Data object in a specific subslot (typically a submodule).

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Substitute Mode

This parameter contains the value of the corresponding attribute of the Output Data object.

Length Output Data

This parameter contains the value of the corresponding attribute of the Output Data object.

Output Substitute Data

This parameter contains the substitute value of the object which shall be written and consists of the number of octets indicated in the parameter Length Output Data. This parameter has to be composed of the data types defined in 4.2.2.1.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.2.4 Behavior of IO data objects**4.3.2.2.4.1 General behavior of the input data object**

For performance reasons, the value of the Input Data object should be transferred on change as fast as possible to the IO AR buffer for transmission. The speed and the jitters of this action is a performance parameter of an IO device.

4.3.2.2.4.2 General behavior of the output data object

For performance reasons, the value for the Output Data object should be transferred on receipt as fast as possible to the object. The speed and the jitters of this action is a performance parameter of a IO device. This parameter determines the synchronization accuracy.

The Read Output service shall provide the current value of the application process which has been fetched with the Get Output service.

NOTE Due to different reasons (e.g. physical output delays) this value may differ from the value of the real output data or from the last transferred value.

4.3.2.3 Logbook data ASE

4.3.2.3.1 Overview

The Logbook Data ASE provides a mean to store locally generated and AR specific events in a circular buffer and provides a service to read that event buffer. The content of entries is manufacturer specific and may be used to identify communication problems e.g. reasons for AR termination.

The Logbook Data ASE defines attributes of Logbook Data objects and provides a set of services used to read remotely, and write their values locally. A Logbook Data ASE is specific for an IO Device. The Logbook Data object is addressed implicitly by the defined services. The entries shall always be responded from the newest to the oldest one up to the requested length.

The access to Logbook Data objects is performed according to the Client/Server access model for read. The Client/Server model is characterized by a Client application sending a read request to a Server application that responds accordingly.

The formal model of the Logbook Data ASE is presented by the Logbook Data class specification, containing a description of its attributes, services as well as invocations, followed by a detailed service specification.

4.3.2.3.2 Logbook data class specification

4.3.2.3.2.1 General

The Logbook Data ASE defines one Logbook Data object type:

4.3.2.3.2.2 Logbook data class specification

4.3.2.3.2.2.1 Template

A Logbook Data object is described by the following template:

DP ASE:	Logbook Data ASE
CLASS:	Logbook Data
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: Current Local Time Stamp
3	(m) Attribute: List of Entries
3.1	(m) Attribute: Local Time Stamp
3.2	(m) Attribute: AR UUID
3.3	(m) Attribute: PNIO Status
3.4	(m) Attribute: Entry Detail
SERVICES:	
1	(m) OpsService: Read Logbook
2	(m) OpsService: Logbook Event

4.3.2.3.2.2.2 Attributes

Implicit

The attribute Implicit indicates that the Logbook Data object is implicitly addressed by the services.

Current Local Time Stamp

This attribute contains the current local time by means of the value of the cycle counter.

Attribute Type: Unsigned64

List of Entries

An IO Device shall support at least 16 logbook entries and an IO Controller shall support at least 4 KByte. This attribute list consists of the following attributes.

Local Time Stamp

This attribute contains the local time when the entry was made by means of the value of the cycle counter at this point of time.

Attribute Type: Unsigned64

AR UUID

This attribute defines the UUID of the IO AR entry which is the source.

Attribute Type: UUID

PNIO Status

This attribute contains the protocol machine defined value of the entry.

Attribute Type: Unsigned32

Allowed Values: see Part 6 of IEC 61158

Entry Detail

This attribute contains the protocol machine defined value of the entry.

Attribute Type: Unsigned32

4.3.2.3.3 Logbook data service specification**4.3.2.3.3.1 Read Logbook**

This confirmed service may be used to read the value of a Logbook Data object. This service can be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 25 shows the parameters of the service.

Table 25 — Read Logbook

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Current Local Time Stamp			M	M(=)
List of Entries			M	M(=)
Local Time Stamp			M	M(=)
AR UUID			M	M(=)
PNIO Status			M	M(=)
Entry Detail			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the

last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of a Logbook Data object that has to be read. The allowed length is from 0 to $2^{32}-65$.

Result(+)

This parameter indicates that the service request succeeded.

Current Local Time Stamp

This parameter contains the value of the corresponding attribute of the ASE object.

List of Entries

This parameter is composed of the following list elements:

NOTE If list of entries have more bytes than the requested length, only the newest entries shall used for the response.

Local Time Stamp

This parameter contains the value of the corresponding attribute of the ASE object.

AR UUID

This parameter contains the value of the corresponding attribute of the ASE object.

PNIO Status

This parameter contains the value of the corresponding attribute of the ASE object.

Entry Detail

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNIORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled, invalid subslot, invalid sequence number, invalid API, invalid AR.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.3.3.2 Logbook Event

This local service indicates value for the Logbook Data object. Table 26 shows the parameters of the service.

Table 26 — Logbook Event

Parameter name	Ind
Argument	M
AREP	M
Local Time Stamp	M
AR UUID	M
PNIO Status	M
Entry Detail	M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Local Time Stamp

This parameter contains the value of the corresponding attribute of the ASE object.

AR UUID

This parameter contains the value of the corresponding attribute of the ASE object.

PNIO Status

This parameter contains the value of the corresponding attribute of the ASE object.

Entry Detail

This parameter contains the value of the corresponding attribute of the ASE object.

4.3.2.4 Diagnosis ASE**4.3.2.4.1 Overview**

In the PROFINET IO Application Layer environment, the Diagnosis ASE of a IO device contains different Diagnosis entries that are used to keep the diagnosis information the IO device. The diagnosis information from the IO device is related to the addressed AP and includes only those slots, subslots and channels which have set diagnosis.

NOTE If a slot, subslot or channel does not have diagnosis set then this information it is not transmitted.

The Diagnosis ASE defines attributes of the Diagnosis objects and provides a set of services used to read their values. The following Diagnosis types are defined:

- Channel Diagnosis with additional info
- Channel Diagnosis
- Manufacturer Specific Diagnosis:
 - Manufacturer Specific Diagnosis indicating a failure (Channel Properties Specifier value: Error Appears)
 - Manufacturer Specific Diagnosis indicating a status (Channel Properties Specifier value: Error Disappears)

It is strongly recommended to use Channel Diagnosis in all matching cases because detailed semantics are fully standardized. Additional vendor specific diagnosis data can be conveyed by means of the Manufacturer Specific Diagnosis.

The application of an IO device may provide the value of one or several Diagnosis objects to the implicit AR, IO AR, and the Supervisor AR.

The Diagnosis model uses the Client/Server access model.

Furthermore, in case of setting the first diagnosis entry the Input Data ASE shall be informed by means of the event (Diagnosis ASE "First Diagnosis Entry set"). In case of unsetting the last diagnosis entry the Input Data ASE shall be informed by means of the event (Diagnosis ASE "Last Diagnosis Entry cleared").

The Alarm ASE may also be informed to send a Diagnosis Alarm if it is configured by the application.

4.3.2.4.2 Diagnosis class specification

4.3.2.4.2.1 Device diagnosis class specification

4.3.2.4.2.1.1 Template

The AP Device Diagnosis object is described by the following template:

DP ASE:	Diagnosis ASE
CLASS:	AP Diagnosis
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: Device Status
3	(c) Constraint: Device Status = DEVICE_FAILURE
3.1	(m) Attribute: List of APs
3.1.1	(m) Attribute: API
3.1.2	(m) Attribute: List of Slots
3.1.2.1	(m) Attribute: Slot Number
3.1.2.2	(m) Attribute: List of Subslots
3.1.2.2.1	(m) Attribute: Subslot Number
3.1.2.2.2	(m) Attribute: Submodule Diagnosis State
3.1.2.2.3	(m) Attribute: List of Channels
3.1.2.2.3.1	(m) Attribute: Channel Number
3.1.2.2.3.2	(o) Attribute: List of Channel Diagnosis
3.1.2.2.3.2.1	(m) Attribute: Channel Properties
3.1.2.2.3.2.1.1	(m) Attribute: Type
3.1.2.2.3.2.1.2	(m) Attribute: Specifier
3.1.2.2.3.2.1.3	(m) Attribute: Direction
3.1.2.2.3.2.2	(m) Attribute: Channel Error Type
3.1.2.2.3.3	(o) Attribute: List of Ext Channel Diagnosis
3.1.2.2.3.3.1	(m) Attribute: Channel Properties
3.1.2.2.3.3.1.1	(m) Attribute: Type
3.1.2.2.3.3.1.2	(m) Attribute: Specifier
3.1.2.2.3.3.1.3	(m) Attribute: Direction
3.1.2.2.3.3.2	(m) Attribute: Channel Error Type
3.1.2.2.3.3.3	(m) Attribute: Ext Channel Error Type
3.1.2.2.3.3.4	(m) Attribute: Ext Channel Add Value
3.1.2.2.3.4	(o) Attribute: List of Manufacturer Specific Diagnosis
3.1.2.2.3.4.1	(m) Attribute: Channel Properties
3.1.2.2.3.4.1.1	(m) Attribute: Type
3.1.2.2.3.4.1.2	(m) Attribute: Specifier
3.1.2.2.3.4.1.3	(m) Attribute: Direction
3.1.2.2.3.4.2	(m) Attribute: User Structure Identifier
3.1.2.2.3.4.3	(m) Attribute: List of Data
3.1.2.2.3.4.3.1	(m) Attribute: Data
SERVICES:	
1	(m) OpsService: Read Device Diagnosis
2	(m) OpsService: Diagnosis Event

4.3.2.4.2.1.2 Attributes

Implicit

The attribute Implicit indicates that the Device Diagnosis object is implicitly addressed by the services.

Device Status

This attribute contains the diagnosis summary. If the value is set to DEVICE_OK no slot/subslot shall contain diagnosis information. If the value is set to DEVICE_FAILURE at least one slot/subslot shall contain diagnosis information.

Attribute Type: Unsigned8

Allowed values: DEVICE_OK, DEVICE_FAILURE

List of APs

This attribute contains the diagnosis summary specific for each supported application process. A list element consists of the following attributes.

API

This attribute shall consist only of APIs with diagnosis information.

List of Slots

This attribute shall contain only slots with diagnosis information. A list element consists of the following attributes.

Slot Number

This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Subslots

This attribute shall contain only subslots with diagnosis information. A list element consists of the following attributes.

Subslot Number

This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Submodule Diagnosis State

This attribute contains the diagnosis summary for the submodule. The value SUBMODULE_FAILURE shall only be set in case of at least on Channel Diagnosis severity other than INFORMATION and at least one Manufacturer Specific Diagnosis with severity other than INFORMATION and channel properties specifier shall be set to Error Appears. In all other cases the value shall be set to SUBMODULE_OK.

Attribute Type: Unsigned8

Allowed values: SUBMODUL_OK, SUBMODUL_FAILURE

List of Channels

This attribute shall contain only channels with diagnosis information. A list element consists of the following attributes.

Channel Number

This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute Type: Unsigned16

Allowed values: 0-0x7FFF user specific channel number, 0x8000 submodule

List of Channel Diagnosis

This attribute contains a list of channel diagnosis. A list element consists of the following attributes.

Channel Properties

This attribute consists of the following attributes.

Type

This attribute indicates the channel-size to which the Channel Diagnosis object is related.

Allowed values:

UNSPECIFIC shall be used if the Channel Number contain the value 0x8000 or no other matches,

1BIT, 2BIT, 4BIT, BYTE, WORD, 2WORDS, 4WORDS

Specifier

This attribute indicates the the Channel Diagnosis appears.

Allowed values: ERROR_APPEARS,
ERROR_DISAPPEARS_AND_ERROR_FREE,
ERROR_DISAPPEARS_BUT_OTHER_ERROR_REMAIN

Direction

This attribute indicates the direction of the channel to which the Channel Diagnosis object is related.

Attribute Type: Unsigned8

Allowed values: MANUFACTURER_SPECIFIC, INPUT_CHANNEL,
OUTPUT_CHANNEL, BIDIRECTIONAL_CHANNEL

Channel Error Type

This attribute indicates the error type of the Channel Related Diagnosis.

Attribute Type: Unsigned16

Allowed values:

SHORT_CIRCUIT,

UNDERVOLTAGE,

OVERVOLTAGE,

OVERLOAD,

OVERTEMPERATURE,

LINE_BREAK,

UPPER_LIMIT_VALUE_EXCEEDED,

LOWER_LIMIT_VALUE_EXCEEDED,

ERROR,

MANUFACTURER_SPECIFIC_1 recommended for “parametrization fault”,

MANUFACTURER_SPECIFIC_2 recommended for “power supply fault”,

MANUFACTURER_SPECIFIC_3 recommended for “fuse blown / open”,

MANUFACTURER_SPECIFIC_4,

MANUFACTURER_SPECIFIC_5 recommended for “ground fault”,

MANUFACTURER_SPECIFIC_6 recommended for “reference point lost”,

MANUFACTURER_SPECIFIC_7 recommended for “process event lost / sampling error”,

MANUFACTURER_SPECIFIC_8 recommended for “threshold warning”,

MANUFACTURER_SPECIFIC_9 recommended for “output disabled”,

MANUFACTURER_SPECIFIC_10 recommended for “safety event”,

MANUFACTURER_SPECIFIC_11 recommended for “external fault”,

MANUFACTURER_SPECIFIC_12 up to MANUFACTURER_SPECIFIC_001F
 MANUFACTURER_SPECIFIC_0100 up to MANUFACTURER_SPECIFIC_2FFF
 MANUFACTURER_SPECIFIC_3000 up to MANUFACTURER_SPECIFIC_4FFF
 MANUFACTURER_SPECIFIC_5000 up to MANUFACTURER_SPECIFIC_7FFF

List of Ext Channel Diagnosis

This attribute contains a list of extended channel diagnosis. A list element consists of the following attributes.

Channel Properties

This attribute has the same definition as in List of Channel Diagnosis.

Channel Error Type

This attribute indicates the error type of the Channel Related Diagnosis.

Attribute Type: Unsigned16

Allowed values: same values as specified for List of Channel Diagnosis. Furthermore, the following additional values are allowed.

DATA_TRANSMISSION_IMPOSSIBLE,
 REMOTE_MISMATCH,
 MEDIUM_REDUNDANCY_MISMATCH,
 SYNC_MISMATCH,
 ISOCHRONOUS_MODE_MISMATCH,
 MULTICAST_CR_MISMATCH,
 MULTICAST_PROVIDER_STATUS_MISMATCH

Ext Channel Error Type

This attribute depends on the attribute Channel Error Type and indicates an extended error type. The allowed values are shown in Table 27.

Attribute Type: Unsigned16

Table 27 — Ext Channel Error Type

ChannelErrorType	Value
SHORT_CIRCUIT	manufacturer specific
UNDERVOLTAGE	manufacturer specific
OVERVOLTAGE	manufacturer specific
OVERLOAD	manufacturer specific
OVERTEMPERATURE	manufacturer specific
LINE_BREAK	manufacturer specific
UPPER_LIMIT_VALUE_EXCEEDED	manufacturer specific
LOWER_LIMIT_VALUE_EXCEEDED	manufacturer specific
ERROR	manufacturer specific
MANUFACTURER_SPECIFIC_1 up to MANUFACTURER_SPECIFIC_001F	manufacturer specific
MANUFACTURER_SPECIFIC_0100 up to MANUFACTURER_SPECIFIC_7FFF	manufacturer specific
DATA_TRANSMISSION_IMPOSSIBLE	LINK_READY
	LINK_FAIL
	LINK_OK
	PORT_DISABLE
	PORT_DISCARDING
	PORT_LEARNING
	PORT_FORWARDING
REMOTE_MISMATCH	REMOTE_SYSTEM_DATA_MISMATCH
	REMOTE_SYSTEM_DATA_MATCH
	DATA_RATE_MISMATCH
	DATA_RATE_OKAY
	DUPLEX_MODE_MISMATCH
	DUPLEX_MODE_OKAY

ChannelErrorType	Value
	PROPAGATION_DELAY_MISMATCH
	PROPAGATION_DELAY_OKAY
	PMD_TYPE_MISMATCH
	PMD_TYPE_OKAY
MEDIUM_REDUNDANCY_MISMATCH	MEDIUM_REDUNDANCY_LOST
	MEDIUM_REDUNDANCY_OKAY
SYNC_MISMATCH	NO_SYNC_MESSAGE_RECEIVED
	WRONG_PTP_SUBDOMAIN_ID
	WRONG_IR_DATA_ID
	JITTER_OUT_OF_BOUNDARY
	SYNC_OKAY
ISOCRONOUS_MODE_MISMATCH	OUTPUT_TIME_FAILURE
	OUTPUT_TIME_FAILURE_DISAPPEARS
	INPUT_TIME_FAILURE
	INPUT_TIME_FAILURE_DISAPPEARS
MULTICAST_CR_MISMATCH	MULTICAST_CONSUMER_CR_TIMED_OUT
	ADDRESS_RESOLUTION_FAILED
	MULTICAST_CONSUMER_CR_OKAY
	MULTICAST_CONSUMER_CR_OKAY_WITH_M_PROVIDER_STATUS_MISMATCH
MULTICAST_PROVIDER_STATUS_MISMATCH	PROVIDER_STATE_CHANGED_TO_VALUE_BAD
	PROVIDER_STATE_CHANGED_TO_VALUE_GOOD

Ext Channel Add Value

This attribute contains an additional value.

Attribute Type: Unsigned32

List of Manufacturer Specific Diagnosis

This attribute contains a list of manufacturer specific diagnosis. A list element consists of the following attributes.

Channel Properties

This attribute has the same definition as in List of Channel Diagnosis.

User Structure Identifier

This attribute indicates the structure of the Manufacturer Specific Diagnosis Data attribute.

Attribute Type: Unsigned16

Allowed values:

MANUFACTURER_SPECIFIC_1 up to
MANUFACTURER_SPECIFIC_0x7FFF,
CHANNEL_DIAGNOSIS,
MULTIPLE_DIAGNOSIS

List of Data

This attribute contains a list of manufacturer specific diagnosis data. A list element consists of the following attributes.

Data

This attribute contains manufacturer specific diagnosis data.

Attribute Type: One of the following data types shall be used:

Boolean, Binary Date, Integer, Time of Day, Unsigned, Time-Difference, Floating Point, Network Time, VisibleString, Network Time Difference, OctetString

4.3.2.4.2.1.3 Invocation of the AP device diagnosis object

An IO device shall invoke one AP Diagnosis object for each application process which shall contain the diagnosis information of the IO device which its faulty slots, subslots and channels.

4.3.2.4.3 Diagnosis service specification**4.3.2.4.3.1 Read Device Diagnosis**

This confirmed service may be used to read the value of an AP Device Diagnosis object. This service shall only be used in conjunction with the implicit AR (Target AR UUID or API shall be set), IO AR or Supervisor AR.

By means of the service parameter API, AR UUID, Slot, and Subslot the content of diagnosis can be restricted to a specific API, AR, a specific Slot or a specific Subslot as a filter function. In case a user specific filter is selected the Read Diagnosis response shall only contain diagnosis items that match the filter criteria and contain diagnosis information. Otherwise all items with diagnosis shall be responded.

Furthermore, the diagnosis information is divided into Channel Diagnosis and Manufacturer Specific Diagnosis which is selected by the parameter Diagnosis Item.

Table 28 shows the parameters of the service.

Table 28 — Read Device Diagnosis

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Diagnosis Item	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of Diagnosis Data			M	M(=)
Channel Diagnosis			S	S(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Channel Diagnostic Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error Type			M	M(=)
Ext Channel Diagnosis			S	S(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Ext Channel Diagnosis Data			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
Channel Error Type			M	M(=)
Ext Channel Error Type			M	M(=)
Ext Channel Add Value			M	M(=)
Manufacturer Specific Diagnosis			S	S
Slot Number			M	M(=)
Subslot Number			M	M(=)
Channel Number			M	M(=)
Channel Properties			M	M(=)
User Structure Identifier			M	M(=)
List of Data			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific diagnosis information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE It is used to read only diagnosis information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only diagnosis information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used is used to address only diagnosis information for a specific slot/subslot. This service parameter shall only be used together with the service parameter Slot Number.

Diagnosis Item

This parameter is used to select CHANNEL or MANUFACTURER specific diagnosis. It is strongly recommended to use CHANNEL or CHANNEL WITH ADD INFO

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Diagnosis Data object that has to be read. The allowed length is from 10 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

List of Diagnosis Data

This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value

This parameter contains the value of the corresponding attribute of the ASE object.

Manufacturer Specific Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier

This parameter contains the value of the corresponding attribute of the ASE object.

List of Data

This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.4.3.2 Diagnosis Event

This service is used to indicate a diagnosis event detected by protocol machines. Table 29 shows the parameters of the service.

Table 29 — Diagnosis Event

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
CREP	M	M(=)	
Alarm Item	M	M(=)	
User Structure Identifier	M	M(=)	
List of Channel Diagnosis Data	S	S(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error Type	M	M(=)	
List of Diagnosis Data	S	S(=)	
Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error Type	M	M(=)	
Ext Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Ext Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error Type	M	M(=)	
Ext Channel Error Type	M	M(=)	
Ext Channel Add Value	M	M(=)	
Manufacturer Specific Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	

Parameter name	Req	Ind	Cnf
User Structure Identifier	M	M(=)	
List of Data	M	M(=)	
Data	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument

The argument shall convey the service specific parameters of the service indication.

AREP

This parameter is the local identifier for the desired AR.

CREP

This parameter is the local identifier for the desired CR.

Alarm Item

This parameter contains alarm specific data if present. The length shall 1408 octets not exceed.

User Structure Identifier

This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnosis Data

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

List of Diagnosis Data

This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value

This parameter contains the value of the corresponding attribute of the ASE object.

Manufacturer Specific Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier

This parameter contains the value of the corresponding attribute of the ASE object.

List of Data

This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data

This parameter contains the value of the corresponding attribute of the ASE object.

4.3.2.5 Alarm ASE

4.3.2.5.1 Overview

The alarm model allows the transfer of an alarm from the IO device to the assigned IO controller or vice versa and the explicit acknowledgement of the alarm.

For the transmission of an alarm the following conditions have to be fulfilled:

- the IO AR shall be established
- the corresponding Alarm Type has not to be enabled, the source can issue an alarm, the sink shall response with a negative acknowledgment “not supported” in case no handler for the Alarm Type is installed

The following Alarm Types are defined. They may be extended specific to the manufacturer:

Diagnosis appears Alarm

A diagnosis appears alarm signals an event within a submodule, for instance over-temperature, short circuit, etc. The content of the alarm is defined by the Diagnosis ASE for this type.

Process Alarm

A process alarm signals the occurrence of an event in the connected process, for instance upper limit value exceeded.

Pull Alarm

A slot signals the withdrawal of a submodule/module or change in configuration (reduction).

Plug Alarm

A slot signals the insertion of a submodule/module, a new need for parametrization, or a change in configuration (addition).

Status Alarm

A status alarm signals a change in the state of a submodule, for instance run, stop or ready.

Update Alarm

An update alarm signals the change of a parameter in a submodule e.g. by a local operation or a remote access.

Redundancy Alarm

A redundancy alarm signals the fault of one IO controller to the remaining IO controller for redundant IO ARs.

Controlled by supervisor

A slot signals the logical withdrawal of a submodule by the IO supervisor. The actions shall be according to the Pull Alarm.

Released Alarm

A slot signals the logical insertion of a submodule by the IO supervisor. The actions shall be according to the Plug Alarm.

Plug Wrong Submodule Alarm

A slot signals the insertion of a wrong submodule/module or a change in configuration (addition).

Return Of Submodule Alarm

A slot signals that a submodule is ready to switch its IOCS/IOPS from “BAD” to “GOOD” again without new parameterization.

Diagnosis disappears Alarm

A diagnosis disappears alarm signals an disappearing diagnosis event within a submodule. The content of the alarm is defined by the Diagnosis ASE for this type.

Multicast Communication Mismatch

A multicast consumer submodule signals that communication relationship to the associated multicast provider is failed.

Port Data Change Notification Alarm

A port submodule signals that port data has been changed.

Sync Data Change Notification Alarm

An interface submodule signals that synchronization data has been changed.

Isochronous Mode Problem Notification Alarm

The application signals that problems with isochronously execution have been detected.

Furthermore, manufacturer specific alarms may be used and alarms are reserved for profile specific definitions.

4.3.2.5.2 Alarm class specification

4.3.2.5.2.1 Template

An Alarm object is described by the following template:

DP ASE: **Alarm ASE**

CLASS: **ALARM**

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Key Attribute: Identifier
1.1	(m)	Attribute: Alarm Type
1.2	(m)	Attribute: Slot Number
1.3	(m)	Attribute: Subslot Number
2	(m)	Attribute: Alarm Specifier
2.1	(m)	Attribute: Sequence Number
2.2	(m)	Attribute: Channel Diagnosis
2.3	(m)	Attribute: Manufacturer Specific Diagnosis
2.4	(m)	Attribute: Submodule Diagnosis State
2.5	(m)	Attribute: AR Diagnosis State
3	(m)	Attribute: Module Ident Number
4	(m)	Attribute: Submodule Ident Number
5	(o)	Attribute: Alarm Item
5.1	(m)	Attribute: User Structure Identifier
5.2	(s)	Attribute: List of Data
5.2.1	(m)	Attribute: Data
5.3	(s)	Attribute: List of Channel Diagnosis Data
5.3.1	(m)	Attribute: Channel Number
5.3.2	(m)	Attribute: Channel Properties
5.3.3	(m)	Attribute: Channel Error Type
5.4	(s)	Attribute: List of Diagnosis Data
5.4.1	(s)	Attribute: Channel Diagnosis
5.4.1.1	(m)	Attribute: Slot Number
5.4.1.2	(m)	Attribute: Subslot Number
5.4.1.3	(m)	Attribute: List of Channel Diagnosis Data
5.4.1.3.1	(m)	Attribute: Channel Number
5.4.1.3.2	(m)	Attribute: Channel Properties
5.4.1.3.3	(m)	Attribute: Channel Error Type
5.4.2	(s)	Attribute: Ext Channel Diagnosis
5.4.2.1	(m)	Attribute: Slot Number
5.4.2.2	(m)	Attribute: Subslot Number
5.4.2.3	(m)	Attribute: List of Ext Channel Diagnosis Data
5.4.2.3.1	(m)	Attribute: Channel Number
5.4.2.3.2	(m)	Attribute: Channel Properties
5.4.2.3.3	(m)	Attribute: Channel Error Type
5.4.2.3.4	(m)	Attribute: Ext Channel Error Type
5.4.2.3.5	(m)	Attribute: Ext Channel Add Value
5.4.3	(s)	Attribute: Manufacturer Specific Diagnosis
5.4.3.1	(m)	Attribute: Slot Number
5.4.3.2	(m)	Attribute: Subslot Number
5.4.3.3	(m)	Attribute: Channel Number
5.4.3.4	(m)	Attribute: Channel Properties
5.4.3.5	(m)	Attribute: User Structure Identifier
5.4.3.6	(m)	Attribute: List of Data
5.4.3.6.1	(m)	Attribute: Data

6 (m) Attribute: Related AREP

7 (m) Attribute: Alarm Priority

SERVICES:

1 (m) OpsService: Alarm Notification

2 (m) OpsService: Alarm Ack

4.3.2.5.2.2 Attributes

Identifier

This key attribute is composed of Slot Number, Subslot Number, and Alarm Type to define to which structural slot (module), subslot and to which Alarm Type the Alarm object belongs. This Identifier is unique for each application process within the IO device and shall not be used by another object.

Alarm Type

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

The allowed values of Alarm Type are shown in Table 30. The value Diagnosis Appears shall only be used if at least one Additional Alarm Info with Channel Properties Specifier value of Diagnosis Appears is present. The value Diagnosis Disappears shall only be used if no Additional Alarm Info with Channel Properties Specifier value of Diagnosis Appears is present or if no Additional Alarm Info is present in case of error free notification for all reported previously errors.

Table 30 — Alarm Type

Alarm Type	Usage
Diagnosis Appears	The value Diagnosis Appears shall only be used - if at least one Additional Alarm Info with Channel Properties Specifier value of Diagnosis Appears is present, - if the manufacturer specific Additional Alarm Info with the meaning of an appearing event is present.
Process	indicates an alarm from the controlled process (usage application and process specific)
Pull	shall indicate an alarm if a used module/submodule is pulled (mandatory)
Plug	shall indicate an alarm if a requested module/submodule is plugged (mandatory)
Status	indicates a status change within a module/submodule
Update	indicates a change of parameter for a module/submodule
Redundancy	indicates a second IO controller that the primary one has failed (mandatory for IO controller redundancy)
Controlled	shall indicate an alarm if a used module/submodule has been locked by an IO supervisor (mandatory)
Released	shall indicate an alarm if a requested module/submodule has been unlocked by an IO supervisor, IO Controller or by IO Device local means (mandatory)
Plug Wrong Submodule	shall indicate an alarm if a wrong module/submodule for a requested module/submodule is plugged (mandatory)
Diagnosis Disappears	The value Diagnosis Disappears shall only be used - if no Additional Alarm Info with Channel Properties Specifier value of Diagnosis Appears is present, - or if no Additional Alarm Info is present in case of error free notification for all reported previously errors, - if no manufacturer specific Additional Alarm Info with the meaning of an appearing event is present.
Return Of Submodule	shall indicate a change of IOCS/IOPS from BAD to GOOD for a submodule (mandatory)
Profile Specific	shall be used according upcoming PNO Profile Guidelines if an IO devices complies to a specific device profile
Multicast Provider Communication Stopped	a multicast consumer shall indicate that the multicast CR has had a timeout (mandatory for multicast CRs with version 2.0)
Multicast Provider Communication Running	a multicast consumer shall indicate that the multicast CR is operating again (mandatory for multicast CRs with version 2.0)
Port Data Change Notification	indicates a change of port data e.g. link up or down

Alarm Type	Usage
Sync Data Changed Notification	indicates a change of clock synchronization
Isochronous Mode Problem Notification	indicates a problem of isochronous applications, e.g. started to late
Manufacturer Specific	Values from 0x0020 to 0x007F may be used for manufacturer specific alarm types

Slot Number

This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. It shall contain the physical or logical slot which issues the alarm.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines the subslot number as a local identifier. This number is unique within a device. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. It shall contain the physical or logical subslot which issues the alarm.

Attribute Type: Unsigned16

Allowed values: 0 to 0x8FFF are allowed

NOTE The value 0 is only used in conjunction with pull alarm.

Alarm Specifier

This attribute consists of the following attributes.

Sequence Number

This attribute defines the sequence number of the alarm. It shall be incremented with every Alarm Notification service. By means of the Sequence Number the receiver detects duplications and reflects the value of the field in the Alarm Ack.

Allowed values: 0 to 2047

Channel Diagnosis

This attribute shall only be used for the Alarm Type "Diagnosis". For all other Alarm Type's this attribute is not valid.

Table 31 shows the allowed values.

Table 31 — Channel Diagnosis

Value	Meaning
NO_CHANNEL_DIAG	means that the Subslot Number contains no Channel Diagnosis
CHANNEL_DIAG	means that the Subslot Number contains at least one Channel Diagnosis

Manufacturer Specific Diagnosis

This attribute shall only be used for the Alarm Type "Diagnosis". For all other Alarm Type's this attribute is not valid.

Table 32 shows the allowed values.

Table 32 — Manufacturer Specific Diagnosis

Value	Meaning
NO_MAN_DIAG	means that the Subslot Number contains no Manufacturer Specific Diagnosis
MAN_DIAG	means that the Subslot Number contains at least one Manufacturer Specific Diagnosis

Submodule Diagnosis State

This attribute shall only be used for the Alarm Type "Diagnosis". For all other Alarm Type's this attribute is not valid.

Table 33 shows the allowed values.

Table 33 — Submodule Diagnosis State

Value	Meaning
NO_DIAG	error free Furthermore, it indicates that all reported diagnosis have been cleared. An individual “disappears” notification can be omitted. However, even in this case a channel diagnosis locally treated as status and/or a Manufacturer Specific Diagnosis treated as status may be present.
DIAG	At least one diagnosis exists at the submodule. It could be one or more channel diagnosis and/or one or more Manufacturer Specific Diagnosis.

AR Diagnosis State

This attribute shall only be used for the Alarm Type “Diagnosis”. For all other Alarm Type's this attribute is not valid. The AR Diagnosis State shall correspond to the current state of the IO Data ASE attribute Problem Indicator Flag, which is related to all modules/submodules that belong to the AR.

Table 34 shows the allowed values.

Table 34 — AR Diagnosis State

Value	Meaning
NO_DIAG	error free Furthermore, it indicates that all reported diagnosis related to the AR have been cleared. An individual “disappears” notification can be omitted. However, even in this case a channel diagnosis locally treated as status and/or a Manufacturer Specific Diagnosis treated as status may be present.
DIAG	At least one diagnosis exists at one submodule related to the AR. It could be one or more channel diagnosis and/or one or more Manufacturer Specific Diagnosis.

Module Ident Number

This attribute contains the module identification.

Attribute Type: Unsigned32

NOTE The attribute value is manufacturer specific.

Submodule Ident Number

This attribute contains the submodule identification.

Attribute Type: Unsigned32

NOTE The attribute value is manufacturer specific.

Alarm Item

This attribute is composed of the following attributes.

User Structure Identifier

This attribute identifies the selection of the Alarm Notification data. The allowed values are shown in Table 35.

Table 35 — User Structure Identifier

Meaning	Usage
Manufacturer Specific	In conjunction with alarm type Diagnosis appears/disappears Manufacturer Specific Diagnosis in Alarm Notification and Diagnosis Data. In conjunction with other alarm types the usage is manufacturer specific.
Profile Specific	The meaning is defined in profile specifications.
Channel Diagnosis	Shall only be used in conjunction with Channel Diagnosis in Alarm Notification and Diagnosis Data.
Channel Diagnosis With Add Info	Shall only be used in conjunction with Channel Diagnosis With Add Info in Alarm Notification and Diagnosis Data.

Meaning	Usage
Multiple	Shall only be used in conjunction with alarm type, which comply to the Block Structure. It shall also be used with Multicast Consumer Info Data, List of Isochronous Mode Info Data, List of Port Info Data, List of Sync Info Data

Manufacturer specific structures shall only be used if PROFINET IO Channel Diagnosis does not cover the diagnosis information.

List of Data

This attribute contains a list of user specific alarm data.

The length of the user data shall 1408 octets not exceed.

A list element is composed of the following attributes:

Data

This attribute contains user specific alarm data.

Attribute Type: One of the following data types shall be used:

Boolean, Binary Date, Integer, Time of Day, Unsigned, Time-Difference, Floating Point, Network Time, VisibleString, Network Time Difference, OctetString

List of Channel Diagnosis Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute Type: Unsigned16

Allowed values: 0-0x7FFF user specific channel number, 0x8000 submodule

Channel Properties

This attribute has the same definition as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

Channel Error Type

This attribute indicates the error type of the Channel Related Diagnosis.

Attribute Type: Unsigned16

Allowed values: same values as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

List of Diagnosis Data

This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis

This parameter contains channel related diagnosis data and consists of the following elements.

Slot Number

This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

List of Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute Type: Unsigned16

Allowed values: 0-0x7FFF user specific channel number, 0x8000 submodule

Channel Properties

This attribute has the same definition as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

Channel Error Type

This attribute indicates the error type of the Channel Related Diagnosis.

Attribute Type: Unsigned16

Allowed values: same values as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

Ext Channel Diagnosis

This attribute contains an extended channel diagnosis. It consists of the following attributes.

Slot Number

This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

List of Ext Channel Diagnostic Data

This parameter contains a list of extended channel diagnosis data. A list element consists of the following parameter.

Channel Number

This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute Type: Unsigned16

Allowed values: 0-0x7FFF user specific channel number, 0x8000 submodule

Channel Properties

This attribute has the same definition as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

Channel Error Type

This attribute indicates the error type of the Channel Related Diagnosis.

Attribute Type: Unsigned16

Allowed values: same values as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

Ext Channel Error Type

This attribute depends on the attribute Channel Error Type and indicates an extended error type. The allowed values are shown in Table 27.

Attribute Type: Unsigned16

Ext Channel Add Value

This attribute contains an additional value.

Attribute Type: Unsigned32

Manufacturer Specific Diagnosis

This attribute contains a manufacturer specific diagnosis and consists of the following attributes.

Slot Number

This attribute defines the slot number as a local identifier. This number is unique within a device. The numbering of slots within a device shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical slots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines the subslot number as a local identifier. This number is unique within a slot. The numbering of subslots within a slot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical subslots which have diagnosis.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Channel Number

This attribute defines the channel number as a local identifier. This number is unique within a subslot. The numbering of channels within a subslot shall be in ascending order but may contain gaps in order to allow logical structuring. The list shall contain all physical or logical channels which have diagnosis.

The special value 0x8000 identifies the whole submodule instead of a special channel.

Attribute Type: Unsigned16

Allowed values: 0-0x7FFF user specific channel number, 0x8000 submodule

Channel Properties

This attribute has the same definition as specified in the Diagnosis ASE (see 4.3.2.4.2.1.2)

User Structure Identifier

This attribute indicates the structure of the Manufacturer Specific Diagnosis Data attribute.

Attribute Type: Unsigned16

Allowed values: MANUFACTURER_SPECIFIC_1 up to MANUFACTURER_SPECIFIC_0x7FFF, CHANNEL_DIAGNOSIS, MULTIPLE_DIAGNOSIS

List of Data

This attribute contains a list of manufacturer specific diagnosis data. A list element consists of the following attributes.

Data

This attribute contains manufacturer specific diagnosis data.

Attribute Type: One of the following data types shall be used:

Boolean, Binary Date, Integer, Time of Day, Unsigned, Time-Difference, Floating Point, Network Time, VisibleString, Network Time Difference, OctetString

Related AREP

This attribute contains the AREP of the related AR for transmission.

Attribute Type: Unsigned32

Alarm Priority

This attribute contains the priority of the alarm.

Allowed Values: ALARM_HIGH, ALARM_LOW

4.3.2.5.2.3 Invocation of the alarm data object

Several Alarm Data objects can be invoked in an IO device or an IO controller.

4.3.2.5.3 Alarm service specification**4.3.2.5.3.1 Alarm Notification**

This service is used to transfer an alarm notification from the IO device to the IO controller and optional vice versa. This service shall only be used in conjunction with the IO AR. Table 36 shows the parameters of the service.

Table 36 — Alarm Notification

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
API	M	M(=)	
Alarm Priority	M	M(=)	
Alarm Type	M	M(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Alarm Specifier	M	M(=)	
Module Ident Number	M	M(=)	
Submodule Ident Number	M	M(=)	
Alarm Item	U	U(=)	
User Structure Identifier	M	M(=)	
List of Data	S	S(=)	
Data	M	M(=)	
List of Channel Diagnosis Data	S	S(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error Type	M	M(=)	
List of Diagnosis Data	S	S(=)	
Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error Type	M	M(=)	
Ext Channel Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
List of Ext Channel Diagnosis Data	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
Channel Error Type	M	M(=)	

Parameter name	Req	Ind	Cnf
Ext Channel Error Type	M	M(=)	
Ext Channel Add Value	M	M(=)	
Manufacturer Specific Diagnosis	S	S(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Channel Number	M	M(=)	
Channel Properties	M	M(=)	
User Structure Identifier	M	M(=)	
List of Data	M	M(=)	
Data	M	M(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter is the identifier for the desired application process.

Alarm Priority

This parameter contains the value ALARM_HIGH or ALARM_LOW.

Alarm Type

This parameter contains the value of the attribute Alarm Type of the Alarm Data object.

Slot Number

This parameter contains the value of the attribute Slot Number of the Alarm Data object.

Subslot Number

This parameter contains the value of the attribute Subslot Number of the Alarm Data object.

Alarm Specifier

This parameter contains the value of the attribute Alarm Specifier of the Alarm Data object.

Module Ident Number

This parameter contains the value of the attribute Module Ident Number of the Alarm Data object.

Submodule Ident Number

This parameter contains the value of the attribute Submodule Ident Number of the Alarm Data object.

Alarm Item

This parameter contains alarm specific data if present. The length shall 1408 octets not exceed.

User Structure Identifier

This parameter contains the value of the corresponding attribute of the ASE object.

List of Data

This parameter contains the value of the corresponding attribute of the ASE object.

Data

This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnosis Data

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

List of Diagnosis Data

This parameter contains a list of diagnosis data. A list element consists of the following parameter.

Channel Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Ext Channel Diagnostic Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameter.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Error Type

This parameter contains the value of the corresponding attribute of the ASE object.

Ext Channel Add Value

This parameter contains the value of the corresponding attribute of the ASE object.

Manufacturer Specific Diagnosis

This parameter is composed of the following elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Number

This parameter contains the value of the corresponding attribute of the ASE object.

Channel Properties

This parameter contains the value of the corresponding attribute of the ASE object.

User Structure Identifier

This parameter contains the value of the corresponding attribute of the ASE object.

List of Data

This parameter contains a list of manufacturer specific diagnosis data. A list element consists of the following parameter.

Data

This parameter contains the value of the corresponding attribute of the ASE object.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Status

This parameter contains the reason for the failure.

Allowed values: AR_NOT_ESTABLISHED, ALARM_TYPE_NOT_SUPPORTED, LIMIT_EXPIRED, SEQUENCE_NR_PENDING

4.3.2.5.3.2 Alarm Ack

This service is used to acknowledge the reception of an alarm notification which has been previously received. This service shall only be used in conjunction with the IO or Supervisor AR.

Table 37 shows the parameters of the service.

Table 37 — Alarm Ack

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
API	M	M(=)	
Alarm Type	M	M(=)	
Slot Number	M	M(=)	
Subslot Number	M	M(=)	
Alarm Specifier	M	M(=)	
PNIO Status	M	M(=)	
Result(+)			S
AREP			M(=)
Result(-)			S
AREP			M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter is the identifier for the desired application process.

Alarm Type

This parameter contains the value of the attribute Alarm Type of the previously received Alarm Data object.

Slot Number

This parameter contains the value of the attribute Slot Number of the of the previously received Alarm Data object.

Subslot Number

This parameter contains the value of the attribute Subslot Number of the of the previously received Alarm Data object.

Alarm Specifier

This parameter contains the value of the attribute Alarm Specifier of the of the previously received Alarm Data object.

PNIO Status

This parameter contains the error status.

Allowed Values: NO_ERROR, ALARM_TYPE_NOT_SUPPORTED,
WRONG_SUBMODULE_STATE

Attribute Type: Unsigned32

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

4.3.2.5.4 Behavior of Alarm data objects**4.3.2.5.4.1 General behavior of the alarm sink (e.g. IO controller application)**

The application behavior based on flow control of alarm transmission, IOPS, IOCS and Problem Indicator of the IO Data ASE. These mechanisms make it convenient for the application to process the alarm.

The application process may not handle the alarm immediately by interrupting other processes. It may also poll for queued alarms to handle these in an appropriate time window.

Therefore, each application process should maintain a receiving queue for each Alarm Type. The queue size is determined by the flow control mechanisms. With the processing of the alarm the acknowledgment shall be sent to the source to enable the sending of further alarms of this type.

Furthermore, alarm queues shall be cleared in case of AR termination and pull/plug alarms.

4.3.2.5.4.2 General behavior of the alarm source (e.g. IO device application)

Figure 25 shows the model at the alarm source. The source is divided in Figure 25 into the IO device (IOD), the application process and the alarm source.

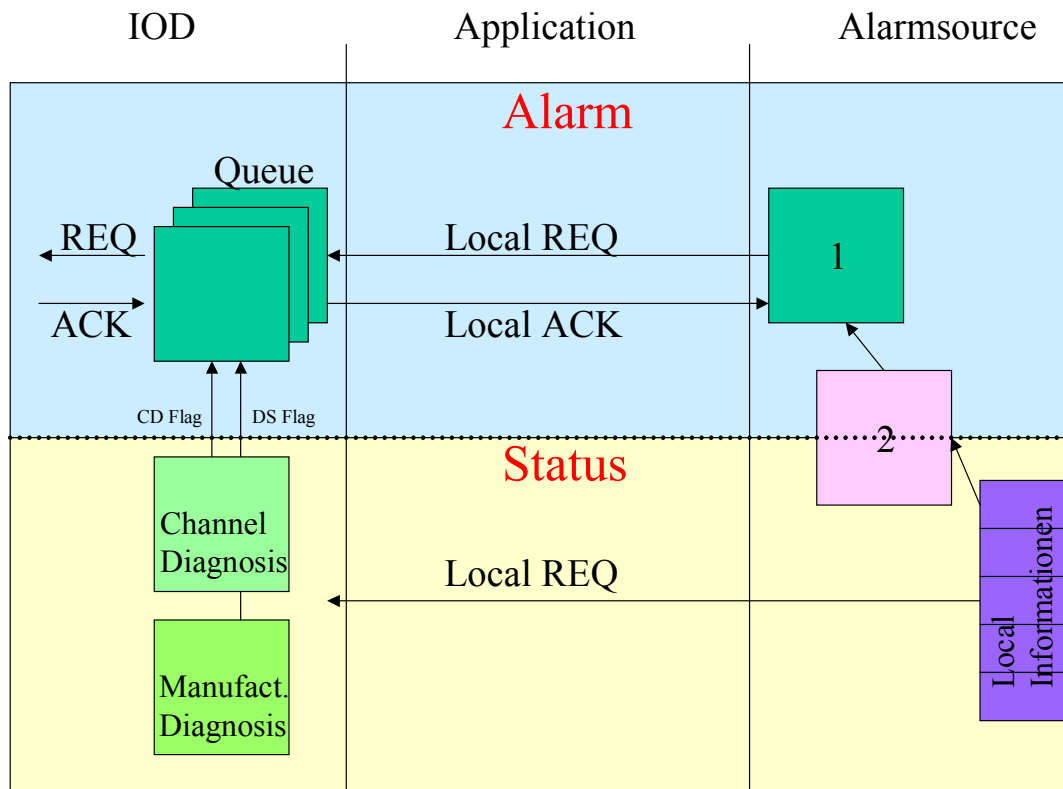


Figure 25 — Example of a resource model at the alarm source

The model shows how to handle event overflow at the source. Event overflow may cause by flow control because the sink determines how fast alarms are handled. A mixture of queue and buffer model solves that problem for the Alarm ASE.

The alarm source maintains local information of the state. If, for example, the status changes from GOOD to BAD it is notified. It maintains at least two buffers for alarms. The first buffer is used to transport the state transition. The second buffer contains the current state. A new state transition will not be reported as long as the acknowledgement is not received in case of only two buffers. But the current state may be read with diagnosis information at any time.

The IOD maintains two queues for each priority. A new alarm of the same priority is blocked as long as the acknowledgement is outstanding.

Furthermore, alarm queues shall be cleared in case of AR termination and pull/plug alarms.

4.3.2.5.4.3 Behavior for diagnosis changes

Diagnosis alarms are the means to report and acknowledge diagnosis changes. It is strongly recommended to use Channel Diagnoses because of the well-defined structure.

Diagnosis alarms include information of the alarm source and the direction of changes. The attribute Alarm Specifier contains information about the direction e.g. appears or disappears.

All channel diagnosis are gone is reported by an alarm with the parameter Alarm Specifier Submodule Diagnosis State "NO_ERROR". It may save a lot of individual messages.

4.3.2.6 Context ASE

4.3.2.6.1 Overview

The Context ASE provides a set of services to establish application relationships, parameterize and configure the AP of an IO device itself and its slots/subslots containing IO data.

The Application Process of IO devices may need parameter (e.g. the range of an analogue signal) from the assigned IO controller to provide the data from the Input Data object and/or for the Output Data object according to the requirements of the AP of the IO controller. These IO devices provide in the Context ASE an IO AR Parameter object or, for more flexibility, several IO AR Structured Parameter objects. The values for the attributes of these objects will be provided by the IO controller and assigned after validation from the AP of the IO device.

The functionality of an IO device application implies the necessary parameterization objects (specific Record Data objects) which have to be conveyed at the start-up phase of the PROFINET IO system from the IO controller to the assigned IO devices via the IO AR. The application of the IO device has to check the parameterization data sent by the IO controller.

NOTE In comparison to the IO Data ASE, where all the actual available slots and subslots with their attributes are stored, the Context ASE contains only the configured or used slots and subslots after a successful connection establishment.

4.3.2.6.2 Context class specification

4.3.2.6.2.1 IO device context class specification

4.3.2.6.2.1.1 Template

This class specifies the object for the parameter data that are related to the IO devices itself and/or to the slots/subslots of the IO device.

The IO Device context object is described by the following template:

DP ASE:	Context ASE
CLASS:	IO Device Context
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of APs
2.1	(m) Attribute: API
2.2	(m) Attribute: Real Identification
2.2.1	(m) Attribute: List of Slots
2.2.1.1	(m) Attribute: Slot Number
2.2.1.2	(m) Attribute: Module Ident Number
2.2.1.3	(m) Attribute: Module Properties
2.2.1.4	(m) Attribute: List of Subslots
2.2.1.4.1	(m) Attribute: Subslot Number
2.2.1.4.2	(m) Attribute: Submodule Ident Number
2.2.1.4.3	(m) Attribute: Submodule Properties
2.2.1.4.3.1	(m) Attribute: Type
2.2.1.4.3.2	(m) Attribute: Shared Input
2.2.1.4.4	(m) Attribute: List of Data Descriptions
2.2.1.4.4.1	(m) Attribute: Data Direction
2.2.1.4.4.2	(m) Attribute: Submodule Data Length
2.2.1.4.4.3	(m) Attribute: Length IOPS
2.2.1.4.4.4	(m) Attribute: Length IOCS
2.3	(m) Attribute: Expected Identification
2.3.1	(m) Attribute: List of ARs
2.3.1.1	(m) Attribute: AREP
2.3.1.2	(o) Attribute: Sync State
2.3.1.3	(m) Attribute: List of CRs
2.3.1.3.1	(m) Attribute: CREP
2.3.1.3.2	(m) Attribute: Multicast Provider State
2.3.1.4	(m) Attribute: List of APIs
2.3.1.4.1	(m) Attribute: API
2.3.1.4.2	(m) Attribute: List of Slots

2.3.1.4.2.1	(m) Attribute:	Slot Number
2.3.1.4.2.2	(m) Attribute:	Module Ident Number
2.3.1.4.2.3	(m) Attribute:	Module Properties
2.3.1.4.2.4	(m) Attribute:	Module State
2.3.1.4.2.5	(m) Attribute:	List of Subslots
2.3.1.4.2.5.1	(m) Attribute:	Subslot Number
2.3.1.4.2.5.2	(m) Attribute:	Submodule Ident Number
2.3.1.4.2.5.3	(m) Attribute:	Submodule Properties
2.3.1.4.2.5.3.1	(m) Attribute:	Type
2.3.1.4.2.5.3.2	(m) Attribute:	Shared Input
2.3.1.4.2.5.4	(m) Attribute:	Submodule State
2.3.1.4.2.5.4.1	(m) Attribute:	Format Indicator
2.3.1.4.2.5.4.2	(c) Constraint:	Format Indicator = V_1_0
2.3.1.4.2.5.4.2.1	(m) Attribute:	Detail
2.3.1.4.2.5.4.3	(c) Constraint:	Format Indicator <> V_1_0
2.3.1.4.2.5.4.3.1	(m) Attribute:	Add Info
2.3.1.4.2.5.4.3.2	(m) Attribute:	Diag Info
2.3.1.4.2.5.4.3.3	(m) Attribute:	AR Info
2.3.1.4.2.5.4.3.4	(m) Attribute:	Ident Info

SERVICES:

1	(m) OpsService:	Connect
2	(m) OpsService:	Release
3	(m) OpsService:	End Of Parameter
4	(m) OpsService:	Application Ready
5	(m) OpsService:	Read Expected Identification Data
6	(m) OpsService:	Read Real Identification Data
7	(m) OpsService:	Read Identification Difference
8	(m) OpsService:	Abort

4.3.2.6.2.1.2 Attributes

Implicit

The attribute Implicit indicates that the User Parameter object is implicitly addressed by the services.

List of APs

One API is composed of the following list elements:

API

This attribute defines the number of the application process.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Real Identification

This attribute consists of the following attributes:

List of Slots

This attribute defines the slots of the application process. One Slot is composed of the following list elements:

Slot Number

This attribute defines to which module the IO Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 7FFF

Module Ident Number

This attribute defines the identification of the slot.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE The attribute value is manufacturer specific.

Module Properties

This attribute is reserved for future use.

Module State

This attribute defines the state of the slot.

Attribute Type: Unsigned16

Allowed values: Shall be set according to the Table 38.

Table 38 — Module State

Value	Use
NO_MODULE	The slot exists and the module is not plugged.
WRONG_MODULE	The module ident number is wrong.
PROPER_MODULE	The slot exists and the module is okay but at least one submodule is locked, wrong or missing.
SUBSTITUTE	The slot exists and the module is not the same as requested – but compatible. The IO device was able to adapt by its own decision.
GOOD	The slot exists and the expected module is plugged.

List of Subslots

This attribute consists of the following attributes.

Subslot Number

This attribute defines to which subslot the IO Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

Submodule Ident Number

This attribute defines the identification of the subslot.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE The attribute value is manufacturer specific.

Submodule Properties

This attribute consists of the following attributes.

Type

This attribute defines the type of the subslot.

Attribute Type: Unsigned16

Allowed values: NO_IO, INPUT, OUTPUT, IO

Shared Input

This attribute defines the properties of the subslot. The value SHARED_INPUT shall only used if the attribute Submodule Type contains the value INPUT or IO.

Attribute Type: Unsigned8

Allowed values: SHARED_INPUT,
NOT_SHARED_INPUT

List of Data Descriptions

This attribute list consists of the following attributes.

Data Direction

The attribute defines the direction input or output.

Attribute Type: Unsigned16

Allowed values: DIRECTION_INPUT, DIRECTON_OUTPUT

Submodule Data Length

The attribute Submodule Data Length defines the number of octets of the Data Element without counting the IOPS.

Attribute Type: Unsigned16

Allowed values: 0 to 1439

Length IOPS

The attribute Length IOPS defines the number of octets of the provider status.

Attribute Type: Unsigned8

Allowed values: 1, (2 to 255 reserved)

Length IOCS

The attribute Length IOCS defines the number of octets of the consumer status.

Attribute Type: Unsigned8

Allowed values: 1, (2 to 255 reserved)

Expected Identification

This attribute consists of the following attributes:

List of ARs

One Slot is composed of the following list elements:

AREP

This parameter is the local identifier for the desired AR.

SYNC State

This attribute is the local synchronization status for the desired AR. It shall only be present if synchronization is used. The behavior shall be according to 4.3.2.6.5.

Attribute Type: Unsigned16

Allowed values: SYNCHRONIZED, SYNC_NOT_AVAILABLE

List of CRs

One CR is composed of the following list elements:

CREP

This parameter is the local identifier for the desired multicast consumer CR.

Multicast Provider State

This parameter is the local status for the multicast consumer CR. The behavior shall be according to 4.3.2.6.5.

Attribute Type: Unsigned16

Allowed values: UP_AND_RUNNING,
MULTICAST_PROVIDER_NOT_AVAILABLE

List of APIs

One API is composed of the following list elements:

API

This attribute defines the number of the application process.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

List of Slots

One Slot is composed of the following list elements:

Slot Number

This attribute defines to which module the IO Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Module Ident Number

This attribute defines the identification of the slot.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE The attribute value is manufacturer specific.

Module Properties

This attribute is reserved for further use.

List of Subslots

This attribute consists of the following attributes.

Subslot Number

This attribute defines to which subslot the IO Data Element belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF, for Slot Number 0 additionally 0x8000 to 0x8FFF are allowed

Submodule Ident Number

This attribute defines the identification of the subslot.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE The attribute value is manufacturer specific.

Submodule Properties

This attribute consists of the following attributes.

Type

This attribute defines the type of the subslot.

Attribute Type: Unsigned16

Allowed values: NO_IO, INPUT, OUTPUT, IO

In conjunction with selected CR attribute CR Type the following dependencies shall be applied:

Table 39 — Usage with respect to CR Type

Submodule Type	Usage with CR Type
NO_IO	INPUT CR, OUTPUT CR
INPUT	INPUT CR, OUTPUT CR, MULTICAST PROVIDER CR
OUTPUT	INPUT CR, OUTPUT CR, MULTICAST CONSUMER CR
IO	INPUT CR, OUTPUT CR, MULTICAST PROVIDER CR (Input part only), MULTICAST CONSUMER CR (Output part only)

Shared Input

This attribute defines the properties of the subslot. The value SHARED_INPUT shall only used if the attribute Submodule Type contains the value INPUT or IO.

Attribute Type: Unsigned8

Allowed values: SHARED_INPUT, NOT_SHARED_INPUT

Submodule State

This attribute consists of the following attributes:

Format Indicator

This attribute indicates the format of Submodule State

Detail

This attribute contains the value for the submodule state compatible to version 1.0 of PROFINET IO.

Shall be set according to the Table 40.

Table 40 — Detail

Value	Meaning
NO_SUBMODULE	The subslot exists and is empty.
WRONG_SUBMODULE	The subslot exists and an incompatible submodule is plugged.
LOCKED_BY_IO_CONTROLLER	The subslot exists and is locked by another IO controller.
APPLICATION_READY_PENDING	The subslot exists and a parameter fault was detected or the application of the subslot is not operating

Value	Meaning
SUBSTITUTE	The subslot exists and the submodule is not the same as requested – but compatible and the IO device was able to adapt In this case the IO device has to adapt e.g. to new input or output length
GOOD	The subslot exists and the expected submodule is plugged.

NOTE The field Submodule State is only responded if there is a difference between expected and real configuration data. Submodule State “GOOD” is not defined because for such submodules no status is reported within the response.

Add Info

This attribute contains a value for the IO supervisor in case the takeover was not allowed.

Allowed Values: NO_ADD_INFO, TAKEOVER_NOT_ALLOWED

Diag Info

This attribute signals the presence of diagnosis information when application ready is generated.

Allowed Values: NO_DIAG, DIAG

AR Info

This attribute contains the submodule state in relation to the established AR.

Shall be set according to the Table 41.

Table 41 — ARInfo

Value	Meaning
OWN	This AR is owner of the submodule
APPLICATION_READY_PENDING	This AR is owner of the submodule but it is blocked. I.e. parameter checking pending
SUPERORDINATED_LOCKED	This AR is not owner of the submodule. It is blocked by superordinated means
LOCKED_BY_IO_CONTROLLER	This AR is not owner of the submodule. It is owned by an other IOAR
LOCKED_BY_IO_SUPERVISOR	This AR is not owner of the submodule. It is owned by an other IOSAR

Ident Info

This attribute contains the submodule state in respect to expected and real identification for an established AR.

Shall be set according to the Table 42.

Table 42 — Ident Info

Value	Meaning
OK	no difference
SUBSTITUTE	different but compatible
WRONG	incompatible
NO	empty

4.3.2.6.2.1.3 Invocation of the IO device context object

Each IO Device shall invoke one IO Device Context object.

4.3.2.6.3 Context service specification

4.3.2.6.3.1 Connect

The Connect service is used to establish an IO AR or Supervisor AR. The application process shall check the service parameter of the request/indication with the local parameter in its ASE objects (Real Identification attributes). In case of a service parameter mismatch (e.g. submodules are already used by other IO ARs) with the object attributes the application process shall respond the mismatching modules within response. The client may take the

appropriate actions. If the establishment phase was successfully the service parameter are stored as attribute values in the Expected Identification attribute group of the IO Device Context object.

Table 43 shows the parameter of the service.

NOTE The service parameter IP Address and Object UUID are necessary to address the destination device. The underlying model assumes that this information is provided by the AR information, which has to be loaded before on behalf of the client application. On server side this information is also handled inside the application layer and does not appear as an explicit service parameter. Local implementation may pass this service parameter to the application process. The same model applies to Read services over the implicit AR.

The application of the server receiving the Connect indication service primitive shall check the service parameter. Especially, the parameter Module Ident Number, Submodule Ident Number, Submodule Data Length, IOCS Length, and IOPS Length shall only be checked by the user application.

Table 43 — Connect

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M	M(=)	M(=)
AR Parameter Block	M	M(=)		
AR Type	M	M(=)		
AR UUID	M	M(=)		
Session Key	M	M(=)		
CM Initiator Mac Add	M	M(=)		
CM Initiator Object UUID	M	M(=)		
AR Properties	M	M(=)		
CM Initiator Activity Timeout Factor	M	M(=)		
UDP CM Initiator Port	M	M(=)		
CM Initiator Station Name	M	M(=)		
List of IO CR Parameter Blocks	M	M(=)		
IO CR Type	M	M(=)		
IO CR Reference	M	M(=)		
LT Field	M	M(=)		
IO CR Properties	M	M(=)		
C_SDU Length	M	M(=)		
Frame ID	M	M(=)		
Send Clock Factor	M	M(=)		
Reduction Ratio	M	M(=)		
Phase	M	M(=)		
Sequence	M	M(=)		
Frame Send Offset	M	M(=)		
Watchdog Factor	M	M(=)		
Data Hold Factor	M	M(=)		
IO CR Tag Header	M	M(=)		
IO CR Multicast MAC Add	M	M(=)		
List Of APIs	M	M(=)		
API	M	M(=)		
List of Related IO Data Objects	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
IO Data Object Frame Offset	M	M(=)		
List of Related IOCS	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
IOCS Frame Offset	M	M(=)		
Alarm CR Parameter Block	M	M(=)		
Alarm CR Type	M	M(=)		
LT Field	M	M(=)		
Alarm CR Properties	M	M(=)		
RTA Timeout Factor	M	M(=)		
RTA Retries	M	M(=)		
Local Alarm Reference	M	M(=)		
Max Alarm Data Length	M	M(=)		

Parameter name	Req	Ind	Rsp	Cnf
Alarm CR Tag Header High	M	M(=)		
Alarm CR Tag Header Low	M	M(=)		
List of Expected Submodule Blocks	U	U(=)		
List of Related APIs	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Module Ident Number	M	M(=)		
Module Properties	M	M(=)		
List of Submodules	M	M(=)		
Subslot Number	M	M(=)		
Submodule Ident Number	M	M(=)		
Submodule Properties	M	M(=)		
List of Data Descriptions	M	M(=)		
Data Description	M	M(=)		
Submodule Data Length	M	M(=)		
Length IOPS	M	M(=)		
Length IOCS	M	M(=)		
Parameter Server Block	U	U(=)		
Parameter Server Object UUID	M	M(=)		
Parameter Server Properties	M	M(=)		
CM Initiator Activity Timeout Factor	M	M(=)		
Parameter Server Station Name	M	M(=)		
List of Multicast CR Blocks	U	U(=)		
IOCR Reference	M	M(=)		
Address Resolution Properties	M	M(=)		
MCI Timeout Factor	M	M(=)		
Provider Station Name	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
AR Response Block			M	M(=)
AR Type			M	M(=)
AR UUID			M	M(=)
Session Key			M	M(=)
CM Responder MAC Add			M	M(=)
UDP Remote Port			M	M(=)
List of IO CR Response Blocks			M	M(=)
IOCR Type			M	M(=)
IOCR Reference			M	M(=)
Frame ID			M	M(=)
Alarm CR Response Block			M	M(=)
Alarm CR Type			M	M(=)
Local Alarm Reference			M	M(=)
Max Alarm Data Length			M	M(=)
Module Diff Block			U	U(=)
List of APIs			M	M(=)
API			M	M(=)
List of Modules			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
Module State			M	M(=)
List of Submodules			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Submodule State			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

AR Parameter Block

The parameter contains of the following attributes:

AR Type

This parameter contains the value for the equivalent attribute of the ARL entry.

AR UUID

This parameter contains the value for the equivalent attribute of the ARL entry.

Session Key

The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute Type: Unsigned16

CM Initiator MAC Add

This parameter contains the value for the equivalent attribute of the ARL entry.

IO Initiator Object UUID

This parameter contains the value for the equivalent attribute of the ARL entry.

AR Properties

This parameter contains the value for the equivalent attribute of the ARL entry.

CM Initiator Activity Timeout Factor

This parameter contains the value for the equivalent attribute of the ARL entry.

UDP CM Initiator Port

This parameter contains the value for the equivalent attribute of the ARL entry.

CM Initiator Station Name

This parameter contains the value for the equivalent attribute of the ARL entry.

List of IO CR Parameter Blocks

The parameter list contains the following attributes:

IO CR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

IO CR Reference

This parameter contains the value to identify the CR within this service.

NOTE It is used within the context of the connect service to link the IO Data elements with its CRs. In the local context of the IO device it is transferred into the CREP.

LT Field

This parameter contains the value for the equivalent attribute of the CRL entry.

IO CR Properties

This parameter contains the value for the equivalent attribute of the CRL entry.

C_SDU Length

This parameter contains the value for the equivalent attribute of the CRL entry.

Frame ID

This parameter contains the value for the equivalent attribute of the CRL entry.

Send Clock Factor

This parameter contains the value for the equivalent attribute of the CRL entry.

Reduction Ratio

This parameter contains the value for the equivalent attribute of the CRL entry.

Phase

This parameter contains the value for the equivalent attribute of the CRL entry.

Sequence

This parameter contains the value for the equivalent attribute of the CRL entry.

Frame Send Offset

This parameter contains the value for the equivalent attribute of the CRL entry.

Watchdog Factor

This parameter contains the value for the equivalent attribute of the CRL entry.

Data Hold Factor

This parameter contains the value for the equivalent attribute of the CRL entry.

IO CR Tag Header

This parameter contains the value for the equivalent attribute of the CRL entry.

IO CR Multicast MAC Add

This parameter contains the value for the equivalent attribute of the CRL entry.

List Of APIs

This parameter contains the value for the equivalent attribute of the CRL entry.

API

This parameter contains the value for the equivalent attribute of the CRL entry.

List of Related IO Data Objects

The parameter list shall always be present. It contains the slots and subslots where read and write access is allowed. The application process of the server shall respond to write services addressed to objects beyond this negotiated address space with a negative response. Read access is always allowed to other objects. The following parameter shall be checked with the corresponding attributes of the IO Data object and the result in case of a positive response shall be stored as attribute values of the Context object. The parameter consists of the following sub-parameter:

Slot Number

This parameter contains the value for the equivalent attribute of the Context ASE.

Sublot Number

This parameter contains the value for the equivalent attribute of the Context ASE.

IO Data Object Frame Offset

This parameter contains the value for the equivalent attribute of the Context ASE.

List of Related IOCS

This parameter contains the value for the equivalent attribute of the Context ASE.

Slot Number

This parameter contains the value for the equivalent attribute of the Context ASE.

Sublot Number

This parameter contains the value for the equivalent attribute of the Context ASE.

IOCS Frame Offset

This parameter contains the value for the equivalent attribute of the Context ASE.

Alarm CR Parameter Block

This parameter consists of the following attributes:

Alarm CR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

LT Field

This parameter contains the value for the equivalent attribute of the CRL entry.

Alarm CR Properties

This parameter contains the value for the equivalent attribute of the CRL entry.

RTA Timeout Factor

This parameter contains the value for the equivalent attribute of the CRL entry.

RTA Retries

This parameter contains the value for the equivalent attribute of the CRL entry.

Local Alarm Reference

This parameter contains the value for the equivalent attribute of the CRL entry.

Max Alarm Data Length

This parameter contains the value for the equivalent attribute of the CRL entry.

Alarm CR Tag Header High

This parameter contains the value for the equivalent attribute of the CRL entry.

Alarm CR Tag Header Low

This parameter contains the value for the equivalent attribute of the CRL entry.

List of Expected Submodule Blocks

This parameter list consists of the following parameter:

List Of Related APIs

This parameter contains the value for the equivalent attribute of the CRL entry.

API

This parameter contains the value for the equivalent attribute of the CRL entry.

Slot Number

This parameter contains the value for the equivalent attribute of the Context ASE.

Module Ident Number

This parameter contains the value for the equivalent attribute of the Context ASE.

Module Properties

This parameter contains the value for the equivalent attribute of the Context ASE.

List of Submodules

This list consists of the following parameter:

Subslot Number

This parameter contains the value for the equivalent attribute of the Context ASE.

Submodule Ident Number

This parameter contains the value for the equivalent attribute of the Context ASE.

Submodule Properties

This parameter contains the value for the equivalent attribute of the Context ASE.

List of Data Descriptions

This list consists of the following attributes:

Data Description

This parameter contains the value for the equivalent attribute of the Context ASE.

Submodule Data Length

This parameter contains the value for the equivalent attribute of the Context ASE.

Length IOPS

This parameter contains the value for the equivalent attribute of the Context ASE.

Length IOCS

This parameter contains the value for the equivalent attribute of the Context ASE.

Parameter Server Block

This parameter consists of the following attributes:

Parameter Server Object UUID

This parameter contains the value for the equivalent attribute of the ASE.

Parameter Server Properties

This parameter contains the value for the equivalent attribute of the ASE.

CM Initiator Activity Timeout Factor

This parameter contains the value for the equivalent attribute of the ASE.

Parameter Server Station Name

This parameter contains the value for the equivalent attribute of the ASE.

List of Multicast CR Block

This parameter consists of the following attributes:

IOCR Reference

This parameter contains the value for the equivalent attribute of the AR ASE.

Address Resolution Properties

This parameter contains the value for the equivalent attribute of the AR ASE.

MCI Timeout Factor

This parameter contains the value for the equivalent attribute of the AR ASE.

Provider Station Name

This parameter contains the value for the equivalent attribute of the AR ASE.

Result(+)

This parameter indicates that the service request succeeded. It shall only contain IO Data Response Block parameter for entries that differ from the requested parameter values.

AR Response Block

The selectable parameter contains of the following attributes:

AR Type

This parameter contains the value for the equivalent attribute of the ARL entry.

AR UUID

This parameter contains the value for the equivalent attribute of the ARL entry.

Session Key

The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute Type: Unsigned16

CM Responder MAC Add

This parameter contains the value for the equivalent attribute of the ARL entry.

UDP Remote Port

This parameter contains the value for the equivalent attribute of the ARL entry.

List of IO CR Response Blocks

The parameter list contains the following attributes:

IOCR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

IOCR Reference

This parameter contains the value mirrored from the request.

Frame ID

This parameter contains the value for the equivalent attribute of the CRL entry.

Alarm CR Response Block

This parameter consists of the following attributes:

Alarm CR Type

This parameter contains the value for the equivalent attribute of the CRL entry.

Local Alarm Reference

This parameter contains the value for the equivalent attribute of the CRL entry.

Max Alarm Data Length

This parameter contains the value for the equivalent attribute of the CRL entry.

Module Diff Block

The parameter consists of the following sub-parameter for values which differ from the requested ones:

List of APIs

This parameter contains the APIs which differ from the requested configuration.

API

This parameter contains the value for the equivalent attribute of the ASE.

List of Modules

This parameter contains the modules which differ from the requested configuration.

Slot Number

This parameter contains the value for the equivalent attribute of the ASE.

Module Ident Number

This parameter contains the value for the equivalent attribute of the ASE.

Module State

This parameter contains the value for the equivalent attribute of the ASE.

List of Submodules

This parameter list consists of the following parameter:

Subslot Number

This parameter contains the value for the equivalent attribute of the ASE.

Submodule Ident Number

This parameter contains the value for the equivalent attribute of the ASE.

Submodule State

This parameter contains the value for the equivalent attribute of the ASE.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter shall have the value PNIO.

Type: Unsigned8

Error Code 1

The Error Code 1 assumes one of the following values:

FAULTY_CONNECT_BLOCK.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

4.3.2.6.3.2 Release

This service shall be used by an IO controller or IO supervisor to disconnect an IO or Supervisor AR.

Table 44 shows the parameter of the service.

Table 44 — Release

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Session Key			M(=)	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Session Key

The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter shall contain the value PNIO.

Type: Unsigned8

Error Code 1

The Error Code 1 assumes one of the following values:

FAULTY_RELEASE_BLOCK

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

4.3.2.6.3.3 Abort

This service shall be used by an IO device to close an IO or Supervisor AR in case of a serious error when operation is not longer possible.

Table 45 shows the parameter of the service.

Table 45 — Abort

Parameter name	Req	Ind	Cnf
Argument	M	M	M(=)
AREP	M	M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

4.3.2.6.3.4 End Of Parameter

This service shall be used by the client (IO controller or IO Parameter server) to inform the IO device that all parameter are written by means of the Write service of the Record Data ASE.

Table 46 shows the parameter of the service.

Table 46 — End Of Parameter

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
Alarm Sequence Number	U	U(=)		
Result(+)			S	S(=)
AREP			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Session Key			M	M(=)
Alarm Sequence Number			U	U(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Session Key

The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Alarm Sequence Number

This parameter shall only be used in conjunction with a plug or pull alarm. The value shall be taken from the Alarm Notification.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter shall have the value PNIO.

Type: Unsigned8

Error Code 1

The Error Code 1 assumes one of the following values:
FAULTY_END_OF_PARAMETER_BLOCK

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

4.3.2.6.3.5 Application Ready

By this service the User of an IO device indicates to the IO controller that the application is ready. This service shall only be used in conjunction with the IO AR.

Table 47 shows the parameters of the service.

Table 47 — Application Ready

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
Alarm Sequence Number	U	U(=)		
Module Diff Block	U	U(=)		
List of APIs	M	M(=)		
API	M	M(=)		
List of Modules	M	M(=)		
Slot Number	M	M(=)		
Module Ident Number	M	M(=)		
Module State	M	M(=)		

Parameter name	Req	Ind	Rsp	Cnf
List of Submodules	M	M(=)		
Subslot Number	M	M(=)		
Submodule Ident Number	M	M(=)		
Submodule State	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Session Key			M	M(=)
Alarm Sequence Number			U	U(=)
Result(-)			S	S(=)
AREP			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Session Key

The value of the parameter Session Key shall be increased by one for each connect by the CM Initiator. The CM Responder shall use this value for each subsequent response within this session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Alarm Sequence Number

This parameter shall only be used in conjunction with a plug or pull alarm. The value shall be taken from the Alarm Notification.

Module Diff Block

This parameter shall only be if differences exists. The parameter consists of the following sub-parameter for values which differ from the requested one's:

List of APIs

This parameter contains a list of APIs for the modules which differ from the requested configuration. A list element consists of the following parameter.

API

This parameter contains the value for the equivalent attribute of the ASE.

List of Modules

This parameter contains a list of modules which differ from the requested configuration. A list element consists of the following parameter.

Slot Number

This parameter contains the value for the equivalent attribute of the ASE.

Module Ident Number

This parameter contains the value for the equivalent attribute of the ASE.

Module State

This parameter contains the value for the equivalent attribute of the ASE.

List of Submodules

This parameter contains a list of submodules which differ from the requested configuration. A list element consists of the following parameter.

Subslot Number

This parameter contains the value for the equivalent attribute of the ASE.

Submodule Ident Number

This parameter contains the value for the equivalent attribute of the ASE.

Submodule State

This parameter contains the value for the equivalent attribute of the ASE.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter shall have the value PNIO.

Type: Unsigned8

Error Code 1

The Error Code 1 assumes one of the following values:

FAULTY_APPLICATION_READY_BLOCK

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

4.3.2.6.3.6 Read Expected Identification

This confirmed service may be used to read the value of the attributes which contain the configured module and submodule identification related to all ARs of the device. The expected identification is configured by project planning. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 48 shows the parameters of the service.

NOTE Using this service the data length may exceed the mandatory limit of the supported data length. In this case the service returns a negative response. In this case the slot specific service can be used to reduce the amount of data.

By means of the service parameter Target AR UUID, Slot, and Subslot the content of expected identification can be restricted to a specific AR, a specific Slot or a specific Subslot as a filter function. In case a user specific filter is selected the Read Expected Identification response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain Expected Identification Data if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 48 — Read Expected Identification

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This conditional parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific expected identification information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE It is used to read only diagnosis information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only diagnosis information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address only diagnosis information for a specific slot/subslot. This service parameter shall only be used together with the service parameter Slot Number.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Expected Identification Data that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

Result(+)

This parameter indicates that the service request succeeded.

List of Slots

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number

This parameter contains the value corresponding attribute of the ASE object.

List of Subslots

This parameter is composed of the following list elements:

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number

This parameter contains the value corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.6.3.7 Read Real Identification

This confirmed service may be used to read the value of the attributes which contain the module and submodule identification from the perspective of the IO device. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 49 shows the parameters of the service.

By means of the service parameter API, Target AR UUID, Slot, and Subslot the content of expected identification can be restricted to a specific AR, a specific Slot or a specific Subslot as a filter function. In case a user specific filter is selected the Read Real Identification response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

NOTE If a module or submodule is pulled it's entry within the real identification is removed.

Implicit AR using Target AR UUID: This service does only contain Real Identification Data if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Implicit AR using API: This service does only contain Real Identification Data if the requested API exist. Otherwise a parameter error is responded.

Table 49 — Read Real Identification

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)

Parameter name	Req	Ind	Rsp	Cnf
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of Slots			M	M(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific expected identification information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE It is used to read only diagnosis information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only diagnosis information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used is used to address only diagnosis information for a specific slot/subslot. This service parameter shall only be used together with the service parameter Slot Number.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Real Identification Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

List of Slots

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number

This parameter contains the value corresponding attribute of the ASE object.

List of Subslots

This parameter is composed of the following list elements:

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number

This parameter contains the value corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.6.3.8 Read Identification Difference

This confirmed service may be used to read the difference of the configured module and submodule identification related to the expected configuration. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 50 shows the parameters of the service.

NOTE Using this service the data length may exceed the mandatory limit of the supported data length. In this case the service returns a negative response. In this case the slot specific service can be used to reduce the amount of data.

By means of the service parameter Target AR UUID the content of identification difference is restricted to a specific AR.

Implicit AR using Target AR UUID: This service does only contain Identification Difference if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

If there exists no difference the Read Identification Difference service shall respond no data with Length zero.

Table 50 — Read Identification Difference

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Target AR UUID	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of APIs			U	U(=)
API			U	U(=)
List of Slots			U	U(=)
Slot Number			M	M(=)
Module Ident Number			M	M(=)
Module State			M	M(=)
List of Subslots			M	M(=)
Subslot Number			M	M(=)
Submodule Ident Number			M	M(=)
Submodule State			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Target AR UUID

This parameter shall only be used to read AR specific expected identification information. If this parameter is used the service parameter Slot Number and Subslot Number shall not be used.

NOTE It is used to read only diagnosis information the requested AR is connected to.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Expected Identification Data that has to be read. The allowed length is from 0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

List of APIs

This parameter consists of the following elements.

API

This parameter contains the value for the equivalent attribute of the ASE.

List of Slots

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Module Ident Number

This parameter contains the value corresponding attribute of the ASE object.

Module State

This parameter contains the value corresponding attribute of the ASE object.

List of Subslots

This parameter is composed of the following list elements:

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Submodule Ident Number

This parameter contains the value corresponding attribute of the ASE object.

Submodule State

This parameter contains the value corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.6.4 Behavior of the context object

The Context object contains the real identification data from the perspective of the IO device. The real identification is detected by local means before the IO device may communicate. This real identification may be changed during operation by a local means. It may effect adding or removing of slots with their subslots. In such a case, the real identification shall be updated and the IO controllers or IO supervisor connected via affected ARs shall be informed by means of an Alarm Notification service. The Alarm Types “Released By Supervisor”, “Controlled By Supervisor”, “Pull Alarm”, “Plug Alarm” or “Plug Wrong Submodule Alarm” shall be used to indicate such events.

An affected AR means an established relationship where at least one of the changed slot/subslot is part of the attribute Expected identification.

When receiving the Alarm Notification with “Plug Alarm” the IO controller shall check if the new added slot/subslot need additional parameter and convey this parameter to the IO device. The IO controller may also inform the local user.

When receiving the Alarm Notification with “Pull Alarm” or “Plug Wrong Submodule” the IO controller may inform the local user and wait for further actions.

However, in the time between a Connect indication and sending an Application Ready request, the real identification shall be kept consistent in any case. It means if “real” modules are plugged or removed within this time interval then it shall be reported afterwards by means of Alarm Notifications. It is not allowed to respond inconsistent or not plausible Module Diff Blocks with the Connect response or Application Ready request service primitives.

NOTE The following example illustrates an inconsistent Module Diff Block. If a submodule state was reported wrong in the Connect response it could not be healthy in the following Application Ready request, even if it was plugged in between. However, a healthy submodule could change to a bad state because of wrong submodule parameter.

4.3.2.6.5 Behavior of the attribute Submodule State

The attribute behavior of an IO device application is described in 4.3.3.7.

4.3.2.7 Isochronous Mode Application ASE

4.3.2.7.1 Overview

4.3.2.7.1.1 General

This ASE specifies the structure of the necessary parameters for isochronous mode of applications. The IsoM ASE provides a set of services used to read, and write their values. This ASE is an option and does not effect non-synchronous applications.

4.3.2.7.2 Isochronous Mode Application data class specification

4.3.2.7.2.1 General

The Isochronous Mode Application defines one Isochronous Mode Application object type:

4.3.2.7.2.2 Isochronous Mode Application class specification

4.3.2.7.2.2.1 Template

A Isochronous Mode Application object is described by the following template:

DP ASE:	Isochronous Mode Application ASE
CLASS:	Isochronous Mode Application
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of Modules
2.1	(m) Attribute: Slot Number
2.2	(m) Attribute: List of Submodules
2.2.1	(m) Attribute: Subslot Number
2.2.2	(m) Attribute: IsoM Data
2.2.2.1	(m) Attribute: Time Data Cycle
2.2.2.2	(m) Attribute: Time IO Input
2.2.2.3	(m) Attribute: Time IO Output
2.2.2.4	(m) Attribute: Time IO Input Valid
2.2.2.5	(m) Attribute: Time IO Output Valid
2.2.2.6	(m) Attribute: Controller Application Cycle Factor
2.2.3	(m) Attribute: Device Parameter
2.2.3.1	(m) Attribute: Time Data Cycle Base
2.2.3.2	(m) Attribute: Time IO Base

2.2.3.3	(m) Attribute:	Time Data Cycle Min
2.2.3.4	(m) Attribute:	Time Data Cycle Max
2.2.3.5	(m) Attribute:	Time IO Input Min
2.2.3.6	(m) Attribute:	Time IO Output Min

SERVICES:

1	(m) OpsService:	Write IsoM Data
2	(m) OpsService:	Read IsoM Data
3	(m) OpsService:	SYNCH Event

4.3.2.7.2.2 Attributes**Implicit**

The attribute Implicit indicates that the Isochronous Mode Application object is implicitly addressed by the service.

List of Modules

This attribute list contains the following attributes:

Slot Number

This attribute defines to which module the Isochronous Mode Application attribute belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

List of Submodules

This attribute list contains the following attributes:

Subslot Number

This attribute defines to which subslot the Isochronous Mode Application attribute belongs.

Attribute Type: Unsigned16

Allowed values: 1 to 0x7FFF

IsoM Data

This attribute consists of the following attributes:

Time Data Cycle (T_DC)

This attribute indicates the time factor for the application data cycle. This time includes all parts of the isochronous data cycle to convey inputs and outputs. The cycle length is a multiple of the Send Clock. The time for T_DC is calculated $T\_DC * \text{Send Clock Factor} * 31,25 \mu\text{s}$.

Attribute Type: Unsigned16

Allowed values: 1 to 1024

Time IO Input (T_IO_Input)

This attribute indicates the time factor to fetch the inputs for the IO device. The time base is value of T_IO_Base multiplied with 1 ns. The time for T_IO_Input is calculated $T\_IO\_Input * T\_IO\_Base * 1 \text{ ns}$. The time (T_IO_Input) is related to the start of the Time Data Cycle and describes a time before the start of Time Data Cycle.

For T_IO_Input the following relation shall be applied:

$$(T\_IO\_InputValid + T\_IO\_InputMin) \leq T\_IO\_Input \leq T\_DC$$

Attribute Type: Unsigned32

Allowed values: 0 to 32000000

Time IO Output (T_IO_Output)

This attribute indicates the time factor to set the outputs for the IO device. The time base is value of T_IO_Base multiplied with 1 ns. The time for T_IO_Output is calculated $T\_IO\_Output * T\_IO\_Base * 1 \text{ ns}$. The time (T_IO_Input) is related to the start of the Time Data Cycle and describes a time after the start of Time Data Cycle.

For T_IO_Output the following relation shall be applied:

$(T\_IO\_OutputValid + T\_IO\_OutputMin) \leq T\_IO\_Output \leq T\_DC$

Attribute Type: Unsigned32

Allowed values: 0 to 32000000

Time IO Input Valid (T_IO_InputValid)

This attribute indicates the point of time the new input data shall be available to convey. This point of time is related to the start of T_DC. The time is based on T_IO_Base.

Attribute Type: Unsigned32

Allowed values: 0 to 32000000

Time IO Output Valid (T_IO_OutputValid)

This attribute indicates the point of time the new output data are available to process. This point of time is related to the start of T_DC. The time is based on T_IO_Base.

Attribute Type: Unsigned32

Allowed values: 0 to 32000000

Controller Application Cycle Factor (CACF)

This attribute indicates the IO Controller Application Cycle Time as a factor in multiples of T_DC. It determines the time the application of the IO Controller needs to process an isochronous application task completely. From isochronous point of view there is exactly one user task for each AR controlling the isochronous part of application. The synchronization point is determined by Send Clock within the selected Phase.

Attribute Type: Unsigned16

Allowed values: 1 to 14

Device Parameter

This attribute consists of the following attributes.

Time Data Cycle Base (T_DC_Base)

This attribute shall contain the factor for the granularity of the adjustment of the data timeliness for the application cycle.

The time basis for T_DC_Base is 31,25 µs. The time for T_DC_Base is calculated $T\_DC\_Base * 31,25 \mu s$.

This field is part of the GSDML.

Attribute Type: Unsigned16

Allowed values: 1 to 1024

Time IO Base (T_IO_Base)

This attribute shall contain the factor for the granularity of the adjustment of the input or output delay time.

The time basis for T_IO_Base is 1 ns. The time for T_IO_Base is calculated $T\_IO\_Base * 1 ns$.

This field is part of the GSDML.

Attribute Type: Unsigned32

Allowed values: 1 to 32000000

Time Data Cycle Min (T_DC_Min)

This attribute shall contain the factor for the smallest possible data cycle.

The time basis for T_DC_Min is the value of the attribute T_DC_Base. The time for T_DC_Min is calculated $T\_DC\_Min * T\_DC\_Base * 31,25 \mu s$.

This field is part of the GSDML.

Attribute Type: Unsigned16

Allowed values: 1 to 1024

Time Data Cycle Max (T_DC_Max)

This attribute shall contain the factor for the longest possible data cycle.

The time basis for T_{DC_Max} is the value of the attribute T_{DC_Base} . The time for T_{DC_Max} is calculated $T_{DC\_Max} * T_{DC\_Base} * 31,25 \mu s$.

This field is part of the GSDML.

Attribute Type: Unsigned16

Allowed values: 1 to 1024

Time IO Input Min ($T_{IO\_InputMin}$)

This attribute shall contain the factor for the smallest possible input refresh delay. It may be the smallest possible time to copy or preprocess the input data. The time is related to the start of the application cycle.

The time basis for $T_{IO\_InputMin}$ is the value of the attribute T_{IO_Base} . The time for $T_{IO\_InputMin}$ is calculated $T_{IO\_InputMin} * T_{IO\_Base} * 1 ns$.

This field is part of the GSDML.

Attribute Type: Unsigned32

Allowed values: 1 to 32000000

Time IO Output Min ($T_{IO\_OutputMin}$)

This attribute shall contain the factor for the smallest possible output refresh delay. It may be the smallest possible time to copy or preprocess the output data. The time is related to the end of the application cycle.

The time basis for $T_{IO\_OutputMin}$ is the value of the attribute T_{IO_Base} . The time for $T_{IO\_OutputMin}$ is calculated $T_{IO\_OutputMin} * T_{IO\_Base} * 1 ns$.

This field is part of the GSDML.

Attribute Type: Unsigned32

Allowed values: 1 to 32000000

Figure 26 shows the relation of the ASE attributes within the isochronous application model.

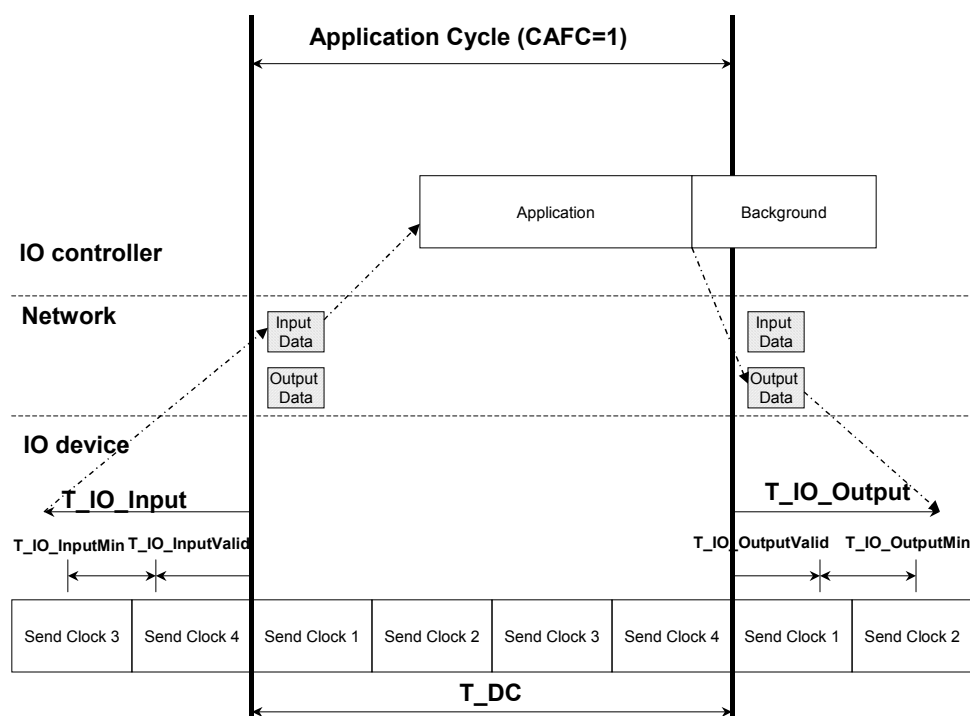


Figure 26 — General isochronous application model (example)

4.3.2.7.3 Isochronous Mode Application service specification

4.3.2.7.3.1 Write IsoM Data

This confirmed service may be used to write the attributes of the IsoM Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 51 shows the parameters of the service.

The service user shall only address modules and submodules within the List of Modules, which are part of the submodule list the AR is connected to. Otherwise the service provider shall issue a negative response to the service and all data shall be ignored.

The service parameter “Multiple” may be used to convey multiple IsoM Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 51 — Write IsoM Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
IsoM Data	M	M(=)		
Time Data Cycle	M	M(=)		
Time IO Input	M	M(=)		
Time IO Output	M	M(=)		
Time IO Input Valid	M	M(=)		
Time IO Output Valid	M	M(=)		
Controller Application Cycle Factor	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used to address a specific slot.

Subslot Number

The parameter Subslot Number is used to address a specific subslot.

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

IsoM Data

This parameter is composed of the following elements:

Time Data Cycle

This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Input

This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Output

This parameter contains the value corresponding attribute of the ASE object.

Time IO Input Valid

This parameter contains the value corresponding attribute of the ASE object.

Time IO Output Valid

This parameter contains the value corresponding attribute of the ASE object.

Controller Application Cycle Factor

This parameter contains the value corresponding attribute of the ASE object.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.7.3.2 Read IsoM Data

This confirmed service may be used to read the values of the attributes of the IsoM Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 52 shows the parameters of the service.

Implicit AR using Target AR UUID: This service does only contain Real Port Data if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 52 — Read IsoM Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
IsoM Data			M	M(=)
Time Data Cycle			M	M(=)
Time IO Input			M	M(=)
Time IO Output			M	M(=)
Time IO Input Valid			M	M(=)
Time IO Output Valid			M	M(=)
Controller Application Cycle Factor			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

Result(+)

This parameter indicates that the service request succeeded.

IsoM Data

This parameter is composed of the following elements:

Time Data Cycle

This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Input

This parameter contains the value of the corresponding attribute of the ASE object.

Time IO Output

This parameter contains the value corresponding attribute of the ASE object.

Time IO Input Valid

This parameter contains the value corresponding attribute of the ASE object.

Time IO Output Valid

This parameter contains the value corresponding attribute of the ASE object.

Controller Application Cycle Factor

This parameter contains the value corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.7.3.3 SYNCH Event

The service SYNCH indicates to the application of the IO device that a new isochronous send clock cycle has been started.

Table 53 shows the parameters of the service.

Table 53 — SYNCH Event

Parameter name	Ind
Argument	M
Global Cycle Counter	M
Status	M

Argument

The argument shall convey the service specific parameters of the service indication.

Global Cycle Counter

This parameter shall contain the local value of the cycle counter.

Status

This parameter shall contain the current status. The allowed values are LOCAL and REMOTE. The value LOCAL shall be used to indicate that no valid synchronization frame has been received within the PLL window. The value REMOTE shall be used to indicate that a valid synchronization frame has been received within the PLL window.

4.3.2.7.4 Behavior of Isochronous Mode Application objects

4.3.2.7.4.1 Overview of relations to other ASE objects

For performance reasons, the value for the Output Data object should be transferred on receipt as fast as possible to the object. The speed and the jitters of this action is a performance parameter of an IO device. This parameter determines the synchronization accuracy.

Figure 27 shows the timing relations between different involved ASEs.

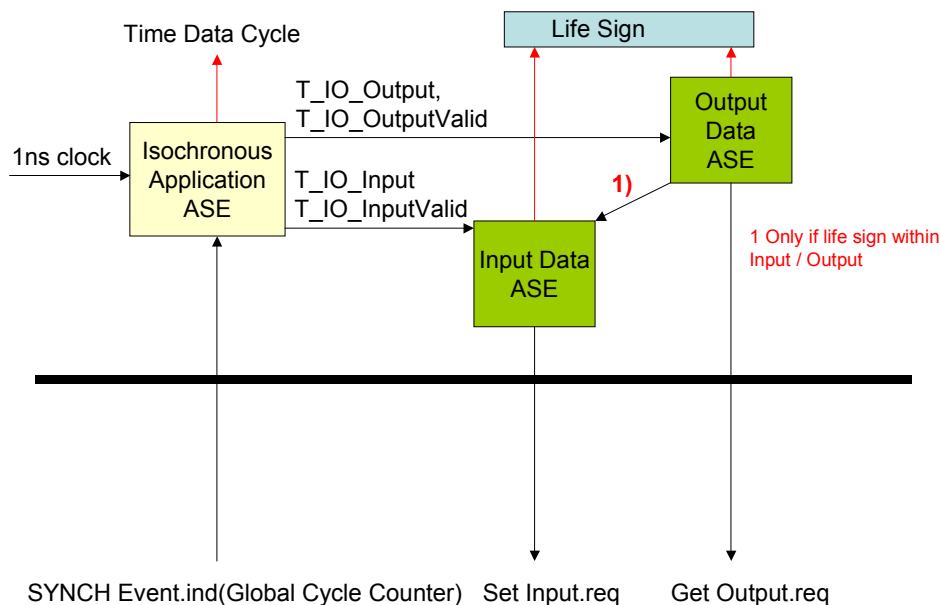


Figure 27 — ASE relations in an IO device operating in isochronous mode

Figure 28 shows the state machine relations between different involved ASEs.

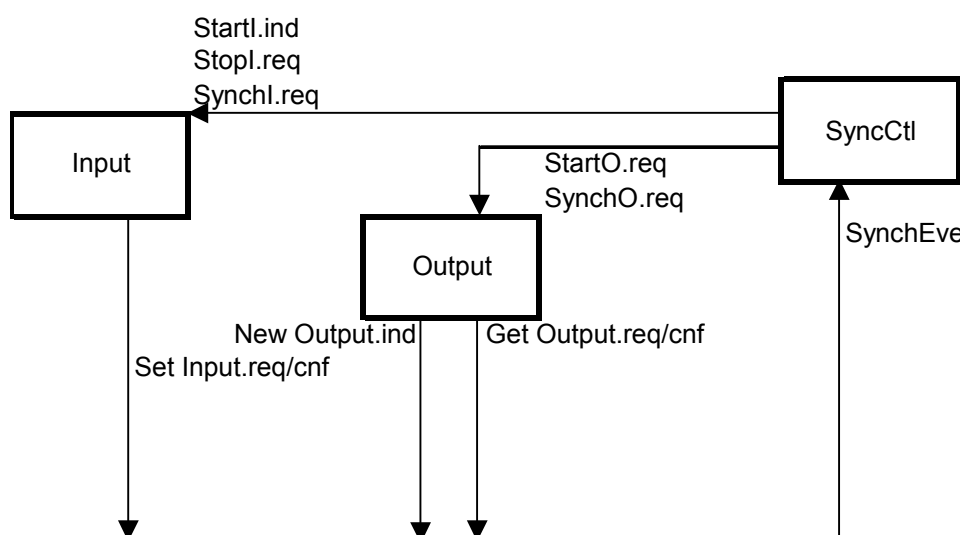


Figure 28 — State machine relations in an IO device operating in isochronous mode

The corresponding state machines are described in 4.3.2.7.4.3.

4.3.2.7.4.2 Basic behavior of input and output data in the isochronous mode

An IO system operating in the isochronous mode provides the following features:

- all IO devices taking part in the isochronous mode functionality synchronize their internal clock system with the synchronization message SYNCH Event with the isochronous data cycle
- all IO devices taking part in the isochronous mode functionality have to be in the same PTP subdomain and furthermore in the same isochronous real-time domain
- the jitter of SYNCH Event.ind is less than 1 μ s (jitters within an IO device is assumed to be zero)
- in each data cycle at least one Ethernet frame of full length may be included

In an IO system with enabled isochronous mode functionality IO devices with basic functionality can be used in conjunction with synchronized IO devices. The isochronous mode functionality tolerates and compensates communication errors like:

- faulty synchronization messages; the synchronization message may be disturbed at the sync master at the generation or at the conveyance to the sync slave or at the recognition at the sync slave
- the isochronous mode is capable to compensate the jitters of the synchronization message produced by the sync master and if present by network devices like switches; this jitter shall be less than 1 μ s (this includes the jitters within the sync master)
- the deviation of the real isochronous data cycle to the ideal, not disturbed isochronous data cycle shall not exceed 0,01%
- if the SYNCH-Event.ind is issued with the status "LOCAL" then the isochronous application shall be informed

4.3.2.7.4.3 State machine description for isochronous mode

4.3.2.7.4.3.1 General

The behavior of a Output Data object is defined by the Output Data and SyncCtl state machine in case of Isochronous Mode.

The Output Data and SyncCtl state machine represents a part of the User functionality and are included to ensure interoperability.

The Output Data and SyncCtl state machine provides events to other parts of the User functionality. The Output Data and SyncCtl state machine have interfaces to the Input Data state machine for co-ordination purposes.

The main functionalities of the Output Data state machine are:

- Transfer the received Output Data to the application on time.
- Check the correct sequence: SYNCH, New Output, read out or the Output Data at Time IO Output.
- Check the Masters Sign of Life (MLS).

The main functionalities of the SyncCtl state machine are:

- Generate timing signals from the SYNCH-Event.ind
- Generation of a pulse scheme for the local application

The synchronization primitive (SYNCH Event) is conveyed by a local PLL within the LMPM.

4.3.2.7.4.3.2 Primitive definitions

4.3.2.7.4.3.2.1 Primitives exchanged between the AL and the SyncCtl state machine

Table 54 shows primitive the exchanged between the User and the PLL state machine.

Table 54 — Primitives issued by the AL to the SyncCtl state machine

Primitive Name	Source	Associated parameters	Functions
SYNCH Event.ind	AL	AREP Global Cycle Counter Status	
Started.ind	AL	AREP	
Stopped.ind	AL	AREP	

The parameters used with the primitives are described in the Service Specification of the Output Data object (see 4.3.2.2.3).

4.3.2.7.4.3.2.2 Primitives exchanged between the user and the SyncCtl state machine

Table 55 show the primitive exchanged between the User and the SyncCtl state machine.

Table 55 — Primitives issued by the user to the SyncCtl state machine

Primitive Name	Source	Associated parameters	Functions
Output Status.req	user	Status	Reports change of Operation Mode, Warnings and Errors

4.3.2.7.4.3.2.3 Primitives exchanged between the user and the input state machine

Table 56 show the primitive exchanged between the user and the Input state machine.

Table 56 — Primitives issued by the user to the input state machine

Primitive Name	Source	Associated parameters	Functions
SYNCH In	Input		Collects the Values of the Input User Data

4.3.2.7.4.3.2.4 Primitives exchanged between the user and the output state machine

Table 57 shows the primitive exchanged between the User and the Output state machine.

Table 57 — Primitives issued by the user to the output state machine

Primitive Name	Source	Associated parameters	Functions
SYNCH Out	Output		Pass the Output Buffer to the Output User Data

4.3.2.7.4.3.2.5 Primitives exchanged between the output state machine and the SyncCtl state machine

Table 58 and Table 59 show the primitive exchanged between the Output state machine and the PLL state machine.

Table 58 — Primitives issued by the SyncCtl to the output state machine

Primitive Name	Source	Associated parameters	Functions
StartO.ind	SyncCtl		Start Output Data processing
SYNCHO.ind	SyncCtl		Trigger Signal for Output Data Processing

Table 59 — Primitives issued by the output to the SyncCtl state machine

Primitive Name	Source	Associated parameters	Functions
Output_Status.req	Output	Status	Report Status of Output Data processing

The parameter Status used with the primitive Output_Status.req reports the different warnings and errors.

4.3.2.7.4.3.2.6 Primitives exchanged between the input state machine and the SyncCtl state machine

Table 60 shows the primitive exchanged between the Input state machine and the SyncCtl state machine.

Table 60 — Primitives issued by the SyncCtl to the input state machine

Primitive Name	Source	Associated parameters	Functions
StopI.ind	SyncCtl		Stop Input Data processing
SYNCHI.ind	SyncCtl		Trigger Signal for Input Data Processing

4.3.2.7.4.3.2.7 Primitives exchanged between the input state machine and the output state machine

Table 61 shows the primitive exchanged between the Input state machine and the Output state machine.

Table 61 — Primitives issued by the output to the input state machine

Primitive Name	Source	Associated parameters	Functions
StartI.ind	Output		Start Output Data processing

4.3.2.7.4.3.2.8 Primitives exchanged between the AL and the output state machine

Table 62 shows the primitive exchanged between the Output state machine and AL.

Table 62 — Primitives issued by the output state machine to the AL

Primitive Name	Source	Associated parameters	Functions
Get Output.req	Output	AREP	

Table 63 shows the primitive exchanged between the AL and Output state machine.

Table 63 — Primitives issued by the AL to the output state machine

Primitive Name	Source	Associated parameters	Functions
Get Output.cnf	AL	AREP Output Data Clear Flag New Flag	
New Output.ind	AL	AREP Clear Flag	

4.3.2.7.4.3.2.9 Primitives exchanged between the input state machine and the AL

Table 64 shows the primitive exchanged between the Input state machine and the AL.

Table 64 — Primitives issued by the input state machine to the AL

Primitive Name	Source	Associated parameters	Functions
Set Input.req	Input	AREP Input Data	

Table 65 shows the primitive exchanged between the Input state machine and the AL.

Table 65 — Primitives issued by the AL to the input state machine

Primitive Name	Source	Associated parameters	Functions
Set Input.cnf	AL	AREP	

4.3.2.7.4.3.3 SyncCtl state diagram

Figure 29 shows the state diagram of the SyncCtl state machine.

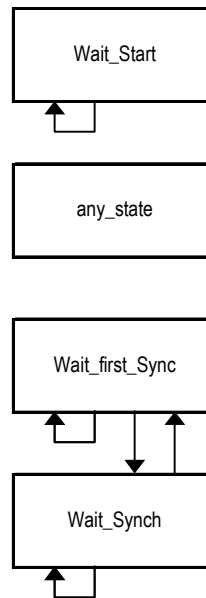


Figure 29 — SyncCtl state diagram

4.3.2.7.4.3.4 SyncCtl state table

Table 66 shows the states of a SyncCtl state machine.

Table 66 — SyncCtl state table

#	Current State	Event /Condition =>Action	Next State
1	Wait_Start_PLL	SYNCH Event.ind(AREP,Status) => ignore	Wait_Start_PLL
2	any_state	Started.ind(AREP) => RUN := 0 count_error := 0	Wait_first_SYNCH
3	any_state	Stopped.ind(AREP) => O_Status := OUTPUT_RESETMLS SYNCH Status.ind(Status)	Wait_Start_PLL
4	any_state	Output Status.req(Status) => O_Status := Status	SAME
5	Wait_Operate	SYNCH Event.ind(AREP,Status) /count_error < e_limit => count_error := count_error + 1	Wait_Startup
6	Wait_Operate	SYNCH Event.ind(AREP,Status) /count_error = e_limit => Status := PLL_SYNCH_ERR W := first_Window Tdp_n := Tdp RUN := 0 count_error := 0 start TimerW(Tdp_n-W) SYNCH Status.ind(Status) Stop_I.ind	Wait_Startup
7	Wait_Operate	TimerW expired /RUN = 1 => cnt:=1 Seg = (Tdp_n)/m Start TimerP(W+Delay) Start TimerW(2*W) SYNCH_I.ind SYNCH_O.ind	Wait_next_SYNCH
8	Wait_Operate	TimerW expired /RUN = 0 => Status := PLL_SYNCED cnt:=1 Seg = (Tdp_n)/m Start TimerP(W+Delay) RUN:= 1 Start TimerW(2*W) Start_O.ind SYNCH_I.ind SYNCH_O.ind SYNCH_Status.ind(Status)	Wait_next_SYNCH
9	Wait_Operate	TimerP expired /cnt < m => cnt:= cnt+1 Start TimerP(Seg) PULSE.ind	Wait_Operate
10	Wait_Operate	TimerP expired /cnt = m => PULSE.ind	Wait_Operate

4.3.2.7.4.3.5 OUTPUT state diagram

Figure 30 shows the state diagram of the OUTPUT state machine.

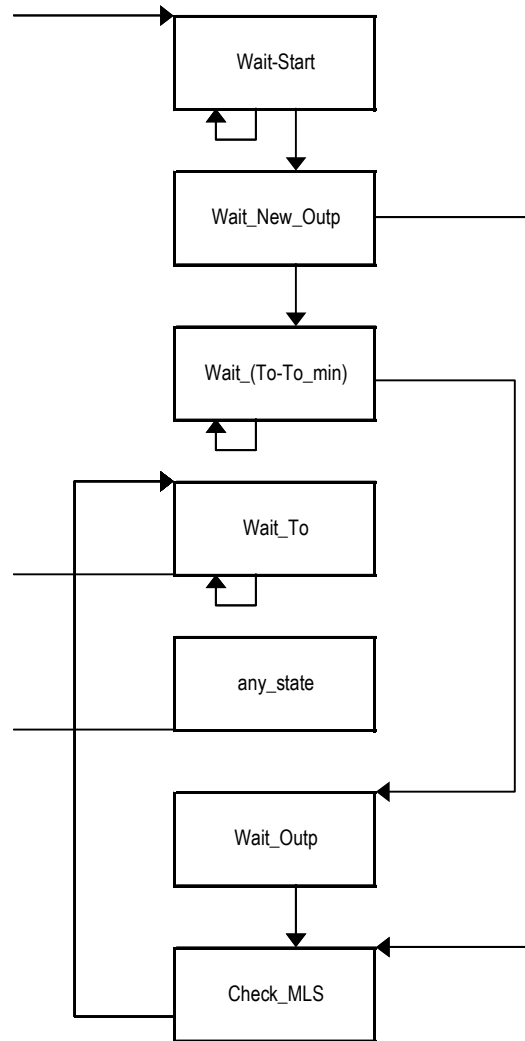


Figure 30 — OUTPUT state diagram

4.3.2.7.4.3.6 OUTPUT state table

Table 67 shows the states of a OUTPUT state machine.

Table 67 — OUTPUT state table

#	Current State	Event /Condition =>Action	Next State
1	Wait-Start	SynchO.req => Start Timer(To-To_valid) Start TimerTo	Wait_New_Outp
2	Wait-Start	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Wait-Start
3	Wait-Start	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) /wrong_output_upd = limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Wait-Start
4	Wait_New_Outp	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) => if (wrong_output_upd>0) wrong_output_upd= wrong_output_upd-1	Wait_(To-To_min)
5	Wait_New_Outp	Timer(To-To_valid) expired /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Check_MLS
6	Wait_New_Outp	Timer(To-To_valid) expired /wrong_output_upd = limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Check_MLS
7	Wait_(To-To_min)	Timer(To-To_valid) expired => Get_Output.req(AREP)	Wait_Outp
8	Wait_(To-To_min)	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Wait_(To-To_min)
9	Wait_(To-To_min)	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) /wrong_output_upd = limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Wait_(To-To_min)
10	Wait_To	TimerTo expired => SYNCH_Out	Wait-Start
11	Wait_To	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) /wrong_output_upd < limit => wrong_output_upd= wrong_output_upd+n	Wait_To
12	Wait_To	New_Output.ind (AREP, CREP, Slot Number, Subslot Number, WatchdogFlag, InDataFlag) /wrong_output_upd = limit => Status := OUTPUT_SEQUENCE Output_Status.ind (Status)	Wait_To

#	Current State	Event /Condition =>Action	Next State
13	any_state	Start_O.req => MLS_State:= Wait_first_MLS O_Buffer:=Nil(with O_Buffer.MLS:=0) old_MLS:=0 MLS_error_counter:=0 TMAPC_counter:=0 MLS_start_counter:=0	Wait-Start
14	Wait_Outp	Get_Output.cnf (AREP, IOPS, Subslot_Output_Data, New_Flag, IOCS) /New_Flag && IOPS == GOOD => O_Buffer := Output_Data	Check_MLS
15	Wait_Outp	Get_Output.cnf (AREP, IOPS, Subslot_Output_Data, New_Flag, IOCS) /!(New_Flag && IOPS == GOOD) =>	Check_MLS
16	Check_MLS	/MLS_State = Wait_first_MLS => MLS_State := Wait_next_MLS Status:=OUTPUT_RESETMLS old_MLS:=O_Buffer.MLS Output_Status.ind (Status)	Wait_To
17	Check_MLS	/MLS_State = Wait_next_MLS TMAPC_counter < MLS_factor => MLS_State := Wait_next_MLS inc(TMAPC_counter)	Wait_To
18	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter < n_TMAPC => MLS_State := Wait_next_MLS inc_LS(old_MLS) TMAPC_counter:=0 MLS_error_counter := MLS_error_counter + n_error	Wait_To
19	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter >= n_TMAPC => MLS_State := Wait_first_MLS Status:=OUTPUT_RESETMLS TMAPC_counter:=0 MLS_error_counter:=0 old_MLS:=0 MLS_start_counter:=0 Output_Status.ind (Status)	Wait_To

#	Current State	Event /Condition =>Action	Next State
20	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS = inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_start_counter < n_MLS => MLS_State := Wait_next_MLS TMAPC_counter:=0 if (MLS_error_counter >0) MLS_error_counter := MLS_error_counter -1 old_MLS:=MLS inc(MLS_start_counter)	Wait_To
21	Check_MLS	/MLS_State = Wait_next_MLS O_Buffer.MLS = inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_start_counter = n_MLS => MLS_State := Wait_MLS Status:=OUTPUT_RUNMLS TMAPC_counter:=0 MLS_error_counter:=0 old_MLS:=O_Buffer.MLS MLS_start_counter:=0 Output_Status.ind (Status)	Wait_To
22	Check_MLS	/MLS_State = Wait_MLS TMAPC_counter < MLS_factor => MLS_State := Wait_MLS inc(TMAPC_counter)	Wait_To
23	Check_MLS	/MLS_State = Wait_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter < n_TMAPC => MLS_State := Wait_MLS inc_LS(old_MLS) TMAPC_counter:=0 MLS_error_counter := MLS_error_counter + n_error	Wait_To
24	Check_MLS	/MLS_State = Wait_MLS O_Buffer.MLS <> inc_LS(old_MLS) & TMAPC_counter = MLS_factor & MLS_error_counter >= n_TMAPC => MLS_State := Wait_first_MLS Status:=OUTPUT_RESETMLS TMAPC_counter:=0 MLS_error_counter:=0 old_MLS:=0 Output_Status.ind (Status)	Wait_To
25	Check_MLS	/MLS_State = Wait_MLS O_Buffer.MLS = inc_LS(old_MLS) & TMAPC_counter = MLS_factor => MLS_State := Wait_MLS TMAPC_counter:=0 if (MLS_error_counter >0) MLS_error_counter := MLS_error_counter -1 old_MLS:=O_Buffer.MLS	Wait_To

4.3.2.7.4.3.7 INPUT state diagram

Figure 31 shows the state diagram of the INPUT state machine.

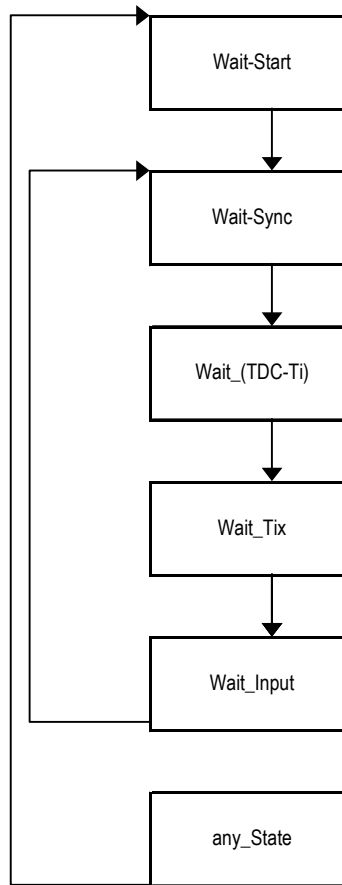


Figure 31 — INPUT state diagram

4.3.2.7.4.3.8 INPUT state table

Table 68 shows the states of a INPUT state machine.

Table 68 — INPUT state table

#	Current State	Event /Condition =>Action	Next State
1	Wait-Start	Start_I.ind =>	Wait-Sync
2	Wait-Sync	SYNCH_I.ind(Output_State) => Start Timer(TDC-Ti) Start TimerTix Store Output_State	Wait_(TDC-Ti)
3	Wait_(TDC-Ti)	Timer(TDC-Ti) expired => SYNCH_In	Wait_Tix
4	Wait_Tix	TimerTix expired /LS in Submodule && Output_State == OUTPUTRUNMLS => Input_Data:= I_Buffer inc_LS(SLS) Input_Data.SLS:= SLS Set_Input.req (AREP,CREP, Slot, Subslot, IOPS, Input_Data)	Wait_Input

#	Current State	Event /Condition =>Action	Next State
5	Wait_Tix	TimerTix expired /LS in Submodule && !(Output_State == OUTPUTRUNMLS) => Input_Data:= I_Buffer SLS := 0 Input_Data.SLS:= SLS Set_Input.req (AREP,CREP, Slot, Subslot, IOPS, Input_Data)	Wait_Input
6	Wait_Tix	TimerTix expired /!LS in Submodule => Input_Data:= I_Buffer Set_Input.req (AREP,CREP, Slot, Subslot, IOPS, Input_Data)	Wait_Input
7	Wait_Input	Set_Input.cnf(AREP) =>	Wait-Sync
8	any_State	Stop_I.ind =>	Wait-Start

4.3.2.8 Physical Device Management ASE

4.3.2.8.1 Overview

4.3.2.8.1.1 General

This ASE specifies the structure of the necessary parameters for the used protocols, e.g. ISO/IEC 8802-3, IP, UDP, RPC, or DCP. The Physical Device Management ASE manages the parameter in a common data base independent of the device instances. It represents the physical device as shown in Figure A.1. The PDMgmt ASE provides a set of services used to read, and write their values.

The attribute Station Name within the DCP group is essential for the function of an IO device and an IO controller. It shall be set by means of the DCP service DCP Set or local means. The IO device and IO controller shall store this attribute persistent. Without a Station Name the operation of an IO device is not permitted.

The attributes may be volatile or persistent. Volatile attributes are invalid after a power cycle. The values of persistent attributes remain after a power cycle.

There are two variants defined obtaining an IP address. Common to all variants is using the Station Name as the key for selecting the IP address.

The DCP protocol is mandatory and provides a service to set an IP address according to the Station Name. An IO device obtains the IP address from the IO controller or IO supervisor via the DCP Set service. An IO device shall support a volatile IP address. The DCP server is only passive in this procedure.

NOTE In order to support legacy implementations an additional mode with permanent IP addresses is permissible.

An IO controller obtains the IP address from the IO supervisor via the DCP Set service, local means, or DHCP. An IO controller shall support a persistent IP address. But it shall store the IP address only persistent if it has not received the IP address over DHCP.

The DHCP protocol should also be supported. It provides

- unique IP addresses for large systems and
- IP access for other applications (e.g. WEB access) to the IO device in case the IO controller is switched off

The DHCP client requests the IP address using the selected options. The used DHCP options are provided by the DCP Set service or local means.

EXAMPLE 1 A small environment consists of one IO controller and two IO devices. Installing and configuring of a DHCP server is not appropriate in this case. Each device has got a unique station name. The IO controller

identifies each IO device by means of the DCP Identify service with the filter station name and assigns its IP address with the DCP Set service.

EXAMPLE 2 A large environment consists of 27 IO controller and 64 IO devices per IO controller. The devices may be connected to different subnets. Installing and configuring of a DHCP server is appropriate in this case. Each device has got a unique station name and the use of DHCP is enabled. Each device requests its IP address from the DHCP server using the option e.g. Client Identifier = station name. The IO controller resolves the IP address of its related IO devices via DCP Identify or via DNS Get Host By Name service. The use of DCP Identify is restricted to a subnet.

The IO controller shall support DCP. Furthermore, it should support DNS and DHCP. DNS is used to provide IP addresses to:

- access IO devices behind IP routers
- use IO devices which receive their IP addresses from a DHCP server
- get an IP address for IO devices (without DHCP) if the IP address is not part of engineering information and DCP is selected to provide the IP address to the IO device

The startup procedure for an IO device if the station name exists provides the following steps:

- start MAC interface
- start LLDP
- start DCP receiver and DHCP client (if DHCP enabled)
- wait for IP address

The startup procedure for an IO controller if the station name exists provides the following steps:

- start MAC interface
- start LLDP
- start DCP receiver, DCP sender, and DHCP client (if DHCP enabled)
- wait for IP address or use local configured IP address
- check its own IP address and station name for multiple use by means of ARP and DCP before usage (error duplicate IP address or duplicate station name)

The IO controller provides the following steps to startup a related IO device:

- a) wait for engineering information
- b) if
 - 1) i) ask DNS server for the IP address of the IO device identified by station name
ii) CM Connect using received IP address or
 - 2) i) ask IO device for MAC address identified by station name
ii) check the IP address and station name (delivered by engineering) of the IO device for multiple use by means of ARP and DCP (error duplicate IP address or duplicate station name)
iii) set IP address of the IO device by means of DCP
iv) CM Connect using IP address

The detailed procedure for startup is provided in the protocol specification.

Furthermore, the Physical Device Management ASE defines the object attributes and access methods specific for common parameter of interface and port submodules. Besides the address resolution and neighborhood detection, these submodules contain additional attributes for global clock synchronization and RT Class 3 routing information.

4.3.2.8.2 Physical Device Management data class specification

4.3.2.8.2.1 General

The Physical Device Management ASE defines one Physical Device Management object type:

4.3.2.8.2.2 Physical Device Management class specification**4.3.2.8.2.2.1 Template**

A Physical Device Management object is described by the following template:

DP ASE:	Physical Device Management ASE
CLASS:	Physical Device Management
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: Real List of Interfaces
2.1	(m) Attribute: Slot Number
2.2	(m) Attribute: Subslot Number
2.3	(m) Attribute: Interface Name
2.4	(m) Attribute: Real List of Ports
2.4.1	(m) Attribute: Slot Number
2.4.2	(m) Attribute: Subslot Number
2.4.3	(m) Attribute: Own Port ID
2.4.4	(m) Attribute: LLDP Time To Live
2.4.5	(o) Attribute: Port Description
2.4.6	(o) Attribute: System Name
2.4.7	(o) Attribute: System Description
2.4.8	(o) Attribute: System Capabilities
2.4.9	(o) Attribute: Management Address
2.4.10	(o) Attribute: Object Identifier
2.4.11	(o) Attribute: List of Organizationally-specific extensions
2.4.11.1	(m) Attribute: Organizationally Unique Identifier
2.4.11.2	(c) Constraint: Organizationally Unique Identifier = 00-80-C2
2.4.11.2.1	(m) Attribute: List of ISO/IEC 8802.1 Subtypes
2.4.11.2.1.1	(m) Attribute: Subtype
2.4.11.2.1.2	(c) Constraint: Subtype = PORT_VLAN_ID
2.4.11.2.1.2.1	(m) Attribute: Port VLAN Identifier
2.4.11.2.1.3	(c) Constraint: Subtype = PORT_AND_PROTOCOL_VLAN_ID
2.4.11.2.1.3.1	(m) Attribute: Flags
2.4.11.2.1.3.2	(m) Attribute: PPVID Reference Number
2.4.11.2.1.4	(c) Constraint: Subtype = VLAN_NAME
2.4.11.2.1.4.1	(m) Attribute: VLAN ID
2.4.11.2.1.4.2	(m) Attribute: VLAN Name
2.4.11.2.1.5	(c) Constraint: Subtype = PROTOCOL_IDENTITY
2.4.11.2.1.5.1	(m) Attribute: List of Protocol Identities
2.4.11.2.1.5.1.1	(m) Attribute: Protocol Identity
2.4.11.3	(c) Constraint: Organizationally Unique Identifier = 00-12-0F
2.4.11.3.1	(m) Attribute: List of ISO/IEC 8802.3 Subtypes
2.4.11.3.1.1	(m) Attribute: Subtype
2.4.11.3.1.2	(c) Constraint: Subtype = MAC_PHY_CONFIGURATION_STATUS
2.4.11.3.1.2.1	(m) Attribute: Auto Negotiation Support And Status
2.4.11.3.1.2.2	(m) Attribute: Advertised Capability
2.4.11.3.1.2.3	(m) Attribute: Operational MAU Type
2.4.11.3.1.3	(c) Constraint: Subtype = POWER_VIA_MDI
2.4.11.3.1.3.1	(m) Attribute: MDI Power Support
2.4.11.3.1.3.2	(m) Attribute: Power Pair
2.4.11.3.1.3.3	(m) Attribute: Power Class
2.4.11.3.1.4	(c) Constraint: Subtype = LINK_AGGREGATION
2.4.11.3.1.4.1	(m) Attribute: Link Aggregation Status

2.4.11.3.1.4.2	(m) Attribute:	Aggregated Port ID
2.4.11.3.1.5	(c) Constraint:	Subtype = MAXIMUM_FRAME_SIZE
2.4.11.3.1.5.1	(m) Attribute:	Maximum Frame Size
2.4.12	(m) Attribute:	List of Remote Systems Data
2.4.12.1	(m) Attribute:	Port ID
2.4.12.2	(m) Attribute:	Chassis ID
2.4.12.3	(m) Attribute:	MAC Address
2.4.12.4	(m) Attribute:	LLDP Time To Live
2.4.12.5	(o) Attribute:	Port Description
2.4.12.6	(o) Attribute:	System Name
2.4.12.7	(o) Attribute:	System Description
2.4.12.8	(o) Attribute:	System Capabilities
2.4.12.9	(o) Attribute:	Management Address
2.4.12.10	(o) Attribute:	Object Identifier
2.4.12.11	(m) Attribute:	Propagation Delay Factor
2.4.12.12	(m) Attribute:	Port Properties
2.4.12.12.1	(m) Attribute:	Domain Boundary
2.4.12.12.2	(m) Attribute:	Multiple Remote Systems
2.4.12.13	(m) Attribute:	List of Organizationally-specific extensions
2.4.12.13.1	(m) Attribute:	Organizationally Unique Identifier
2.4.12.13.2	(c) Constraint:	Organizationally Unique Identifier = 00-80-C2
2.4.12.13.2.1	(m) Attribute:	List of ISO/IEC 8802.1 Subtypes
2.4.12.13.2.1.1	(m) Attribute:	Subtype
2.4.12.13.2.1.2	(c) Constraint:	Subtype = PORT_VLAN_ID
2.4.12.13.2.1.2.1	(m) Attribute:	Port VLAN Identifier
2.4.12.13.2.1.3	(c) Constraint:	Subtype = PORT_AND_PROTOCOL_VLAN_ID
2.4.12.13.2.1.3.1	(m) Attribute:	Flags
2.4.12.13.2.1.3.2	(m) Attribute:	PPVID Reference Number
2.4.12.13.2.1.4	(c) Constraint:	Subtype = VLAN_NAME
2.4.12.13.2.1.4.1	(m) Attribute:	VLAN ID
2.4.12.13.2.1.4.2	(m) Attribute:	VLAN Name
2.4.12.13.2.1.5	(c) Constraint:	Subtype = PROTOCOL_IDENTITY
2.4.12.13.2.1.5.1	(m) Attribute:	List of Protocol Identities
2.4.12.13.2.1.5.1.1	(m) Attribute:	Protocol Identity
2.4.12.13.3	(c) Constraint:	Organizationally Unique Identifier = 00-12-0F
2.4.12.13.3.1	(m) Attribute:	List of ISO/IEC 8802.3 Subtypes
2.4.12.13.3.1.1	(m) Attribute:	Subtype
2.4.12.13.3.1.2	(c) Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
2.4.12.13.3.1.2.1	(m) Attribute:	Auto Negotiation Support And Status
2.4.12.13.3.1.2.2	(m) Attribute:	Advertised Capability
2.4.12.13.3.1.2.3	(m) Attribute:	Operational MAU Type
2.4.12.13.3.1.3	(c) Constraint:	Subtype = POWER_VIA_MDI
2.4.12.13.3.1.3.1	(m) Attribute:	MDI Power Support
2.4.12.13.3.1.3.2	(m) Attribute:	Power Pair
2.4.12.13.3.1.3.3	(m) Attribute:	Power Class
2.4.12.13.3.1.4	(c) Constraint:	Subtype = LINK_AGGREGATION
2.4.12.13.3.1.4.1	(m) Attribute:	Link Aggregation Status
2.4.12.13.3.1.4.2	(m) Attribute:	Aggregated Port ID
2.4.12.13.3.1.5	(c) Constraint:	Subtype = MAXIMUM_FRAME_SIZE
2.4.12.13.3.1.5.1	(m) Attribute:	Maximum Frame Size
2.5	(m) Attribute:	Expected List of Ports
2.5.1	(m) Attribute:	Slot Number
2.5.2	(m) Attribute:	Subslot Number
2.5.3	(m) Attribute:	LLDP Time To Live

2.5.4	(o) Attribute:	Port Description
2.5.5	(o) Attribute:	System Name
2.5.6	(o) Attribute:	System Description
2.5.7	(o) Attribute:	Management Address
2.5.8	(o) Attribute:	Object Identifier
2.5.9	(o) Attribute:	List of Organizationally-specific extensions
2.5.9.1	(m) Attribute:	Organizationally Unique Identifier
2.5.9.2	(c) Constraint:	Organizationally Unique Identifier = 00-80-C2
2.5.9.2.1	(m) Attribute:	List of ISO/IEC 8802.1 Subtypes
2.5.9.2.1.1	(m) Attribute:	Subtype
2.5.9.2.1.2	(c) Constraint:	Subtype = PORT_VLAN_ID
2.5.9.2.1.2.1	(m) Attribute:	Port VLAN Identifier
2.5.9.2.1.3	(c) Constraint:	Subtype = PORT_AND_PROTOCOL_VLAN_ID
2.5.9.2.1.3.1	(m) Attribute:	Flags
2.5.9.2.1.3.2	(m) Attribute:	PPVID Reference Number
2.5.9.2.1.4	(c) Constraint:	Subtype = VLAN_NAME
2.5.9.2.1.4.1	(m) Attribute:	VLAN ID
2.5.9.2.1.4.2	(m) Attribute:	VLAN Name
2.5.9.3	(c) Constraint:	Organizationally Unique Identifier = 00-12-0F
2.5.9.3.1	(m) Attribute:	List of ISO/IEC 8802.3 Subtypes
2.5.9.3.1.1	(m) Attribute:	Subtype
2.5.9.3.1.2	(c) Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
2.5.9.3.1.2.1	(m) Attribute:	Auto Negotiation Support And Status
2.5.9.3.1.3	(c) Constraint:	Subtype = LINK_AGGREGATION
2.5.9.3.1.3.1	(m) Attribute:	Link Aggregation Status
2.5.9.3.1.3.2	(m) Attribute:	Aggregated Port ID
2.5.10	(m) Attribute:	List of Remote Systems Data
2.5.10.1	(m) Attribute:	Port ID
2.5.10.2	(m) Attribute:	Chassis ID
2.5.10.3	(m) Attribute:	LLDP Time To Live
2.5.10.4	(o) Attribute:	System Name
2.5.10.5	(m) Attribute:	Propagation Delay Factor
2.5.10.6	(m) Attribute:	Port Properties
2.5.10.6.1	(m) Attribute:	Domain Boundary
2.5.10.6.2	(m) Attribute:	Multiple Remote Systems
2.5.10.7	(m) Attribute:	List of Organizationally-specific extensions
2.5.10.7.1	(m) Attribute:	Organizationally Unique Identifier
2.5.10.7.2	(c) Constraint:	Organizationally Unique Identifier = 00-80-C2
2.5.10.7.2.1	(m) Attribute:	List of ISO/IEC 8802.1 Subtypes
2.5.10.7.2.1.1	(m) Attribute:	Subtype
2.5.10.7.2.1.2	(c) Constraint:	Subtype = VLAN_NAME
2.5.10.7.2.1.2.1	(m) Attribute:	VLAN ID
2.5.10.7.2.1.2.2	(m) Attribute:	VLAN Name
2.5.10.7.3	(c) Constraint:	Organizationally Unique Identifier = 00-12-0F
2.5.10.7.3.1	(m) Attribute:	List of ISO/IEC 8802.3 Subtypes
2.5.10.7.3.1.1	(m) Attribute:	Subtype
2.5.10.7.3.1.2	(c) Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
2.5.10.7.3.1.2.1	(m) Attribute:	Auto Negotiation Support And Status
2.5.10.7.3.1.2.2	(m) Attribute:	Advertised Capability
2.5.10.7.3.1.2.3	(m) Attribute:	Operational MAU Type
2.5.10.7.3.1.3	(c) Constraint:	Subtype = POWER_VIA_MDI
2.5.10.7.3.1.3.1	(m) Attribute:	MDI Power Support
2.5.10.7.3.1.3.2	(m) Attribute:	Power Pair
2.5.10.7.3.1.3.3	(m) Attribute:	Power Class

2.5.10.7.3.1.4	(c) Constraint:	Subtype = LINK_AGGREGATION
2.5.10.7.3.1.4.1	(m) Attribute:	Link Aggregation Status
2.5.10.7.3.1.4.2	(m) Attribute:	Aggregated Port ID
2.5.10.7.3.1.5	(c) Constraint:	Subtype = MAXIMUM_FRAME_SIZE
2.5.10.7.3.1.5.1	(m) Attribute:	Maximum Frame Size
2.6	(m) Attribute:	DL
2.6.1	(m) Attribute:	MAC Address
2.7	(m) Attribute:	IP
2.7.1	(m) Attribute:	IP Address
2.7.2	(m) Attribute:	Subnet Mask
2.7.3	(m) Attribute:	Standard Gateway
2.8	(m) Attribute:	DNS
2.8.1	(m) Attribute:	Primary DNS Server
2.8.2	(m) Attribute:	Secondary DNS Server
2.9	(m) Attribute:	Station Name
2.9.1	(m) Attribute:	Real Station Name
2.9.2	(m) Attribute:	List of Station Name Aliases
2.9.2.1	(m) Attribute:	Station Name Alias
2.10	(m) Attribute:	ARP
2.10.1	(m) Attribute:	Cache Size
2.10.2	(m) Attribute:	Cache Timeout
2.11	(m) Attribute:	IR Data
2.11.1	(m) Attribute:	List of RT Class 3 Interfaces
2.11.1.1	(m) Attribute:	Slot Number
2.11.1.2	(m) Attribute:	Subslot Number
2.11.1.3	(m) Attribute:	IR Global Data
2.11.1.3.1	(m) Attribute:	IR Data ID
2.11.1.4	(m) Attribute:	List of IR Frame Data Elements
2.11.1.4.1	(m) Attribute:	Frame Send Offset
2.11.1.4.2	(m) Attribute:	Data Length
2.11.1.4.3	(m) Attribute:	Reduction Ratio
2.11.1.4.4	(m) Attribute:	Phase
2.11.1.4.5	(m) Attribute:	Frame ID
2.11.1.4.6	(m) Attribute:	Ethertype
2.11.1.4.7	(m) Attribute:	Rx Port
2.11.1.4.8	(m) Attribute:	Frame Details
2.11.1.4.8.1	(m) Attribute:	Sync Frame
2.11.1.4.8.2	(m) Attribute:	Meaning Frame Send Offset
2.11.1.4.9	(m) Attribute:	Tx Port Group
2.11.1.4.9.1	(m) Attribute:	Number Of Tx Port Group Array Elements
2.11.1.4.9.2	(m) Attribute:	Tx Port Group Array
2.12	(m) Attribute:	Real Sync Data
2.12.1	(m) Attribute:	List of Interfaces
2.12.1.1	(m) Attribute:	Slot Number
2.12.1.2	(m) Attribute:	Subslot Number
2.12.1.3	(m) Attribute:	PTP Subdomain ID
2.12.1.4	(m) Attribute:	IR Data ID
2.12.1.5	(m) Attribute:	Reserved Interval Begin
2.12.1.6	(m) Attribute:	Reserved Interval End
2.12.1.7	(m) Attribute:	PLL Window
2.12.1.8	(m) Attribute:	Sync Send Factor
2.12.1.9	(m) Attribute:	Send Clock Factor
2.12.1.10	(m) Attribute:	Sync Properties
2.12.1.10.1	(m) Attribute:	Role

2.12.1.10.2	(m) Attribute:	Sync Class
2.12.1.11	(m) Attribute:	Sync Frame Address
2.13	(m) Attribute:	Expected Sync Data
2.13.1	(m) Attribute:	List of Interfaces
2.13.1.1	(m) Attribute:	Slot Number
2.13.1.2	(m) Attribute:	Subslot Number
2.13.1.3	(m) Attribute:	PTP Subdomain ID
2.13.1.4	(m) Attribute:	IR Data ID
2.13.1.5	(m) Attribute:	Reserved Interval Begin
2.13.1.6	(m) Attribute:	Reserved Interval End
2.13.1.7	(m) Attribute:	PLL Window
2.13.1.8	(m) Attribute:	Sync Send Factor
2.13.1.9	(m) Attribute:	Send Clock Factor
2.13.1.10	(m) Attribute:	Sync Properties
2.13.1.10.1	(m) Attribute:	Role
2.13.1.10.2	(m) Attribute:	Sync Class
2.13.1.11	(m) Attribute:	Sync Frame Address
3	(m) Attribute:	Host Name

SERVICES:

1	(m) OpsService:	DCP Get
2	(m) OpsService:	DCP Set
3	(m) OpsService:	DCP Identify
4	(m) OpsService:	DCP Identify Q
5	(o) OpsService:	DHCP Get IP
6	(o) OpsService:	DNS Get Host By Name
7	(o) OpsService:	LLDP Notification
8	(o) OpsService:	SNMP Get
9	(m) OpsService:	DL Get Parameter
10	(m) OpsService:	Write Expected Port Data
11	(m) OpsService:	Write Adjusted Port Data
12	(m) OpsService:	Read Real Port Data
13	(m) OpsService:	Read Expected Port Data
14	(m) OpsService:	Read Adjusted Port Data
15	(m) OpsService:	Write IR Data
16	(m) OpsService:	Read IR Data
17	(m) OpsService:	Write Sync Data
18	(m) OpsService:	Read Real Sync Data
19	(m) OpsService:	Read Expected Sync Data
20	(m) OpsService:	Read PDev Data
21	(m) OpsService:	Sync Event

4.3.2.8.2.2.2 Attributes**Implicit**

The attribute Implicit indicates that the PDMgmt object is implicitly addressed by the service.

Real List of Interfaces

This attribute list contains the following attributes:

Slot Number

This attribute contains the slot number of the module containing the interface submodule.

Attribute Type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number

This attribute contains the subslot number of the interface.

Attribute Type: Unsigned16

Allowed Values shall be used according to Table 69.

Table 69 — Subslot Number for Interface Submodules

Value (hexadecimal)	Meaning
0x8000	First Interface Submodule
0x8100	Interface Submodule 2
0x8200	Interface Submodule 3
0x8300	Interface Submodule 4
0x8400	Interface Submodule 5
0x8500	Interface Submodule 6
0x8600	Interface Submodule 7
0x8700	Interface Submodule 8
0x8900	Interface Submodule 9
0x8A00	Interface Submodule 10
0x8B00	Interface Submodule 11
0x8C00	Interface Submodule 12
0x8D00	Interface Submodule 13
0x8E00	Interface Submodule 14
0x8F00	Interface Submodule 15

Interface Name

This non-volatile attribute contains the name of the interface which shall be used as Chassis ID subtype locally assigned compliant with IEEE P802-1AB. This field shall contain an octet string indicating the specific identifier for the particular chassis in this system. Because Chassis ID and Port ID are concatenated to form the local MSAP identifier, the value chosen for Chassis ID shall be non-null. The value of the attribute shall be equal to the value Real Station Name.

NOTE This attribute can be written via DCP services (Name of Station).

Attribute Type: VisibleString[240]

Real List of Ports

This attribute list contains the following attributes:

NOTE The subordinated attributes of Real List of Ports are accessible via SNMP and LLDP MIB.

Slot Number

This attribute contains the slot number of the module containing the port submodule.

Attribute Type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number

This attribute contains the subslot number of the local port.

Attribute Type: Unsigned16

Allowed Values shall be used according to Table 70. The value for the parameter “i” in Table 70 shall correspond with the related interface submodule. Each interface may contain up to 255 ports.

NOTE The related interface module may be placed in a different slot.

Table 70 — Subslot Number for Port Submodules

Value (hexadecimal)	Meaning
0x8i01	First Port Submodule (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i02	Port Submodule 2 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i03	Port Submodule 3 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i04	Port Submodule 4 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i05	Port Submodule 5 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)
0x8i06	Port Submodule 6 (0<=i<=F; shall correspond to the subslot number of the related interface submodule)

Value (hexadecimal)	Meaning
0x8i07	Port Submodule 7 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i08	Port Submodule 8 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i09	Port Submodule 9 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i0A	Port Submodule 10 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i0B	Port Submodule 11 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i0C	Port Submodule 12 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i0D	Port Submodule 13 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i0E	Port Submodule 14 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i0F	Port Submodule 15 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i10	Port Submodule 16 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
...	...
0x8iFF	Port Submodule 255 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)

Own Port ID

This attribute contains the name of the local port which shall be used as Port ID subtype locally assigned compliant with IEEE P802-1AB. The value shall be “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 value “port-001” or “port-001.00000” shall be used for the first port submodule for an interface.

Attribute Type: OctetString[8] or OctetString[14]

NOTE Within the context of LLDP protocol a DTE is referred to as port. It does not mean a port from the UDP context.

LLDP Time To Live

This attribute indicates the number of seconds that the recipient LLDP agent is to regard the information associated with this chassis and port identifier to be valid. It shall be compliant with IEEE P802-1AB.

Attribute Type: Unsigned16

Allowed Values: 10

Port Description

This attribute shall contain an alpha-numeric string that indicates the port description. It shall be compliant with IEEE P802-1AB.

NOTE If IETF RFC 2863 is implemented, the use of the ifDescr object is recommended for this field.

System Name

This attribute shall contain an alpha-numeric string that indicates the system's administratively-assigned name. It shall be compliant with IEEE P802-1AB.

NOTE This should be the system's fully-qualified domain name. If implementations support IETF RFC 3418, the use of the sysName object is recommended for this field.

System Description

This attribute shall contain an alpha-numeric string that is the textual description of the network entity. It shall be compliant with IEEE P802-1AB.

NOTE This value should include the full name and version identification of the system's hardware type, software operating system, and networking software. If implementations support IETF RFC 3418, the use of the sysDescr object is recommended for this field.

System Capabilities

This attribute shall contain a bit-map of the capabilities that define the primary function(s) of the system. The bit positions for each function and the associated MIB or standard that are likely, but not guaranteed, to be supported are listed in Table 71. A

binary one in the associated bit indicates the existence of that capability. It shall be compliant with IEEE P802-1AB.

NOTE Individual systems may indicate more than one implemented functional capability (for example, both a bridge and router capability).

Table 71 — System Capabilities

Position	Meaning
Bit 0	Other
Bit 1	Repeater
Bit 2	Bridge
Bit 3	WLAN Access Point
Bit 4	Router
Bit 5	Telephone
Bit 6	DOC SIS cable device
Bit 7	Station Only

Management Address

This attribute shall contain an octet string indicating the particular management address. It shall be compliant with IEEE P802-1AB.

NOTE The returned address should be the most appropriate for management use, typically a layer-3 address such as the IPv4 address, 192.168.254.10.

Object Identifier

This attribute shall contain an OID that identifies the type of hardware component or protocol entity associated with the indicated management address. If no OID is available, this field shall not be provided. It shall be compliant with IEEE P802-1AB.

NOTE The interface number and OID are included to assist NMS discovery by indicating Enterprise-specific or other starting points for the search, such as the Interface (see RFC 2863) or Entity MIB (see RFC 2737).

List of Organizationally-specific extensions

This attribute list contains the following attributes:

Organizationally Unique Identifier

This attribute contains the TLV category that is provided to allow different organizations, such as ISO/IEC 8802.1, ISO/IEC 8802.3, IETF, as well as individual software and equipment vendors, to define TLVs that advertise information to remote entities.

List of ISO/IEC 8802.1 Subtypes

This attribute list contains the following attributes:

Subtype

This attribute shall be compliant with IEEE P802-1AB.

Allowed values: PORT_VLAN_ID, PORT_AND_PROTOCOL_VLAN-ID, VLAN_NAME, PROTOCOL_IDENTITY

Port VLAN Identifier

This attribute shall contain the VLAN ID for the bridge port as defined in IEEE 802.1Q. A value of zero shall be used if the system either does not know the PVID or does not support port-based VLAN operation.

Flags

This attribute shall contain a bit map indicating the port and protocol VLAN capability and status.

Allowed Values: PPVID_SUPPORTED, PPVID_ENABLED

PPVID Reference Number

This attribute shall contain the PPVID number for this 802 LAN station.

NOTE If the port is not capable of supporting port and protocol VLANs and/or the port is not enabled with any port protocol VLAN, the PPVID number should be zero.

VLAN ID

This attribute shall contain the VID number associated with the VLAN name.

VLAN Name

This attribute shall contain the VLAN's name.

Attribute type: OctetString[32]

NOTE If implementations support IETF RFC 2674, the use of the dot1QVLANStaticName object is recommended for this field.

List of Protocol Identities

This attribute list contains the following attributes:

Protocol Identity

This attribute shall contain the first *n* octets of the protocol after the layer-2 addresses (i.e., starting with the length/type field) that the sender would like to advertise. The value of *n* is determined by the need for the protocol to disambiguate itself. The protocol information string shall include enough octets to allow the receiver to correctly identify the protocol and its version.

NOTE Spanning tree protocols, for example, would need to include five octets: LLC address (two octets), Protocol ID (two octets), plus the protocol version (one octet).

List of ISO/IEC 8802.3 Subtypes

This attribute list contains the following attributes:

Subtype

This attribute shall be compliant with IEEE P802-1AB.

Allowed values: MAC_PHY_CONFIGURATION_STATUS, POWER_VIA_MDI, LINK_AGGREGATION, MAXIMUM_FRAME_SIZE

Auto Negotiation Support And Status

This attribute shall contain a bit map that identifies the auto-negotiation support and current status of the local ISO/IEC 8802.3 LAN station as defined in Table 72. If auto-negotiation support bit (bit-0) is one and the auto-negotiation status bit (bit-1) is zero, the ISO/IEC 8802.3 physical media dependent sublayer operating mode will be determined the default configuration field value rather than by auto-negotiation.

Table 72 — Auto Negotiation Support And Status

Position	Meaning
Bit 0	Autonegotiation Support (0 – not supported, 1 – supported)
Bit 1	Autonegotiation Status (0 – disabled, 1 – enabled)

NOTE This attribute can be written via Write Adjusted Port Data.

Advertised Capability

This attribute shall contain an integer value as defined by the ifMauAutoNegCapAdvertisedBits object in IETF RFC 3636.

Operational MAU Type

This attribute is an integer value indicating the operational MAU type of the sending device. This value is derived from the list position of the corresponding dot3MauType as listed in IETF RFC 3636 (or subsequent revisions) and is equal to the last number in the respective dot3MauType OID. For MAU types not listed in RFC 3636 (or subsequent revisions), the value of this field shall be set to zero.

MDI Power Support

This attribute shall contain a bit-map of the MDI power capabilities and status as defined in Table 73.

Table 73 — MDI Power Support

Position	Meaning
Bit 0	Port Class (0 – Power Sourcing Equipment, 1 – Powered Device)
Bit 1	MDI Power Support (0 – not supported, 1 – supported)
Bit 2	MDI Power State (0 – disabled, 1 – enabled)
Bit 3	Pairs Control Ability (0 – pair selection can not be controlled, 1 – pair selection can be controlled)

Power Pair

This attribute shall contain an integer value as defined by the pethPsePortPowerPairs object in IETF RFC 3621.

Power Class

This attribute shall contain an integer value as defined by the pethPsePortPowerClassification object in IETF RFC 3621.

Link Aggregation Status

This attribute shall contain a bit map of the link aggregation capabilities and the current aggregation status of the link as defined in Table 74.

Table 74 — Link Aggregation Status

Position	Meaning
Bit 0	Aggregation Capability (0 – not supported, 1 – supported)
Bit 1	Aggregation Status (0 – disabled, 1 – enabled)

Aggregated Port ID

This attribute shall contain the ISO/IEC 15802 aggregated port identifier. An AggPortID is derived from the ifNumber in the ifIndex for the interface.

Maximum Frame Size

This attribute shall contain an integer value indicating the maximum supported frame size in octets shall be set to 1522.

List of Remote Systems Data

This attribute list contains the same attributes as specified for Real List of Ports including all subsequent attributes until Maximum Frame Size. Furthermore, the following additional attributes are included.

Port ID

This attribute has the same definition as Own Port ID.

Chassis ID

This attribute has the same definition as Interface Name.

MAC Address

This attribute shall contain the devices physical address according to ISO/IEC8802-3 MAC address.

Attribute Type: OctetString[6]

Propagation Delay Factor

This attribute shall contain the measured line delay. The time base shall be 1 ns.

Attribute Type: Unsigned32

Port Properties

This attribute contains the following attributes:

Attribute Type: Unsigned8

Domain Boundary

This attribute shall contain the indication of the synchronization domain boundary.

Allowed values: SYNC_DOMAIN_CONTINUED, IRT_BOUNDARY

Multiple Peers

This attribute shall contain the cardinality of peer stations.

Allowed Values shall be used according to Table 75.

Table 75 — Multiple Peers

Value (hexadecimal)	Meaning
NO_PEER	no LLDP frame received
SINGLE_PEER	LLDP frames from only one MAC source address received
MULTIPLE_PEER	LLDP frames from more than one MAC source address received

Expected List of Ports

This attribute list contains the same attributes as specified for Real List of Ports including all subsequent attributes until Maximum Frame Size.

DL

This attribute contains the following attributes:

MAC Address

This attribute shall contain the devices physical address according to ISO/IEC8802-3 MAC address.

Attribute Type: OctetString[6]

IP

This attribute contains the following attributes:

IP Address

This attribute shall contain the IP address according to to RFC791.

Attribute Type: Unsigned32

Subnet Mask

This attribute shall contain the subnet mask according to to RFC791.

Attribute Type: Unsigned32

Standard Gateway

This attribute shall contain the IP address of the standard gateway according to to RFC791.

Attribute Type: Unsigned32

DNS

This attribute contains the following attributes:

Primary DNS Server

This attribute shall contain the IP address of the primary DNS server according to to RFC791.

Attribute Type: Unsigned32

Secondary DNS Server

This attribute shall contain the IP address of the primary DNS server according to to RFC791.

Attribute Type: Unsigned32

Station Name

This attribute contains the following attributes:

Real Station Name

This attribute contains the name of the station provided by engineering. The value of this attribute shall also be used for the attribute Interface Name (LLDP Chassis ID). The length shall not exceed 240 octets. The value "port-xyz." where x,y,z is in the range "0"-"9" shall not be used.

List of Station Name Aliases

This attribute list contains the following attributes:

Station Name Alias

This attribute contains the alias name of the station derived from LLDP options. The station name alias shall concatenate the Peer Port ID, "." and Peer Chassis ID.

NOTE Station Name Aliases = "port-001.mill-1.factory3.org"

ARP

This attribute contains the following attributes:

Cache Size

This attribute contains the size of the ARP cache and should contain a value to cache all devices that the device is connected to. The default value should be 64 for IO devices and 256 for IO controller.

NOTE The cache size influences the performance of the PROFINET IO operation.

Cache Timeout

This attribute contains the timeout value for the refresh of the ARP cache entry (see RFC 826). This attributes influences the time to recognize changed IP address MAC address couples in case of device replacement. The default value should be 180 s.

NOTE Static ARP cache entries for IO devices provide a suitable solution if supported by the UDP/IP stack.

IR Data

This attribute contains the following attributes:

List of RT Class 3 Interfaces

This attribute shall contain the RT Class 3 interfaces for IR Data Elements. A list element contains the following attributes:

Slot Number

This attribute contains the slot number of the module containing the interface submodule.

Attribute Type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number

This attribute contains the subslot number of the interface.

Attribute Type: Unsigned16

Allowed Values shall be used according to Table 76.

Table 76 — Subslot Number for Interface Submodules

Value (hexadecimal)	Meaning
0x8000	First Interface Submodule
0x8100	Interface Submodule 2
0x8200	Interface Submodule 3
0x8300	Interface Submodule 4
0x8400	Interface Submodule 5
0x8500	Interface Submodule 6
0x8600	Interface Submodule 7
0x8700	Interface Submodule 8
0x8900	Interface Submodule 9
0x8A00	Interface Submodule 10
0x8B00	Interface Submodule 11
0x8C00	Interface Submodule 12
0x8D00	Interface Submodule 13
0x8E00	Interface Submodule 14
0x8F00	Interface Submodule 15

IR Global Data

This attribute contains the following attributes:

IR Data ID

This attribute shall contain a unique identifier to identify the generation of the IR Data unambiguously.

Attribute Type: UUID

List of IR Frame Data Elements

This attribute shall contain the IR Frame Data Elements. A list element contains the following attributes:

Frame Send Offset

This attribute shall contain the send or receive offset in relation to start of the cycle. The time base is according to the RT Class 3 Time Base.

Attribute Type: Unsigned32

Data Length

This attribute shall contain the length of data including C_SDU length plus APDU_Status length.

Attribute Type: Unsigned16

Allowed Values: 44 to 1440

Reduction Ratio

This attribute shall contain the reduction ratio of the RT CLASS 3 Time Base.

Attribute Type: Unsigned16

Supported Values for RT CLASS 3: 1,2,4,8,16,32,64

Phase

This attribute shall contain the particular cycle for the frame.

Attribute Type: Unsigned16

Allowed Values: 1 until current value of the attribute Reduction Ratio

Frame ID

This attribute shall contain the identifier of the frame which is configured by project planning.

Attribute Type: Unsigned32

The allowed values according to Table 77 shall be used.

Table 77 — Frame IDs for RT Class 3

Value (hexadecimal)	Meaning
0x0000 – 0x00FF	dedicated to synchronization
0x0100 – 0x7FFF	dedicated to RT Class 3 unicast and multicast

Ethertype

This attribute shall contain the value of the Type/Length fields of the DLPDU.

Attribute Type: Unsigned16

Allowed Values: 0x8892

Rx Port

This attribute shall contain the number of the receiving port.

Attribute Type: Unsigned8

Allowed Values: 1 to 255

Frame Details

This attribute consists of the following elements:

Attribute Type: Unsigned8

Sync Frame

This field shall identify a sync frame.

Allowed values shall be according to Table 78.

Table 78 — Sync Frame

Meaning
no sync frame
sync frame

FrameSendOffset

This attribute shall contain the interpretation of the attribute FrameSendOffset with the values according to Table 79.

Table 79 —FrameSendOffset

Value (hexadecimal)	Meaning
TRANSMISSION_TIME	Field FrameSendOffset specifies the point of time for receiving or transmitting a frame
RT_CLASS_3_BEGIN	Field FrameSendOffset specifies the beginning of the RT Class 3 interval within a phase
RT_CLASS_3_END	Field FrameSendOffset specifies the ending of the RT Class 3 interval within a phase

Tx Port Group

This attribute consists of the following elements:

Number Of Tx Port Group Array Elements

This field shall count the successing Tx Port Group Array entries.

Allowed values are 1,3,5,7,9,11,13,15, ..., 31,33.

Attribute Type: Unsigned8

Tx Port Group Array

The Tx Port Group Array is an array of octets that shall contain at least one and at most 33 octets referred to as Tx Port Group Array entry. A Tx Port Group Array Element 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

$$1) N = M_{\text{highest}} \text{ DIV } 8 + 1$$

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).

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.

NOTE The term DIV stands for division without rest. The term MOD stands for the rest of the division.

Bit 0: TxPortEntry_0

This bit shall be set with the values according to Table 80 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 80 — Tx Port Entry

Value (hexadecimal)	Meaning
0x00	transmission off
0x01	transmission on

The TxPort of local injection is always placed in TxPortEntry_1 of the TxPortGroup octet number one.

Bit 1: TxPortEntry_1

This bit shall be set with the values according to Table 80 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 80 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 80 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 80 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 80 if the TxPort with the number that meets $m \text{ MOD } 8 = 5$ 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 80 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 80 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.

Where m is the number of the current TxPort with $1 \leq m \leq N$.

Real Sync Data

This attribute contains the following attributes:

List of Interfaces

This attribute shall contain the Data Elements. A list element contains the following attributes:

Slot Number

This attribute contains the slot number of the module containing the interface submodule.

Attribute Type: Unsigned16

Allowed Values: 0x0000 to 0x7FFF

Subslot Number

This attribute contains the subslot number of the interface.

Attribute Type: Unsigned16

Allowed Values shall be used according to Table 81.

Table 81 — Subslot Number for Sync Interface Submodules

Value (hexadecimal)	Meaning
0x8000	First Interface Submodule
0x8100	Interface Submodule 2
0x8200	Interface Submodule 3
0x8300	Interface Submodule 4
0x8400	Interface Submodule 5
0x8500	Interface Submodule 6
0x8600	Interface Submodule 7
0x8700	Interface Submodule 8
0x8900	Interface Submodule 9
0x8A00	Interface Submodule 10
0x8B00	Interface Submodule 11
0x8C00	Interface Submodule 12
0x8D00	Interface Submodule 13
0x8E00	Interface Submodule 14
0x8F00	Interface Submodule 15

PTP Subdomain ID

This attribute shall contain the unique identifier of the PTP subdomain which is configured by project planning. A PTP subdomain is a certain amount of IO controller and/or IO devices, which synchronize their send clocks.

Attribute Type: UUID

IR Data ID

This attribute shall contain the unique identifier of the isochronous realtime project which is configured by project planning.

Attribute Type: UUID

Reserved Interval Begin

This attribute shall contain the start of RT Class 2 reserved bandwidth relatively to the current phase in nanoseconds.

Attribute Type: Unsigned32

Reserved Interval End

This attribute shall contain the end of RT Class 2 reserved bandwidth relatively to the current phase in nanoseconds.

Attribute Type: Unsigned32

PLL Window

This attribute shall contain the maximum jitter of the start of the isochrone cycle with a time of 1 ns. The value 0 indicates no PLL Window defined within the Read Real Sync Data service. It is don't care for the Sync Master.

Attribute Type: Unsigned32

Sync Send Factor

This attribute contains the send interval of non-real-time sync PDUs. The Sync Send Factor shall be set in multiples of one millisecond.

Attribute Type: Unsigned16

The values shall be in the range of 1 to 64000.

Send Clock Factor

This attribute contains the Send Clock Factor. The Send Clock Factor shall be set in relation to the signaling rate. The Send Clock Factor for 100Mbps shall not be less than 5. The Send Clock Factor for 1000Mbps shall not be less than 1.

Attribute Type: Unsigned16

Time Base: 31,25 µs

The values shall be in the range of 1 to 128 (default 32)

Sync Properties

This attribute contains the following attributes:

Attribute Type: Unsigned16

Role

This attribute shall contain the role of the device related to time synchronization.

Allowed values shall be according to Table 82.

Table 82 — Sync Properties Role

Meaning
local sync
external sync.: Role Sync. Slave
Sync Master

Sync Class

This attribute shall contain the value for the Sync Class. Allowed values shall be according to Table 83.

Table 83 — Sync Class

Meaning
Test Stratum
Primary (GPS, atom clock)
Secondary (directly connected to a primary clock)
Connected to an external time signal (over boundary clocks)
Not connected to an external time signal
No clock synchronization

Sync Frame Address

This attribute shall contain the 5 least significant bits of the of the multicast MAC destination address indicating the PTP subdomain.

Attribute Type: Unsigned8

Expected Sync Data

This attribute list contains the same attributes as specified for Real Sync Data including all subsequent attributes.

4.3.2.8.3 Physical Device Management service specification

The used DCP services are described separately in the PROFINET DCP specification. The DHCP, DNS, and LLDP specification do not define local service interfaces. Therefore, the implementation of the mentioned service primitives is local matter and not defined in this specification.

The following services are defined to read or write ASE attributes.

4.3.2.8.3.1 Write Expected Port Data

This confirmed service may be used to write expected port data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 84 shows the parameters of the service.

The service user shall only address modules and submodules within the port data, which are part of the submodule list the AR is connected to. Otherwise the service provider shall issue a negative response to the service and all data shall be ignored. However, the service user may address less submodules. In this case the port difference service provides the real data of the missing ports.

The service parameter “Multiple” may be used to convey multiple port data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 84 — Write Expected Port Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Expected List of Ports	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
List of Peers	O	O(=)		
Peer Port ID	M	M(=)		
Peer Chassis ID	M	M(=)		
Propagation Delay Factor	O	O(=)		
MAU Type	O	O(=)		
Domain Boundary	O	O(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port information for a specific subslot.

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Expected List of Ports that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Expected List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Peers

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Chassis ID

This parameter contains the value of the corresponding attribute of the ASE object.

Propagation Delay Factor

This parameter contains the value of the corresponding attribute of the ASE object.

MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.2 Write Adjusted Port Data

This confirmed service may be used to adjust port data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 84 shows the parameters of the service.

The service user shall only address modules and submodules within the port data, which are part of the submodule list the AR is connected to. Otherwise the service provider shall issue a negative response to the service and all data shall be ignored. However, the service user may address fewer submodules. In this case the port difference service provides the real data of the missing ports.

The service parameter “Multiple” may be used to convey multiple adjusted port data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 85 — Write Adjusted Port Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Adjusted List of Ports	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Adjust MAU Type	O	O(=)		
Adjust Domain Boundary	O	O(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port information for a specific subslot.

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Expected List of Ports that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

Adjusted List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Adjust MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Adjust Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.3 Read Real Port Data

This confirmed service may be used to read real port data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 86 shows the parameters of the service.

By means of the service parameter Target AR UUID and Subslot the content of the port difference can be restricted to a specific AR or a specific Subslot as a filter function. In case a user specific filter is selected the Read Real Port Data response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain real port data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 86 — Read Real Port Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	M	M(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Real List of Ports			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
Own Port ID			M	M(=)
List of Peers			O	O(=)
Peer Port ID			M	M(=)
Peer Chassis ID			M	M(=)
Propagation Delay Factor			M	M(=)
MAU Type			M	M(=)
Domain Boundary			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

Real List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Own Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

List of Peers

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Chassis ID

This parameter contains the value of the corresponding attribute of the ASE object.

Propagation Delay Factor

This parameter contains the value of the corresponding attribute of the ASE object.

MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values: read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.4 Read Expected Port Data

This confirmed service may be used to read expected port data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 87 shows the parameters of the service.

By means of the service parameter Target AR UUID and Subslot the content of the port difference can be restricted to a specific AR or a specific Subslot as a filter function. In case a user specific filter is selected the Read Expected Port Data response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain expected port data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 87 — Read Expected Port Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Expected List of Ports			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Peers			O	O(=)
Peer Port ID			M	M(=)
Peer Chassis ID			M	M(=)
Propagation Delay Factor			O	O(=)
MAU Type			O	O(=)
Domain Boundary			O	O(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port difference information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

Expected List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Peers

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Chassis ID

This parameter contains the value of the corresponding attribute of the ASE object.

Propagation Delay Factor

This parameter contains the value of the corresponding attribute of the ASE object.

MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.5 Read Adjusted Port Data

This confirmed service may be used to read adjusted port data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 87 shows the parameters of the service.

By means of the service parameter Target AR UUID and Subslot the content of the port difference can be restricted to a specific AR or a specific Subslot as a filter function. In case a user specific filter is selected the Read Adjusted Port Data response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain adjusted port data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 88 — Read Adjusted Port Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Adjusted List of Ports			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
Adjust MAU Type			O	O(=)
Adjust Domain Boundary			O	O(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port difference information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

Result(+)

This parameter indicates that the service request succeeded.

Adjusted List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Adjust MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Adjust Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.6 Write IR Data

This confirmed service may be used to write the attributes of the IR Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 89 shows the parameters of the service.

The service user shall only use this service if the used AR is connected to one or more interface submodules. Otherwise the service provider shall issue a negative response to the service and all data shall be ignored.

The service parameter “Multiple” may be used to convey multiple IR Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 89 — Write IR Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
IR Data	M	M(=)		
List of RT CLASS 3 Interfaces	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
IR Global Data	M	M(=)		
IR Data ID	M	M(=)		
List of IR Frame Data Elements	M	M(=)		
Frame Send Offset	M	M(=)		
Data Length	M	M(=)		
Reduction Ratio	M	M(=)		
Phase	M	M(=)		
Frame ID	M	M(=)		
Ethertype	M	M(=)		
Rx Port	M	M(=)		
Frame Details	M	M(=)		
Tx Port Group	M	M(=)		
Number Of Tx Port Group Array Elements	M	M(=)		
Tx Port Group Array	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port information for a specific subslot.

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

This parameter indicates the number of octets of the IR Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

IR Data

This parameter consists of the following elements.

List of RT CLASS 3 Interfaces

This parameter contains a list of RT Class 3 Interfaces. A list element consists of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

IR Global Data

This parameter is composed of the following list elements:

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

List of IR Frame Data Elements

This parameter is composed of the following list elements:

Frame Send Offset

This parameter contains the value of the corresponding attribute of the ASE object.

Data Length

This parameter contains the value of the corresponding attribute of the ASE object.

Reduction Ratio

This parameter contains the value of the corresponding attribute of the ASE object.

Phase

This parameter contains the value of the corresponding attribute of the ASE object.

Frame ID

This parameter contains the value of the corresponding attribute of the ASE object.

Ethertype

This parameter contains the value of the corresponding attribute of the ASE object.

Rx Port

This parameter contains the value of the corresponding attribute of the ASE object.

Frame Details

This parameter contains the value of the corresponding attribute of the ASE object.

Tx Port Group

This parameter contains the value of the corresponding attribute of the ASE object.

Number Of Tx Port Group Array Elements

This parameter contains the value of the corresponding attribute of the ASE object.

Tx Port Group Array

This parameter contains the value of the corresponding attribute of the ASE object.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.7 Read IR Data

This confirmed service may be used to read the values of the attributes in IR Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 90 shows the parameters of the service.

By means of the service parameter Target AR UUID the content of the port difference can be restricted to a specific AR as a filter function using the Implicit AR. In case a user specific filter is selected the Read IR Data response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain IR Data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 90 — Read IR Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
A REP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
IR Data			M	M(=)
List of RT CLASS 3 Interfaces			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
IR Global Data			M	M(=)
IR Data ID			M	M(=)
List of IR Frame Data Elements			M	M(=)
Frame Send Offset			M	M(=)
Data Length			M	M(=)
Reduction Ratio			M	M(=)
Phase			M	M(=)
Frame ID			M	M(=)
Ethertype			M	M(=)
Rx Port			M	M(=)
Frame Details			M	M(=)
Tx Port Group			M	M(=)
Number Of Tx Port Group Array Elements			M	M(=)
Tx Port Group Array			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

This parameter indicates the number of octets of the IR Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

IR Data

This parameter consists of the following elements.

List of RT CLASS 3 Interfaces

This parameter contains a list of RT Class 3 Interfaces. A list element consists of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

IR Global Data

This parameter is composed of the following list elements:

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

List of IR Frame Data Elements

This parameter is composed of the following list elements:

Frame Send Offset

This parameter contains the value of the corresponding attribute of the ASE object.

Data Length

This parameter contains the value of the corresponding attribute of the ASE object.

Reduction Ratio

This parameter contains the value of the corresponding attribute of the ASE object.

Phase

This parameter contains the value of the corresponding attribute of the ASE object.

Frame ID

This parameter contains the value of the corresponding attribute of the ASE object.

Ethertype

This parameter contains the value of the corresponding attribute of the ASE object.

Rx Port

This parameter contains the value of the corresponding attribute of the ASE object.

Frame Details

This parameter contains the value of the corresponding attribute of the ASE object.

Tx Port Group

This parameter contains the value of the corresponding attribute of the ASE object.

Number Of Tx Port Group Array Elements

This parameter contains the value of the corresponding attribute of the ASE object.

Tx Port Group Array

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.8 Write Sync Data

This confirmed service may be used to write the attributes of the Sync Data. This service shall be used in conjunction with the IO AR or Supervisor AR. Table 91 shows the parameters of the service.

The service user shall only use this service if the used AR is connected to one or more interface submodules. Otherwise the service provider shall issue a negative response to the service and all data shall be ignored.

The service parameter “Multiple” may be used to convey multiple Sync Data within one APDU. The server shall mirror the value of “Multiple” within the response. The number of responses shall be equal to the number of requests. The order may be arbitrary.

NOTE It is possible to convey Record Data objects and attributes from other ASEs (e.g. Physical Device Management ASE or Record Data ASE) in one APDU.

Table 91 — Write Sync Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Multiple	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Sync Data	M	M(=)		
List of Interfaces	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
PTP Subdomain ID	M	M(=)		
IR Data ID	M	M(=)		
Reserved Interval Begin	M	M(=)		
Reserved Interval End	M	M(=)		
PLL Window	M	M(=)		
Sync Send Factor	M	M(=)		
Send Clock Factor	M	M(=)		
Sync Properties	M	M(=)		
Sync Frame Address	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Multiple			U	U(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Slot Number

The parameter Slot Number is used to address only port information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address port information for a specific subslot.

Multiple

The parameter Multiple shall be used to convey multiple Record Data objects within one APDU. It shall not be present if a single Record Data object shall be conveyed. The value MULTIPLE_START indicates a first Record Data object of a sequence. The value MULTIPLE_SEGMENT indicates an arbitrary further Record Data object. The value MULTIPLE_END indicates the last Record Data object and triggers the transmission of the APDU. The same applies to the response service primitive.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

Sync Data

This parameter is composed of the following list elements:

List of Interfaces

This parameter contains a list of Sync Interfaces. A list element is composed of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

PTP Subdomain ID

This parameter contains the value of the corresponding attribute of the ASE object.

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval Begin

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval End

This parameter contains the value of the corresponding attribute of the ASE object.

PLL Window

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Send Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Send Clock Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Frame Address

This parameter contains the value of the corresponding attribute of the ASE object.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.9 Read Real Sync Data

This confirmed service may be used to read the values of the attributes in Sync Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 92 shows the parameters of the service.

By means of the service parameter Target AR UUID the content of the port difference can be restricted to a specific AR as a filter function using the Implicit AR. In case a user specific filter is selected the Read Sync Data response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain Sync Data if there is a established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 92 — Read Real Sync Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Sync Data			U	U(=)
List of Interfaces			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
PTP Subdomain ID			M	M(=)
IR Data ID			M	M(=)
Reserved Interval Begin			M	M(=)
Reserved Interval End			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
PLL Window			M	M(=)
Sync Send Factor			M	M(=)
Send Clock Factor			M	M(=)
Sync Properties			M	M(=)
Sync Frame Address			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32} - 256$.

Result(+)

This parameter indicates that the service request succeeded.

Sync Data

This parameter is composed of the following list elements:

List of Interfaces

This parameter contains a list of Sync Interfaces. A list element is composed of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

PTP Subdomain ID

This parameter contains the value of the corresponding attribute of the ASE object.

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval Begin

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval End

This parameter contains the value of the corresponding attribute of the ASE object.

PLL Window

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Send Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Send Clock Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Frame Address

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNIORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.10 Read Expected Sync Data

This confirmed service may be used to read the values of the attributes in Sync Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 92 shows the parameters of the service.

By means of the service parameter Target AR UUID the content of the port difference can be restricted to a specific AR as a filter function using the Implicit AR. In case a user specific filter is selected the Read Sync Data response shall only contain items that match the filter criteria. Otherwise all items shall be responded.

Implicit AR using Target AR UUID: This service does only contain Sync Data if there is an established IO or Supervisor AR with the requested Target AR UUID. Otherwise a parameter error is responded.

Table 93 — Read Expected Sync Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	U	U(=)		
Target AR UUID	U	U(=)		
Slot Number	U	U(=)		
Subslot Number	U	U(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Sync Data			U	U(=)
List of Interfaces			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
PTP Subdomain ID			M	M(=)
IR Data ID			M	M(=)
Reserved Interval Begin			M	M(=)
Reserved Interval End			M	M(=)
PLL Window			M	M(=)
Sync Send Factor			M	M(=)
Send Clock Factor			M	M(=)
Sync Properties			M	M(=)
Sync Frame Address			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

API

This parameter shall be used to address the desired API.

Target AR UUID

This parameter shall only be used to read AR specific port difference information. If this parameter is used the service parameter Subslot Number shall not be used.

NOTE It is used to read only port difference information the requested AR is connected to.

Slot Number

The parameter Slot Number is used to address only information for a specific slot or in combination with the service parameter Subslot Number for a specific subslot.

Subslot Number

The parameter Subslot Number is used to address information for a specific subslot.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The

requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

Sync Data

This parameter is composed of the following list elements:

List of Interfaces

This parameter contains a list of Sync Interfaces. A list element is composed of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

PTP Subdomain ID

This parameter contains the value of the corresponding attribute of the ASE object.

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval Begin

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval End

This parameter contains the value of the corresponding attribute of the ASE object.

PLL Window

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Send Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Send Clock Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Frame Address

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNIORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.11 Read PDev Data

This confirmed service may be used to read the values of the attributes in Real Port Data, IR Data, and Sync Data. This service shall be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 94 shows the parameters of the service.

Table 94 — Read PDev Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Real List of Ports			U	U(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
Own Port ID			M	M(=)
List of Peers			O	O(=)
Peer Port ID			M	M(=)
Peer Chassis ID			M	M(=)
Propagation Delay Factor			M	M(=)
MAU Type			M	M(=)
Domain Boundary			M	M(=)
Expected List of Ports			U	U(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
List of Peers			O	O(=)
Peer Port ID			M	M(=)
Peer Chassis ID			M	M(=)
Propagation Delay Factor			O	O(=)
MAU Type			O	O(=)
Domain Boundary			O	O(=)
Adjusted List of Ports			U	U(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
Adjust MAU Type			O	O(=)
Adjust Domain Boundary			O	O(=)
IR Data			U	U(=)
List of RT CLASS 3 Interfaces			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
IR Global Data			M	M(=)
IR Data ID			M	M(=)

Parameter name	Req	Ind	Rsp	Cnf
List of IR Frame Data Elements			M	M(=)
Frame Send Offset			M	M(=)
Data Length			M	M(=)
Reduction Ratio			M	M(=)
Phase			M	M(=)
Frame ID			M	M(=)
Ethertype			M	M(=)
Rx Port			M	M(=)
Frame Details			M	M(=)
Tx Port Group			M	M(=)
Number Of Tx Port Group Array Elements			M	M(=)
Tx Port Group Array			M	M(=)
Sync Data			U	U(=)
List of Interfaces			M	M(=)
Slot Number			M	M(=)
Subslot Number			M	M(=)
PTP Subdomain ID			M	M(=)
IR Data ID			M	M(=)
Reserved Interval Begin			M	M(=)
Reserved Interval End			M	M(=)
PLL Window			M	M(=)
Sync Send Factor			M	M(=)
Send Clock Factor			M	M(=)
Sync Properties			M	M(=)
Sync Frame Address			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16}-1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of the Port Difference Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

Real List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Own Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

List of Peers

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Chassis ID

This parameter contains the value of the corresponding attribute of the ASE object.

Propagation Delay Factor

This parameter contains the value of the corresponding attribute of the ASE object.

MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Expected List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

List of Peers

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Port ID

This parameter contains the value of the corresponding attribute of the ASE object.

Peer Chassis ID

This parameter contains the value of the corresponding attribute of the ASE object.

Propagation Delay Factor

This parameter contains the value of the corresponding attribute of the ASE object.

MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

Adjusted List of Ports

This parameter is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Adjust MAU Type

This parameter contains the value of the corresponding attribute of the ASE object.

Adjust Domain Boundary

This parameter contains the value of the corresponding attribute of the ASE object.

IR Data

This parameter consists of the following elements.

List of RT CLASS 3 Interfaces

This parameter contains a list of RT Class 3 Interfaces. A list element consists of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

IR Global Data

This parameter is composed of the following list elements:

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

List of IR Frame Data Elements

This parameter is composed of the following list elements:

Frame Send Offset

This parameter contains the value of the corresponding attribute of the ASE object.

Data Length

This parameter contains the value of the corresponding attribute of the ASE object.

Reduction Ratio

This parameter contains the value of the corresponding attribute of the ASE object.

Phase

This parameter contains the value of the corresponding attribute of the ASE object.

Frame ID

This parameter contains the value of the corresponding attribute of the ASE object.

Ethertype

This parameter contains the value of the corresponding attribute of the ASE object.

Rx Port

This parameter contains the value of the corresponding attribute of the ASE object.

Frame Details

This parameter contains the value of the corresponding attribute of the ASE object.

Tx Port Group

This parameter contains the value of the corresponding attribute of the ASE object.

Number Of Tx Port Group Array Elements

This parameter contains the value of the corresponding attribute of the ASE object.

Tx Port Group Array

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Data

This parameter is composed of the following list elements:

List of Interfaces

This parameter contains a list of Sync Interfaces. A list element is composed of the following parameter.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

PTP Subdomain ID

This parameter contains the value of the corresponding attribute of the ASE object.

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval Begin

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval End

This parameter contains the value of the corresponding attribute of the ASE object.

PLL Window

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Send Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Send Clock Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Frame Address

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.2.8.3.12 Sync State Info

This local service indication shall be used to indicate any change of the synchronization state of an interface. Table 95 shows the parameters of the service.

Table 95 — Sync State Info

Parameter name	Ind
Argument	M
Sync Data	S
Slot Number	M
Subslot Number	M
PTP Subdomain ID	M
IR Data ID	M
Reserved Interval Begin	M
Reserved Interval End	M
PLL Window	M
Sync Send Factor	M
Send Clock Factor	M
Sync Properties	M
Sync Frame Address	M
Sync Error Status	S
Slot Number	M
Subslot Number	M

Argument

The argument shall convey the service specific parameters of the service indication.

Sync Data

This selective parameter shall be present if the interface has been synchronized. It is composed of the following list elements.

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value of the corresponding attribute of the ASE object.

PTP Subdomain ID

This parameter contains the value of the corresponding attribute of the ASE object.

IR Data ID

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval Begin

This parameter contains the value of the corresponding attribute of the ASE object.

Reserved Interval End

This parameter contains the value of the corresponding attribute of the ASE object.

PLL Window

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Send Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Send Clock Factor

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Properties

This parameter is composed of the following elements:

Sync Frame Address

This parameter contains the value of the corresponding attribute of the ASE object.

Sync Error Status

This selective parameter shall be present if the interface has been lost the synchronization. It is composed of the following list elements:

Slot Number

This parameter contains the value of the corresponding attribute of the ASE object.

Subslot Number

This parameter contains the value corresponding attribute of the ASE object.

4.3.2.9 Time ASE

4.3.2.9.1 Overview

The Time ASE specifies the Clock Synchronization mechanism to synchronize the time used within devices of a network. This time could be used for example to add a time value to an alert information and to sequence messages in a time wise order.

A system consists of one or more Time Master and several Time Receiver. Clock synchronization is done by the communication system. Adjustment and management of the clock is task of the application.

Time Master tasks are:

- Sending Time Event and Clock Value within Clock Sync Interval
- Resolution of Time Master conflicts
- Report Errors in Clock Synchronization

Time Receiver tasks are:

- Monitoring and check of the clock via clock sync interval
- Control of clock synchronization sequence
- Interpretation clock value
- Check status in clock value
- Adjustment of local time (not part of the Clock Synchronization specification)

The time object specifies two classes:

- Slave Time Class
- Link Time Class

The Slave Time Class comprises the Time Receiver tasks. The Link Time Class comprises Time Master and Time Receiver tasks. But Link Time Class differ from Slave Time Class definition.

The difference is based on a system architecture which provide one or more Time Master in a hierarchical topology and Time Receiver which receive a synchronized time value to set a clock within the device.

Two services are defined:

- Set Time
- Sync Interval Violation

The Set Time service is requested by the Time master device and the Sync Interval Violation service through Time slave devices.

Synchronization of time between devices on a fieldbus segment and a Time Master application is provided in parallel with other communication functions.

The scheme used is a "backwards time based correction". This results in very few real-time constraints being imposed on a field device. There is no requirement for generating messages at precise instants. Instead, knowledge of when a special Timer event message has been broadcasted is subsequently distributed and used to calculate appropriate clock adjustments.

NOTE The clock synchronization assumes that there are no delays between the invocation of the Set Time service and the processing begin of the related service in the Data Link layer.

The clock synchronization protocol operates in parallel with the other AL functions. The clock synchronization interval specifies the rate at which the Time Master initiates the clock synchronization sequence. The value of this parameter is a system design issue. It shall be the same for all devices on the network.

The value of the clock synchronization interval is a system design issue. For each station, there is a minimum required clock synchronization interval given by the desired clock accuracy divided by the clock drift rate. The value for the system should be set to the

minimum required value for all stations on the network segment. The value should not be set so large that it causes intolerable start-up delays when devices, such as hand-held terminals, are added to a network.

4.3.2.9.2 Time class specification

4.3.2.9.2.1 Slave time class specification

4.3.2.9.2.1.1 Template

DP ASE: Time ASE
CLASS: Slave Time
CLASS ID: not used
PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Time Data Description
3	(o)	Attribute:	Local Time Diff
4	(o)	Attribute:	Time Last Rcvd
5	(o)	Attribute:	Last Calc Diff
6	(m)	Attribute:	CS Status
7	(o)	Attribute:	Summertime

SERVICES:

1	(m)	OpsService:	Set Time
2	(m)	OpsService:	Sync Interval Violation

4.3.2.9.2.1.2 Attributes

Implicit

The attribute Implicit indicates that the Slave Time object is implicitly addressed by the services.

Time Data Description

This attribute contains a Simple Data Description. Only the Data Type Network Time shall be used. For Data Description the template given in 4.2.2.1 is used.

Local Time Diff

This attribute contains the value that needs to be added to the value of the Slave Time object to obtain the local time.

The default value shall be 0 (no local time used).

Attribute Type: Network Time Difference

Time Last Rcvd

This attribute contains the parameter Time Value of the last Set Time service.

Attribute Type: Network Time

Last Calc Diff

This attribute contains the difference between the value of the Slave Time object and the parameter Time Value of the Set Time service plus Delay Factor at the receipt of the last Set Time service.

Attribute Type: Network Time Difference

CS Status

This attribute contains the current status of the synchronization.

Attribute Type: Boolean

The allowed values are shown in Table 96.

Table 96 — CS status

Value	Meaning
FALSE	Synchronized
TRUE	Not Synchronized Time since last reception of a Set Time service is too long (greater than 3 Clock Sync Intervals).

Summertime

This attribute indicates whether the value of the attribute Local Time Diff is based on Summertime or Wintertime.

Attribute Type: Boolean

The allowed values are shown in Table 97.

Table 97 — Summertime

Value	Meaning
FALSE	Wintertime
TRUE	Summertime

4.3.2.9.2.1.3 Invocation of the slave time object

For the invocation of the Slave Time object the following rules apply:

- Only one Slave Time object shall be invoked in a device.

4.3.2.9.2.2 Link time class specification**4.3.2.9.2.2.1 Template**

DP ASE: Time ASE

CLASS: Link Time

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Time Data Description
3	(o)	Attribute:	Local Time Diff
4	(o)	Attribute:	Time Last Rcvd
5	(o)	Attribute:	Last Calc Diff
6	(m)	Attribute:	Synchronization Active
7	(o)	Attribute:	Announcement Hour
8	(o)	Attribute:	Summertime
9	(o)	Attribute:	Accuracy

SERVICES:

1	(m)	OpsService:	Set Time
2	(m)	OpsService:	Sync Interval Violation

4.3.2.9.2.2.2 Attributes**Implicit**

The attribute Implicit indicates that the Link Time object is implicitly addressed by the services.

Time Data Description

This attribute contains a Simple Data Description. Only the Data Type Network Time shall be used. For Data Description the template given in 4.2.2.1 is used.

Local Time Diff

This attribute contains the value that needs to be added to the value of the Link Time object to obtain the local time. It shall be a multiple of 30 min, between 0 and 24 hours.

The default value shall be 0 (no local time used).

Attribute Type: Network Time Difference

Time Last Rcvd

This attribute contains the parameter Time Value of the last Set Time service.

Attribute Type: Network Time

Last Calc Diff

This attribute contains the difference between the value of the Link Time object and the parameter Time Value of the Set Time service plus Delay Factor at the receipt of the last Set Time service.

Attribute Type: Network Time Difference

Synchronization Active

This attribute indicates whether the value of the Link Time object is synchronized or not. This attribute is set to 0 if either this station has received a Set Time service with the equivalent Parameter set to 0 or if the time since last reception of a Set Time service is too long (greater than 3 Clock Sync Intervals).

Attribute Type: Boolean

The allowed values are shown in Table 98

Table 98 — Synchronization Active

Value	Meaning
FALSE	Synchronization Failed
TRUE	Synchronization Active

Default value: FALSE

Announcement Hour

This attribute indicates whether a change from Summertime to Wintertime or vice versa shall be performed with the next change of the hour.

Attribute Type: Boolean

The allowed values are shown in Table 99.

Table 99 — Announcement hour

Value	Meaning
FALSE	No change with the next hour change
TRUE	Change from Summertime to Wintertime or vice versa with the next hour change

Summertime

This attribute indicates whether the value of the attribute Local Time Diff is based on Summertime or Wintertime.

Attribute Type: Boolean

The allowed values are shown in Table 97.

Accuracy

This attribute indicates the precision of the value of the Link Time object, i.e. which bit positions are relevant. The value will be taken from the Set Time service parameter Accuracy.

Attribute Type: Unsigned8

The allowed values are shown in Table 100.

Table 100 — Accuracy

Value	Meaning
0	< 1 ms
1	< 10 ms
2	< 100 ms
3	< 1 s
4	< 1µs

Default value: 0

4.3.2.9.2.3 Invocation of the link time object

For the invocation of the Link Time object the following rules apply:

- Only one Link Time object shall be invoked in a Link.

4.3.2.9.3 Time service specification

4.3.2.9.3.1 Set time

This confirmed service shall be used to set the value and some attributes of a Slave Time or Link Time object.

Table 101 shows the parameters of the Set Time service.

Table 101 — Set time

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
AREP	M	M(=)	
Time Value	M	M(=)	
Local Time Diff	M	M(=)	
Summertime	M	M(=)	
Accuracy	M	U(=)	
Synchronization Active	M	U(=)	
Announcement Hour	M	U(=)	
Result(+)			S
AREP			M
Result(-)			S
AREP			M
Status			M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Time Value

This Parameter contains the value for the Link Time or Slave Time object.

Local Time Diff

This attribute contains the value for the attribute Local Time Diff of the Link Time or Slave Time object.

Summertime

This attribute contains the value for the attribute Summertime of the Link Time or Slave Time object.

Accuracy

This attribute contains the value for the attribute Accuracy of the Link Time object and shall be present only in the indication if a Link Time object is present.

Synchronization Active

This attribute contains the value for the attribute Synchronization Active of the Link Time object and shall be present only in the indication if a Link Time object is present.

Announcement Hour

This attribute contains the value of the attribute Announcement Hour of the Link Time object and shall be present only in the indication if a Link Time object is present.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Status:

This parameter provides detailed information about the occurred error.

Allowed values: LS, LR, DS, SV, IV

4.3.2.9.3.2 Sync interval violation

The Sync Interval Violation service signals to the application of the Time Receiver that no Set Time service has been received during the last 3 Clock Sync Intervals.

Table 102 shows the parameter of the Sync Interval Violation service.

Table 102 — Sync interval violation

Parameter name	Ind
Argument	M
AREP	M

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

4.3.2.10 AR ASE

4.3.2.10.1 Overview

4.3.2.10.1.1 General

This ASE specifies the structure of the ARs including the necessary DL parameters for the specific CRs. The AR ASE in each device manages an end point of the AR and all related CRs, and maintains their local context. The AR endpoint is related to a dedicated list of application processes, which is addressed during the connection establishment phase, and all further services are within this context.

Furthermore, the PROFINET IO Application Layer defines an implicit AR which does not have a special AR ASE entry. The implicit AR is used for all remote Context Management services to establish and release dedicated IO or Supervisor ARs and for read access to other APOs without establishing a dedicated AR. Write access is not permitted over the implicit AR. This AR is locally addressed by an implicit AREP. The implicit AR requires to address the application process via API read service primitive parameter.

An AR is characterized by all of the following basic roles of the AREP used in PROFINET IO:

AR cardinality

one-to-one

In a one-to-one AR an AP communicates with exactly one remote AP.

one-to-many

In a one-to-many AR the AP of a IO device communicates with the APs of a group of IO controller and IO devices. Exactly one AP is addressed per device.

Device type relations

IO communication

between IO controller and IO devices

between IO controller and other IO controller

between IO devices and IO devices (at least one IO Controller shall be involved to configure and monitor the relationship between IO devices)

Supervisor communication

between IO supervisor and IO devices

between IO devices and IO parameter server

Implicit communication

between IO controller and IO devices

between IO supervisor and IO devices

Each IO device shall provide one implicit AR that allows to process at least one outstanding service all the time. It may be able to process more services in parallel according to local resources.

Each IO device shall provide resources for at least two explicit ARs. One of them shall be an IO AR. The second may be an IO AR or a Supervisor AR. An IO device may be able to provide more ARs in parallel according to local resources.

4.3.2.10.1.2 Communication relationship

4.3.2.10.1.2.1 General

An AR is mapped to one or more communication relationships (CRs). A CR carries out the exchange of information based on a set of rules.

4.3.2.10.1.2.2 Communication relationship endpoint

CRs are defined as a set of co-operating Communication Relationship End Points (CREPs). The local context of an CR endpoint is used to control the conveyance of APDUs on the CR. Figure 32 illustrates the Assignment of Communication Relationship to Application Relationship.

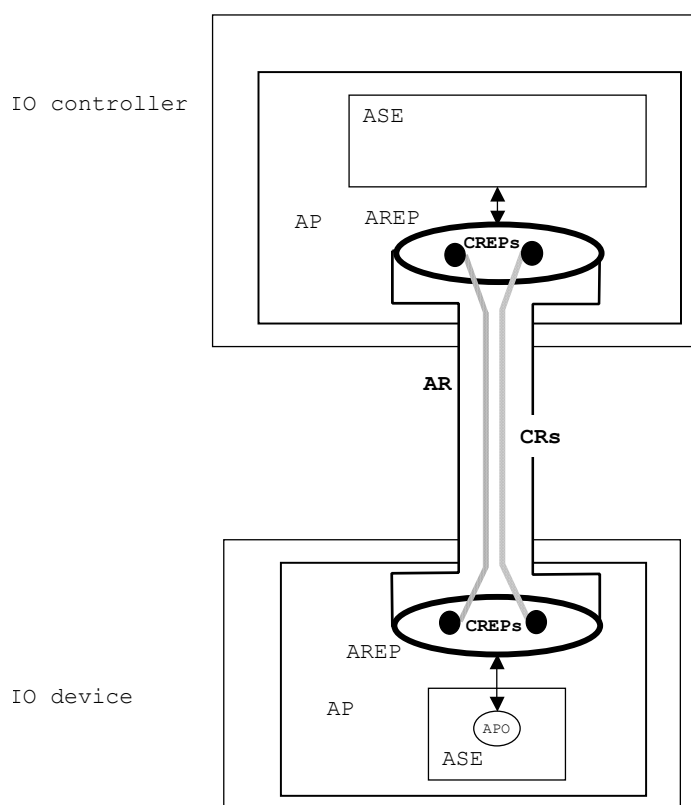


Figure 32 — Assignment of communication relationship to application relationship

4.3.2.10.1.2.3 Buffer and queues

CREPs may be modeled as a queue or as a buffer. Application data transferred over a queued CREP are delivered in the order received for conveyance. The transfer of application data over a buffered CREP is different. In this case, application data is placed in an application layer buffer for transfer.

When the CREP receives another conveyance request, it replaces the previous contents of the buffer whether or not transmitted. Once application data are written into a buffer for transfer, it is preserved in the buffer until the next application data to be transmitted replaces it. While in the buffer, application data may be read more than once without deleting it from the buffer or changing its contents.

At the receiving end, the operation is similar. The receiving application layer places the received application data into a buffer for access by the different ASEs. When subsequent application data are received, it overwrites the previous application data in the buffer whether or not it was read by the corresponding ASE. Reading the application data from the buffer is not destructive it does not destroy or change the contents of the buffer, allowing the contents to be read from the buffer one or more times.

4.3.2.10.1.2.4 Communication relationship endpoint roles

A communication relationship is characterized by all of the following basic roles of the CREP used in PROFINET IO:

CR cardinality

one-to-one:

In a one-to-one communication relationship a CREP communicates with exactly one remote CREP.

one-to-many:

In a one-to-many relationship one CREP communicates simultaneously with the CREPs of a group of devices. A one-to-many communication relationship does not permit confirmation of the execution of a request.

Connection mode

connection-oriented

In a connection-oriented application relationship a logical connection is established between application processes. Three different phases are identified in connection-oriented ARs:

- connection establishment phase
- data transfer phase and
- connection release phase

Exchanging data on a connection shall only be possible after a successful connection establishment. In the connection establishment phase a request to establish the connection is announced at the remote application process (e.g. Connect service). The request to establish the connection-oriented AR contains the information about the context to be used in the data transfer phase. The remote application informs the requesting application process about the acceptance of the context. If the context is accepted both application processes are in the data transfer phase and are able to communicate with each other according to the agreed context. Connection-oriented AR can be monitored. A connection-oriented AR is dissolved by its release (e.g. Release service).

connectionless

Connectionless application relationships have neither a connection establishment nor a connection release. They are always in the data transfer phase. It is not possible to monitor the connectionless AR.

Transmission types

cyclic

The transmission type "cyclic" means that after an initial request by the application process the application relationship is established and afterwards the data transfer is executed periodically.

acyclic

The transmission type "acyclic" means that a single transfer is executed after each request by the application process.

both

The transmission type "both" means that a cyclic as well as acyclic transfer is supported.

Transmission characteristics

Buffer to Buffer

The transmission is characterized by a buffer to buffer transfer.

Queue to Queue

The transmission is characterized by a queue to queue transfer.

Buffer to Queue

The transmission is characterized by a buffer to queue transfer.

Queue to Buffer

The transmission is characterized by a queue to buffer transfer.

Service types

Confirmed

Only confirmed services are allowed.

Unconfirmed

Only unconfirmed services are allowed.

Transmission directions:**Bi-directional**

APDUs are transmitted in both directions.

Unidirectional

APDUs are transmitted only in one direction.

Service relations**Client-Server communication**

The services are used according to the Client/Server model.

Producer-Consumer communication

The services are used according to the Producer/Consumer model.

4.3.2.10.1.2.5 Overview of communication relationships

The PROFINET IO Application Layer uses the following types of communication relationships:

BBUU (Buffer Buffer Unconfirmed Unidirectional)

This type of communication relationship is characterized by the following role attributes:

- one to one
- connection-oriented
- cyclic
- buffer to buffer
- unconfirmed
- unidirectional
- Producer/Consumer

BBUB (Buffer Buffer Unconfirmed Bidirectional)

This type of communication relationship is characterized by the following role attributes:

- one to one
- connection-oriented
- cyclic
- buffer to buffer
- unconfirmed
- bi-directional
- Client/Server

BBUU-OM (Buffer Buffer Unconfirmed Unidirectional – One to Many)

This type of communication relationship is characterized by the following role attributes:

- one to many
- connection-oriented
- cyclic
- buffer to buffer
- unconfirmed
- unidirectional
- Producer/Consumer

QQCB-CO (Queue Queue Confirmed Bi-directional – Connection-Oriented)

This type of communication relationship is characterized by the following role attributes:

- one to one
- connection-oriented
- acyclic
- queue to queue
- confirmed
- bi-directional
- Client/Server

QQCB-CL (Queue Queue Confirmed Bi-directional – Connectionless)

This type of communication relationship is characterized by the following role attributes:

- one to one
- connectionless
- acyclic
- queue to queue
- confirmed
- bi-directional
- Client/Server

4.3.2.10.2 PROFINET IO ARs

4.3.2.10.2.1 Implicit application relationship

The Implicit AR reflects the application relationship (see Figure 33) between the application processes of one IO controller and one IO device, one IO supervisor and one IO device, and of one IO device and one IO parameter server. Furthermore, the Implicit AR provides the application relationship for the following purposes:

- establish and release IO ARs
- establish and release Supervisor ARs
- read information from Record Data objects, Diagnosis objects, Identification Input or Output objects

NOTE Write access is only allowed via explicit established IO or Supervisor ARs.

The Implicit AR contains the following CR type:

- QQCB-CL

Each IO device shall provide one implicit AR that allows to process at least one outstanding service all the time. It may be able to process more services in parallel according to local resources.

The implicit AR contains one implicit CR. The endpoints AREP and CREP are unique and implicit addressed by local means. The AR UUID of the implicit AR shall be zero.

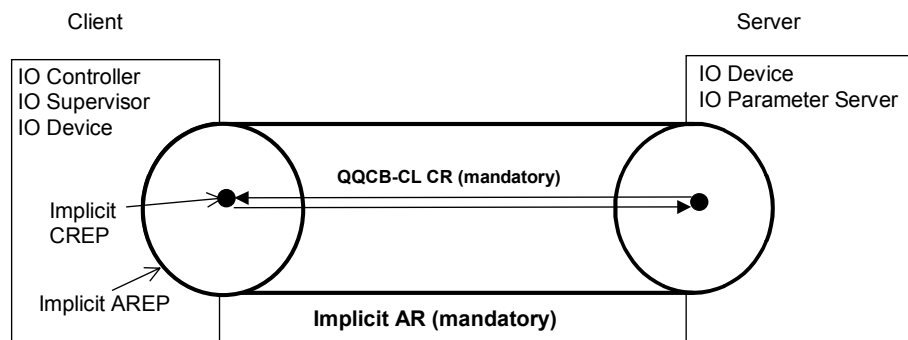


Figure 33 — Implicit application relationship

4.3.2.10.2.2 IO application relationship

The IO AR reflects the application relationship (see Figure 34) between the application processes of one IO controller and one AP of an IO device or more IO devices in case input data are published. Furthermore, the IO AR model supports redundancy and re-configuration during operation by means of using primary and backup IO ARs. The IO AR provides the application relationship for the following purposes:

- cyclic exchange of output data, optional as multicast
- cyclic exchange of input data, optional as multicast
- cyclic exchange of provider and consumer status information
- acyclic data transfer for general purpose data access e.g. identification, parameterization, and diagnosis
- report and acknowledgment of alarms

The IO AR contains the following CR types:

- at least one BBUU (mandatory) to transmit input data with its provider status from the IO device (provider) to the IO controller (consumer) and/or to transmit consumer status information (IOCS) for output data from the IO device to the IO controller

or alternatively

one BBUU-OM (optional) to transmit input data with its provider status from the IO device (publisher) to several IO controller (consumer) and several IO devices (consumer)

- at least one BBUU (mandatory) to transmit output data with its provider status from the IO controller (provider) to the IO device (consumer) and/or to transmit consumer status information (IOCS) for input data from the IO controller to the IO device

or alternatively

one BBUU-OM (optional) to transmit output data with its provider status from the IO controller (publisher) to several IO devices (consumer) and to several IO controller (consumer)

- one QQCB (mandatory) to transmit alarms with IO device as source and IO controller as destination and acknowledge in the opposite direction, it can be used to transmit alarms with IO controller as source and IO device as destination and acknowledge in the opposite direction
- one QQCB-CO (mandatory) to access general purpose data (read or write), diagnosis and identification information

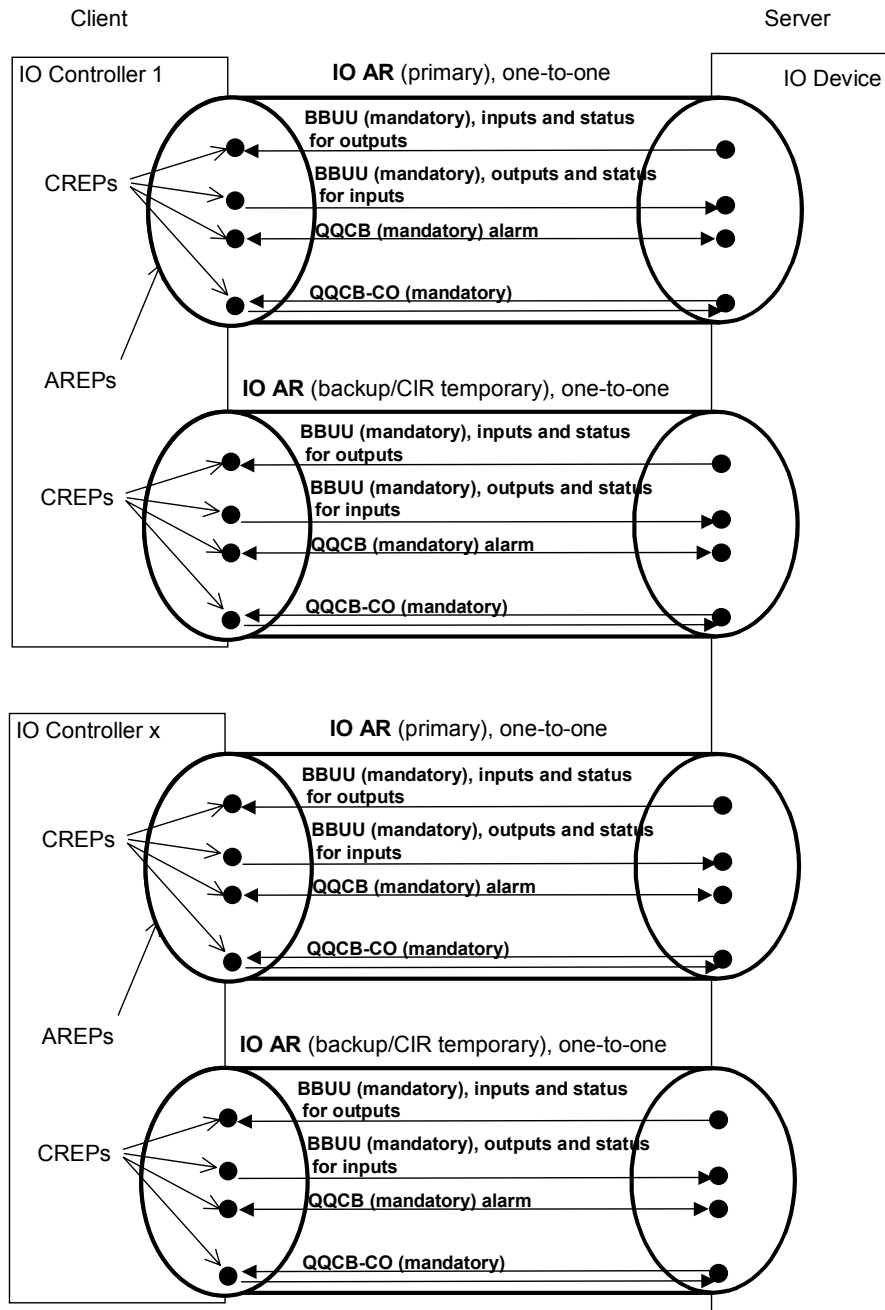


Figure 34 — Example IO application relationship (one-to-one)

- Figure 35 illustrates the IO AR with one M CR for one to many uses.

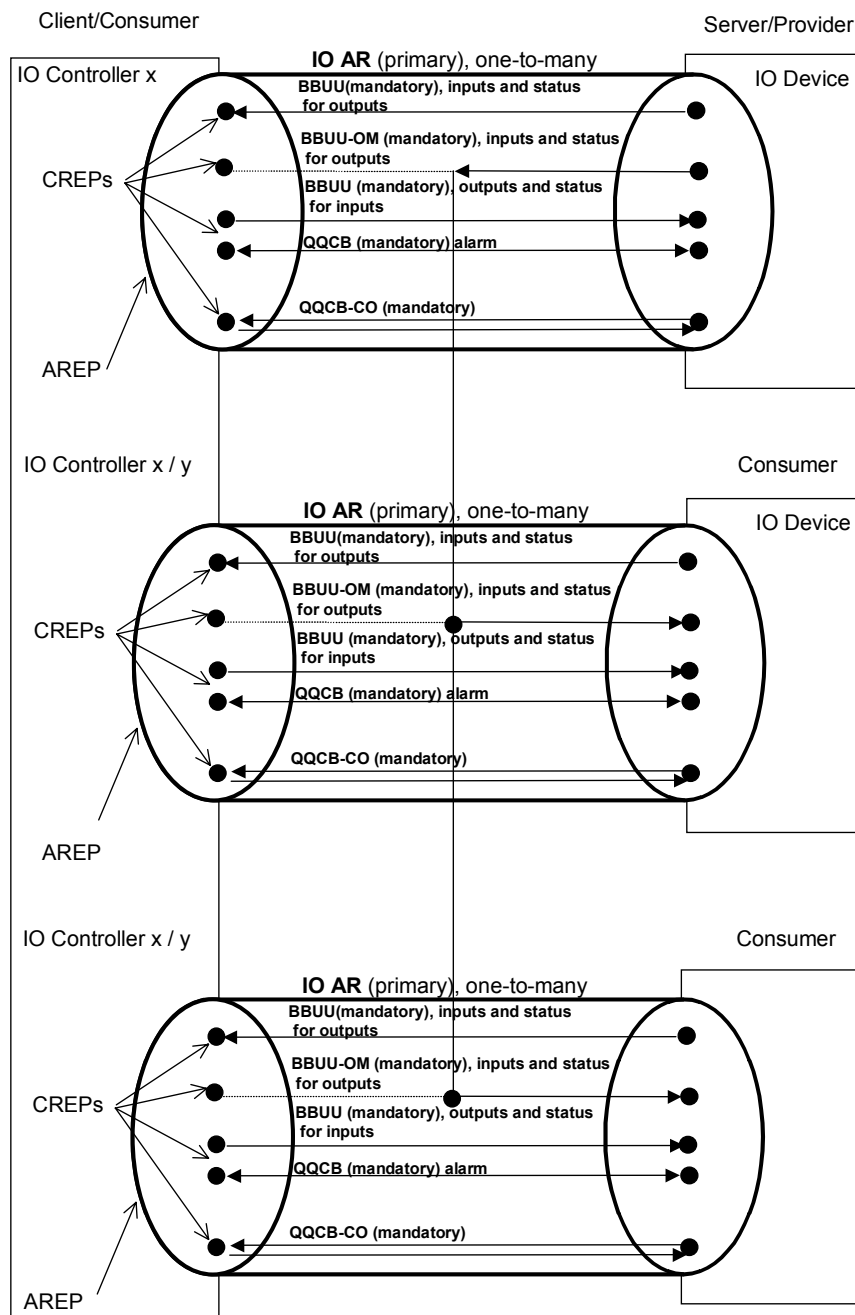


Figure 35 — Example IO application relationship one-to-many

4.3.2.10.2.3 Supervisor application relationship

The Supervisor AR is a specialized IO AR where the attribute AR Type is set to IOS AR. This kind of AR allows to takeover a submodule if allowed. An IO AR may prevent this supervisor takeover with the attribute Takeover Allowed is set to NOT_ALLOWED.

4.3.2.10.3 Application relationship list class specification

4.3.2.10.3.1 ARL class specification

4.3.2.10.3.1.1 Template

The ARL object is described by the following template:

DP ASE:	AR ASE
CLASS:	ARL
CLASS ID:	not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m) Key Attribute:	Implicit
2	(m) Attribute:	List of ARs
2.1	(m) Attribute:	AREP
2.2	(m) Attribute:	AR Type
2.3	(m) Attribute:	AR UUID
2.4	(m) Attribute:	Session Key
2.5	(m) Attribute:	CM Initiator MAC Add
2.6	(m) Attribute:	CM Responder MAC Add
2.7	(m) Attribute:	CM Initiator Object UUID
2.8	(m) Attribute:	AR Properties
2.8.1	(m) Attribute:	State
2.8.2	(m) Attribute:	Supervisor Takeover Allowed
2.8.3	(m) Attribute:	Parametrization Server
2.8.4	(m) Attribute:	Data Rate
2.8.5	(m) Attribute:	CR Transport
2.9	(m) Attribute:	CM Initiator Activity Timeout Factor
2.10	(m) Attribute:	UDP CM Initiator Port
2.11	(m) Attribute:	UDP CM Responder Port
2.12	(m) Attribute:	CM Initiator Station Name
2.13	(m) Attribute:	List of CRs
2.13.1	(m) Attribute:	CREP

SERVICES:

1	(m) OpsService:	Read AR Data

4.3.2.10.3.1.2 Attributes

Implicit

The attribute Implicit indicates that the ARL object is implicitly addressed by the service.

List of ARs

This attribute list contains the following attributes:

AREP

This attribute is the local identifier for the AR entry.

AR Type

This attribute defines the type of the AR entry.

Attribute Type: Unsigned16

Allowed values: IO_AR_SINGLE, IO_AR_CIR, IO_AR_REDUNDANT_CONTROLLER, IO_AR_REDUNDANT_DEVICE, SUPERVISOR_AR

If the device supports dynamic reconfiguration during operation than it shall accept the AR Type IO_AR_CIR (CIR means Configuration In Run).

AR UUID

This attribute defines the UUID of the AR entry which is provided by project planning or the CM Initiator.

Attribute Type: UUID

Session Key

This attribute shall be increased by one for each connect by the CM Initiator and stored within the whole session.

NOTE The Session Key allows the CM Initiator to detect sequence errors during the establishment and release phase of an AR.

Attribute Type: Unsigned16

CM Initiator MAC Add

This attribute shall contain the value of the ISO/IEC8802-3 MAC address of the initiator (IO controller or IO supervisor).

Attribute Type: OctetString[6]

CM Responder MAC Add

This attribute shall contain the value of the ISO/IEC8802-3 MAC address of the IO device.

Attribute Type: OctetString[6]

CM Initiator Object UUID

This attribute defines the UUID of the CM Initiator.

Attribute Type: UUID

AR Properties

This attribute contains the following attributes:

State

This attribute is used to distinguish between redundant and regular systems. It contains the redundancy status of the AR. The value „backup“ shall only be present if the attribute AR Type contains the value IO_AR_REDUNDANT or IO_AR_CIR. The value „primary“ indicates the operating AR and the value „backup“ indicates the hot standby AR. The „backup“ AR takes over the operation in case the PRIMARY fails. The take over is requested by the BACKUP IO controller by means of changing the APDU data status. The BACKUP IO controller shall take this action in case of receiving a Redundancy Alarm issued by the IO device.

NOTE The BACKUP and PRIMARY IO Controller may communicate over additional services to reduce the reaction time. These services are beyond the scope of this specification.

Allowed values: „primary“, „backup“

Supervisor Takeover Allowed

This attribute is used to enable or disable the supervisor takeover function for all submodules used by the AR.

Allowed values: ALLOWED, NOT_ALLOWED

Parametrization Server

This attribute contains the source for parameter data.

Allowed values: EXTERNAL_PRM_SERVER, CM_INITIATOR

Data Rate

This attribute contains the source for parameter data.

Allowed values: AT_LEAST_100Mbps, 100Mbps, 1Gbps, 10Gbps

CR Transport

This attribute is used to select the transport protocol for the IO and Alarm CR.

Allowed values: RT_CLASS_1_TRANSPORT, UDP_TRANSPORT

CM Initiator Activity Timeout Factor

This attribute defines the CM Initiator Activity Timeout Factor for the connection establishment phase. It defines the timeout value for monitoring the time period between sending the Connection response service primitive and receiving a new service request from the IO controller. The IO device application stops this monitoring when the IO CRs start their operation. Further monitoring is done by the IO CR with the values based on the CR attributes Watchdog Factor and Data Hold Factor.

Attribute Type: Unsigned16

Allowed value: 1 to 1000

Time Base: 100 ms

UDP CM Initiator Port

This attribute defines the UDP Port 0x8892.

Attribute Type: Unsigned16

UDP CM Responder Port

This attribute defines the UDP Port 0x8892.

Attribute Type: Unsigned16

CM Initiator Station Name

This attribute defines the IP address or name of the CM Initiator.

This field shall be coded as data type OctetString with 1 to 240 octets.

The following syntax applies:

- 1 or more labels, separated by [.]
- total length is 1-240
- label-length is 1-63
- labels consist of [a-z0-9-]
- labels do not start with [-]
- labels do not end with [-]
- labels do not have the form “port-xyz” with x, y, z = 0...9
- station-names do not have the form n.n.n.n, n = 0...999
- labels do only start with ‘xn—’ if RFC 3490 is applied

Furthermore, the definition of RFC 1034, and RFC 3490 shall be applied.

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

Attribute Type: VisibleString [256]

List of CRs

This attribute list shall have at least one entry. It contains the following attributes:

CREP

This attribute is the local identifier for the CR entry.

4.3.2.10.4 Communication relationship list class specification

4.3.2.10.4.1 CRL class specification

4.3.2.10.4.1.1 Template

The CRL device object is described by the following template:

DP ASE:	AR ASE
CLASS:	CRL
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of CRL Entries
2.1	(m) Attribute: CREP
2.2	(s) Attribute: IO CR Block
2.2.1	(m) Constraint: IO CR Type
2.2.2	(m) Attribute: IO CR Reference
2.2.3	(m) Attribute: LT Field
2.2.4	(m) Attribute: IO CR Properties
2.2.4.1	(m) Attribute: RT Class
2.2.4.2	(m) Attribute: Interface Type
2.2.4.3	(m) Attribute: Send Clock Synchronisation
2.2.4.4	(m) Attribute: Address Resolution
2.2.4.5	(m) Attribute: MProvider Data Status
2.2.5	(m) Attribute: C_SDU Length
2.2.6	(m) Attribute: Frame ID
2.2.7	(m) Attribute: Send Clock Factor
2.2.8	(m) Attribute: Reduction Ratio
2.2.9	(m) Attribute: Phase
2.2.10	(m) Attribute: Sequence
2.2.11	(m) Attribute: Frame Send Offset
2.2.12	(m) Attribute: Watchdog Factor
2.2.13	(m) Attribute: Data Hold Factor
2.2.14	(m) Attribute: IO CR Tag Header
2.2.14.1	(m) Attribute: Vlan ID

2.2.14.2	(m) Attribute:	IO CR User Priority
2.2.15	(m) Attribute:	IO CR Multicast MAC Add
2.2.16	(m) Attribute:	List of APIs
2.2.16.1	(m) Attribute:	API
2.2.16.2	(m) Attribute:	List of Related IO Data Objects
2.2.16.2.1	(m) Attribute:	Slot Number
2.2.16.2.2	(m) Attribute:	Subslot Number
2.2.16.2.3	(m) Attribute:	IO Data Object Frame Offset
2.2.16.3	(m) Attribute:	List of Related IOCS
2.2.16.3.1	(m) Attribute:	Slot Number
2.2.16.3.2	(m) Attribute:	Subslot Number
2.2.16.3.3	(m) Attribute:	IOCS Frame Offset
2.2.17	(o) Attribute:	Multicast Consumer Data
2.2.17.1	(m) Attribute:	Address Resolution Properties
2.2.17.2	(m) Attribute:	MCI Timeout Factor
2.2.17.3	(m) Attribute:	Provider Station Name
2.3	(s) Attribute:	Alarm CR Block
2.3.1	(m) Constraint:	Alarm CR Type
2.3.2	(m) Attribute:	LT Field
2.3.3	(m) Attribute:	Alarm CR Properties
2.3.3.1	(m) Attribute:	Priority
2.3.4	(m) Attribute:	RTA Timeout Factor
2.3.5	(m) Attribute:	RTA Retries
2.3.6	(m) Attribute:	Local Alarm Reference
2.3.7	(m) Attribute:	Remote Alarm Reference
2.3.8	(m) Attribute:	Max Alarm Data Length
2.3.9	(m) Attribute:	Alarm CR Tag Header High
2.3.9.1	(m) Attribute:	Vlan ID
2.3.9.2	(m) Attribute:	Alarm User Priority
2.3.10	(m) Attribute:	Alarm CR Tag Header Low
2.3.10.1	(m) Attribute:	Vlan ID
2.3.10.2	(m) Attribute:	Alarm User Priority

4.3.2.10.4.1.2 Attributes

Implicit

The attribute Implicit indicates that the CRL object is implicitly addressed by the service.

List of CRL Entries

This attribute is composed of the following list elements:

CREP

This static attribute contains the local identifier of this CR.

IO CR Block

This selective attribute consists of the following attributes:

NOTE A certain CR can either be an IOCR or an Alarm CR.

IO CR Type

This attribute indicates the type of this CR.

Attribute Type: Unsigned16

Allowed values: INPUT_CR, OUTPUT_CR, MULTICAST_PROVIDER_CR, MULTICAST_CONSUMER_CR

IO CR Reference

This attribute contains the identifier for the CR.

Attribute Type: Unsigned16

LT Field

This attribute contains the length type field of the Physical Layer Frame.

Attribute Type: Unsigned16

Allowed values: 0-0x8891 reserved, 0x8892 = PROFINET IO, 0x8893 to 2¹⁶-1 reserved

IO CR Properties

This attribute consists of the following attributes:

RT Class

This attribute contains the supported real time class.

The class real-time (RT) uses optional RT_CLASS_1 and mandatory RT_CLASS_2. The class isochronous real-time (IRT) uses in addition RT_CLASS_3.

Attribute Type: Unsigned8

Allowed values: RT_CLASS_1, RT_CLASS_2, RT_CLASS_3

Interface Type

This attribute contains the type of the local interface.

Attribute Type: Unsigned8

Allowed values: DETERMINED_BY_IODEVICE – operation of the Input or Output Data ASE interface is local matter, it may operate synchronized or non synchronized

Other values are reserved for future versions.

Send Clock Synchronisation

This attribute contains the type of the synchronisation of the Send Clock. In this version, this feature is a local device characteristic for all CR's and not an individual characteristic of a single CR. Therefore, it shall not differ between CR's and is not changeable by the IO controller during connection establishment.

Attribute Type: Unsigned8

Allowed values: FLOATING_LOCAL – no synchronization between IO controller and IO device. It is the only value that shall be set by the IO controller during AR establishment,

LOCAL_PROVIDER – the IO device synchronize its SendClock with the CycleCounter of those related IO controller which has established the first AR. It is only usable with the ReductionRatio is greater or equal 4. LOCAL_PROVIDER is only a local device characteristic and shall be equal for all CR's. If set, it overrules the IO controller setting (FLOATING_LOCAL).

Address Resolution

This attribute is not used in this version. It contains the used protocol for address resolution for IO device to IO device communication in future versions.

Attribute Type: Unsigned8

Allowed values: reserved in this version

MProvider Data Status

This attribute shall contain a control flag for the field data status of an multicast provider during AR establishment. The values shall be according to the Table 103.

Table 103 — MProvider Data Status

Value	Meaning
INVALID	Data status of multicast provider frame is invalid in start up phase till Application Ready
VALID	Data status of multicast provider frame is valid in start up phase

C_SDU Length

This attribute contains the length of the transmitted Data (CSDU Length).

Attribute Type: Unsigned16

Allowed values: 40 to 1440

Frame ID

This attribute contains the identifier of the data within the frame. The attribute value is provided by the consumer for Output or Input CRs. It is provided by project planning tool for M_CR.

Attribute Type: Unsigned16

The allowed values according to Table 104 shall be used.

Table 104 — Frame ID

Value (hexadecimal)	Meaning	Use
0x0000 – 0x001F	PTP sync	PTP synchronization
0x0020 – 0x003F	PTP delay request	PTP synchronization
0x0040 – 0x005F	PTP additional delay request	PTP synchronization
0x0060 – 0x007F	PTP additional delay response	PTP synchronization
0x0080 – 0x009F	PTP sync for RT class 3	IRT PTP synchronization
0x00A0 – 0x00FF	reserved	reserved
0x0100 – 0x7FFF	dedicated to RT Class 3 unicast and multicast	reserved for RTOverUDP
0x8000 – 0xBEFF	dedicated to RT Class 2 unicast	reserved for RTOverUDP
0xBF00 – 0xBFFF	dedicated to RT Class 2 multicast	reserved for RTOverUDP
0xC000 – 0xFAFF	dedicated to RT Class 1 unicast	RTOverUDP
0xFB00 – 0xFBFF	dedicated to RT Class 1 multicast	RTOverUDP
0xFC00	reserved	
0xFC01	Alarm High	RTOverUDP
0xFC02 – 0xFDFF	reserved	
0xFE00	reserved	
0xFE01	Alarm Low	RTOverUDP
0xFE02 – 0xFEFC	reserved	
0xFEFD – 0xFEFF	reserved for DCP	
0xFF00 – 0xFF1F	PTP follow up	PTP synchronization
0xFF20 – 0xFF3F	PTP delay response	PTP synchronization
0xFF40- 0xFF5F	PTP additional delay followup request	Precision time protocol synchronization
0xFF60	MRP	Medium redundancy protocol
0xFF61 – 0xFFFF	reserved	reserved

Send Clock Factor

This attribute contains the Send Clock Factor.

Attribute Type: Unsigned16

Time Base: 31,25 µs

The values shall be in the range of 1 to 128 (default 32)

Reduction Ratio

This attribute contains the reduction ratio of the send clock of the device. The actual send cycle of the data set is Reduction Ratio multiplied with the send clock. It defines a macro cycle for this particular CR.

Attribute Type: Unsigned16

The following values shall be used for RT_CLASS_1_TRANSPORT: 1 to 512

The values 32,64,128,256,512 are optional for RT_CLASS_3. Furthermore, the value 256 shall only be used for a Send Clock Factor up to 64 if RT_CLASS_1_TRANSPORT is set. The value 512 shall only be used for a Send Clock Factor up to 32 if RT_CLASS_1_TRANSPORT is set.

The following values shall be used for UDP_TRANSPORT: 1 to 16384.

Phase

This attribute indicates in which particular send cycle the data has to be sent. This parameter depends from the parameter Reduction Ratio allowed values are between 1 and selected Reduction Ratio.

Attribute Type: Unsigned16

NOTE The parameter phase allows spreading the sending of the frames over the whole macro cycle and provides a means to manage distribution of the used bandwidth. The phases e.g. 1 to 4 are possible if the Reduction Ratio is set to 4.

Sequence

This attribute contains the position of the data frame in the sending queue. The value 0 shall be used to indicate an undefined sequence.

Attribute Type: Unsigned16

Frame Send Offset

This attribute contains the relative send offset to the start of the related reduction ratio and phase. The time base is 1 ns. The value shall be less than the duration of one phase.

Attribute Type: Unsigned32

Allowed Values: BEST_EFFORT – (mandatory) as soon as possible, 0-0x003D08FF for optimized scheduling

Watchdog Factor

This attribute contains a factor, which is used to calculate the Watchdog Time. The interval for the Watchdog Time is the product of the attribute values of Watchdog Factor, Send Clock Factor, Reduction Ratio and the time base of 31,25 µs related to the monitored consumer. The Watchdog Time shall equal or less 1,92 s if the value of the AR attribute CR Transport is RT_CLASS_1_TRANSPORT and equal or less 49,152 s if the value of the AR attribute CR Transport is UDP_TRANSPORT.

Attribute Type: Unsigned16

Allowed values: 0x0001 to 0x1E00

Data Hold Factor

This attribute contains a factor, which is used to calculate the Data Hold Time. This time is the time interval for holding last output data if no IO data have been received. For non-redundant AR's this value should be equal to the value of the attribute Watchdog Factor. Time. The interval for the Data Hold Time is the product of the attribute values of Data Hold Factor, Send Clock Factor, Reduction Ratio and the time base of 31,25 µs related to the monitored consumer. The Data Hold Time shall equal or less 1,92 s if the value of the AR attribute CR Transport is RT_CLASS_1_TRANSPORT and equal or less 49,152 s if the value of the AR attribute CR Transport is UDP_TRANSPORT.

Attribute Type: Unsigned16

Allowed values: 0x0001 to 0x1E00

IO CR Tag Header

This attribute consists of the following attributes:

Vlan ID

This attribute contains the IEEE 802.1Q VLAN ID. The default value is NO_VLANID that indicates that the tag header contains only priority information.

Allowed values: according to IEEE 802.1Q

IO CR User Priority

This attribute contains the IEEE 802.1Q VLAN ID priority. The default value is IOCR_PRIORITY.

Allowed values: IOCR_PRIORITY – higher priority than IP traffic

IO CR Multicast MAC Add

This attribute contains the multicast address according ISO/IEC 8802.3 in case of multicast transmission. The value shall be set to zero for unicast connections.

Attribute Type: OctetString[6]

List of APIs

This attribute list shall contain at least one API that the CR belongs to. A list element consists of the following attributes:

API

This attribute defines to which application process the IOCR belongs to.

Attribute Type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

List of Related IO Data Objects

This attribute contains the IO Data Objects that are related to the CR. A List element consists of the following attributes:

Slot Number

This attribute defines to which slot the IO Data belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines to which subslot the IO Data belongs.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

IO Data Object Frame Offset

This attribute defines the offset of the DataItem. It shall be in a range off 0 to 1439. The maximum number of used octets shall not exceed the maximum C_SDU size.

Attribute Type: Unsigned16

List of Related IOCS

This attribute contains the IOCS that are related to the CR. A list element consists of the following attributes:

Slot Number

This attribute defines to which slot the IOCS belongs.

Attribute Type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines to which subslot the IOCS belongs.

Attribute Type: Unsigned16

Allowed values: 1 to 0x8FFF are allowed

IOCS Frame Offset

This attribute defines the offset of the IOCS of the DataItem. It shall be in a range off 0 to 1439. The maximum number of used octets shall not exceed the maximum C_SDU size.

Attribute Type: Unsigned16

Multicast Consumer Data

This attribute shall be present if the attribute IO CR Type contains the value MULTICAST_CONSUMER_CR and consists of the following attributes:

Address Resolution Properties

This attribute defines the protocol and interval for address resolution of the multicast provider. It includes the waiting time between two subsequent address resolution cycles.

Attribute Type: Unsigned32

Allowed values for protocol: DNS, DCP

Allowed values for interval: 0x0001 to 0xFFFF with a time base of 100 ms

MCI Timeout Factor

This attribute defines the monitoring interval for address resolution and in data state of the multicast provider. The application ready shall be delayed until the in data state is reached or the Multicast Initiator Timeout is reached.

Attribute Type: Unsigned16

Allowed values for interval: 0x0000 to 0x0064 with a time base of 100 ms

Provider Station Name

This attribute defines the IP address or name of the Multicast Provider.

This field shall be coded as data type OctetString with 1 to 240 octets. The values shall be "a"-"z", "0"-"9", ".", and "-". It shall not start with ".", or "-". The last character shall not be "-" or ".". Between two "." 2 up to 64 octets shall be used. The top level domain is right most in the string.

EXAMPLE "device-1.machine-1.plant-1.vendor"

Alarm CR Block

This selective attribute consists of the following attributes:

NOTE A certain CR can either be an IOCR or an Alarm CR.

Alarm CR Type

This attribute indicates the type of this CR.

Attribute Type: Unsigned16

Allowed value: ALARM_CR

LT Field

This attribute contains the length type field of the Physical Layer Frame.

Attribute Type: Unsigned16

Allowed values: 0-0x8891 reserved, 0x8892 = PROFINET IO, 0x8893 to $2^{16}-1$ reserved

Alarm CR Properties

This attribute consists of the following attributes:

Attribute Type: Unsigned16

Priority

This attribute contains the priority of the alarm.

Allowed Values: USER_PRIORITY (default) – the priority given by the user is used and two alarm resources are available,

LOW_PRIORITY – user priority is ignored and only one alarm resource is available

RTA Timeout Factor

This attribute indicates the timeout value of the alarm source with a base of 100 ms. The default value is 100 ms.

Attribute Type: Unsigned16

Time Base: 100 ms

Allowed Values: 1 up to 100 mandatory, 101 to $2^{16}-1$ optional

RTA Retries

This attribute contains the number of retries. The value range is from 3 to 15 retries.

Attribute Type: Unsigned16

Local Alarm Reference

This attribute contains the identifier of the CM Initiator (e.g. IO controller).

Attribute Type: Unsigned16

Remote Alarm Reference

This attribute contains the identifier of the CM Responder (e.g. IO device).

Attribute Type: Unsigned16

Max Alarm Data Length

This attribute contains the length of the alarm data. This attribute shall be set accordingly with the Load CRL service.

Attribute Type: Unsigned16

Allowed value: 200 to 1432

Alarm CR Tag Header High

This attribute consists of the following attributes:

Vlan ID

This attribute contains the IEEE 802.1Q VLAN ID. The default value is NO_VLAN that indicates that the tag header contains only priority information.

Allowed values: according to IEEE 802.1Q

Alarm User Priority

This attribute contains the priority of the alarm.

Allowed Values: USER_PRIORITY_HIGH

Alarm Tag Header Low

This attribute consists of the following attributes:

Vlan ID

This attribute contains the IEEE 802.1Q VLAN ID. The default value is NO_VLAN that indicates that the tag header contains only priority information.

Allowed values: according to IEEE 802.1Q

Alarm User Priority

This attribute contains the priority of the alarm.

Allowed Values: USER_PRIORITY_LOW

4.3.2.10.4.1.3 Invocation of CRL object

For the invocation of the CRL object the following rule has to be considered:

- Only one CRL object shall be invoked in a IO device.

4.3.2.10.5 AR service specification**4.3.2.10.5.1 Read AR Data**

This confirmed service may be used to read the value of a AR Endpoint object. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR. Table 105 shows the parameters of the service.

Table 105 — Read AR Data

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
AREP	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
List of ARs			M	M(=)
AR UUID			M	M(=)
AR Type			M	M(=)
AR Properties			M	M(=)
CM Initiator Object UUID			M	M(=)
CM Initiator Station Name			M	M(=)
List of IOCRs			M	M(=)
IO CR Type			M	M(=)
IO CR Properties			M	M(=)
Frame ID			M	M(=)
APDU Status			M	M(=)
UDP CM Initiator Port			U	U(=)
UDP CM Responder Port			U	U(=)
Alarm CR Type			M	M(=)
Local Alarm Reference			M	M(=)
Remote Alarm Reference			M	M(=)
Parameter Server Object UUID			M	M(=)
Parameter Server Station Name			M	M(=)
List of APIs			M	M(=)
API			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Error Decode			M	M(=)
Error Code 1			M	M(=)
Error Code 2			M	M(=)
Add Data 1			M	M(=)
Add Data 2			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

AREP

This parameter is the local identifier for the desired AR.

Seq Number

The parameter Seq Number is used by the server to identify the duplication of services with a sequence number. The range of this service parameter is from 0 to $2^{16} - 1$. The requesting application process shall provide a unique Seq Number to each outstanding service request. The value parameter Seq Number shall be incremented for each service request during a session. Starting a new session the sequence number shall start with the last value of the session before. Confirmations with a Seq Number that is not outstanding will be ignored. Indications with a Seq Number that is not outstanding will be rejected. The Seq Number shall be maintained for each established AR separately.

Length

The parameter Length indicates the number of octets of a Alarm Data that has to be read. The allowed length is from 2^0 to $2^{32}-256$.

Result(+)

This parameter indicates that the service request succeeded.

List of ARs

This parameter is composed of the following list elements:

AR UUID

This parameter contains the value of the corresponding attribute of the ASE object.

AR Type

This parameter contains the value of the corresponding attribute of the ASE object.

AR Properties

This parameter contains the value of the corresponding attribute of the ASE object.

CM Initiator Object UUID

This parameter contains the value of the corresponding attribute of the Context ASE object.

CM Initiator Station Name

This parameter contains the value of the corresponding attribute of the ASE object.

List of IOCRs

This parameter list is composed of the following list elements:

IO CR Type

This parameter contains the value of the corresponding attribute of the ASE object.

IO CR Properties

This parameter contains the value of the corresponding attribute of the ASE object.

Frame ID

This parameter contains the value of the corresponding attribute of the ASE object.

APDU Status

This parameter contains the value of the corresponding attribute of the ASE object.

UDP CM Initiator Port

This parameter contains the value of the corresponding attribute of the ASE object.

UDP CM Responder Port

This parameter contains the value of the corresponding attribute of the ASE object.

Alarm CR Type

This parameter contains the value of the corresponding attribute of the ASE object.

Local Alarm Reference

This parameter contains the value of the corresponding attribute of the ASE object.

Remote Alarm Reference

This parameter contains the value of the corresponding attribute of the ASE object.

Parameter Server Object UUID

This parameter contains the value of the corresponding attribute of the ASE object.

Parameter Server Station Name

This parameter contains the value of the corresponding attribute of the ASE object.

List of APIs

This parameter list is composed of the following list elements:

API

This parameter contains the value of the corresponding attribute of the ASE object.

Result(-)

This parameter indicates that the service request failed.

Error Decode

This parameter selects one Error scheme for Error Code 1 and 2 according IEC 61158-5 Type 3 specification.

Type: Unsigned8

Allowed Value: PNORW

Error Code 1

The Error Code 1 assumes one of the following values:

read error, module failure, version conflict, feature not supported, user specific, invalid index, invalid slot/subslot, type conflict, invalid area, state conflict, access denied, invalid range, invalid parameter, invalid type, read constrain conflict, resource busy, resource unavailable, service cancelled.

Type: Unsigned16

Error Code 2

The parameter Error Code 2 is user specific.

Type: Unsigned8

Add Data 1

The parameter Add Data is API specific (profile). The value 0 shall be transmitted if no additional data 1 is defined.

Type: Unsigned 16

NOTE Add Data 1 may be used by profile specifications to transmit specific error messages.

Add Data 2

The parameter Add Data is user specific. The value 0 shall be transmitted if no additional data 2 is defined.

Type: Unsigned 16

NOTE Add Data 2 may be used by device vendors to transmit specific error messages.

4.3.3 Behavior of an IO device**4.3.3.1 Overview**

Each IO Device shall have the following ASE state machines:

- Startup SM (RT Class 1 and 2)
- Submodule SM
- Alarm Queue SM

NOTE Actually, the Alarm Queue SM is just an instance storing alarms in order to bring them in a coordinated sequence rather than a real state machine.

Each IO Device supporting RT Class 3 shall additionally have the following ASE state machine:

- Startup SM (RT Class 3)

Each IO Device supporting Isochronous Mode ASE shall additionally have the following ASE state machine:

- Input SM
- Output SM
- SyncCtl SM

Figure 36 shows the overview of the ASE machines.

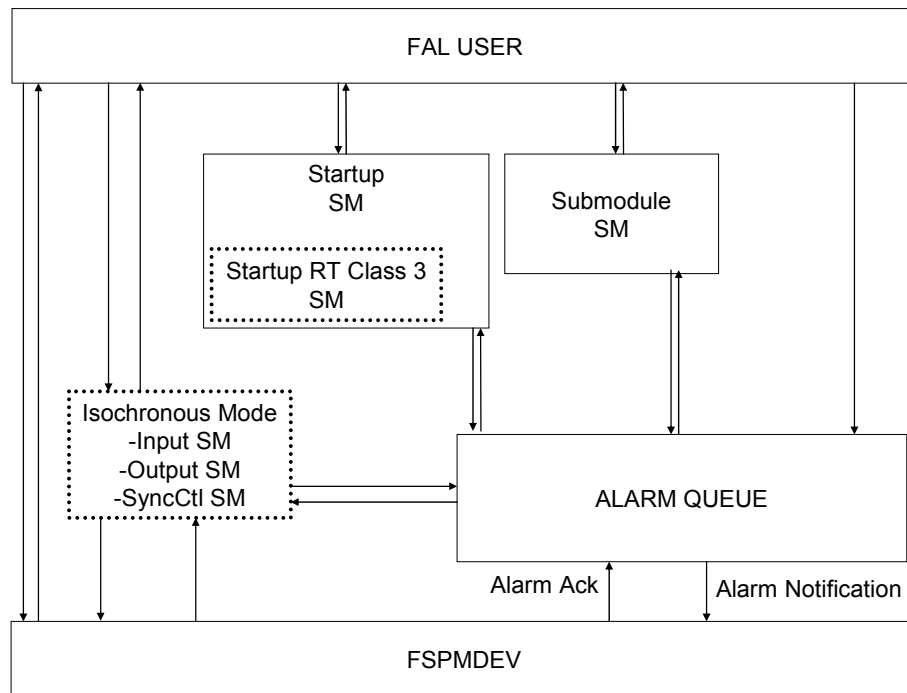


Figure 36 — Overview ASE state machines for IO device

4.3.3.2 Startup of an IO device (RT class 1 and 2)

This subclause describes the application behavior of an IO device during startup of the user application.

It includes extracts of the behavior of the following ASEs:

Context ASE
Record Data ASE
IO Data ASE

NOTE This subclause provides selected behavior only and is not a comprehensive definition of the object behavior of the above mentioned ASEs.

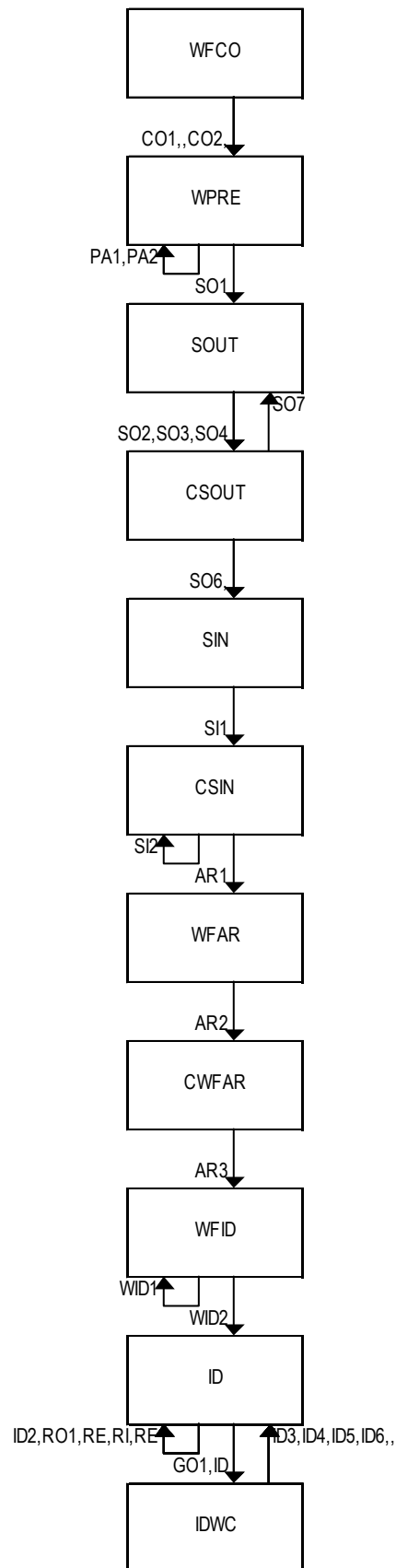


Figure 37 — State diagram application startup IO device

Figure 37 shows the state diagram and Table 106 defines the state table for the selected behavior.

Table 106 — State table application startup IO device (RT class 1 and 2)

#	Current State	Event /Condition =>Action	Next State
1	WFCO	Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) /CheckExpectedCnf()==Ok => SetConnectResponseData("Zero") StoredSessionKey:=ARBlockReq.SessionKey Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	WPRE
2	WFCO	Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq) /CheckExpectedCnf()==Ok => SetConnectResponseData("Zero") StoredSessionKey:=ARBlockReq.SessionKey Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	WPRE
3	WFCO	Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) /CheckExpectedCnf()!=Ok => DeltaConfiguration=AdaptConfiguration() StoredSessionKey:=ARBlockReq.SessionKey SetConnectResponseData(DeltaConfiguration) Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	WPRE
4	WFCO	Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq) /CheckExpectedCnf()!=Ok => DeltaConfiguration=AdaptConfiguration() StoredSessionKey:=ARBlockReq.SessionKey SetConnectResponseData(DeltaConfiguration) Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	WPRE
5	WPRE	Write.ind(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) /CheckParameter()==Ok => StoreParameter() Write.rsp(+)(AREP)	WPRE
6	WPRE	Write.ind(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) /CheckParameter()!=Ok => SetWrErrorResponseData("ErrorCode=PNIORW", OTHER=UserSpecific) Write.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	WPRE
7	WPRE	EndOfParameter.ind(AREP, SessionKey) => EndOfParameter.rsp(+) (AREP, SessionKey)	SOUT
8	SOUT	/Submodule Properties.Safe State Behavior == REPLACEMENT_VALUE => For all involved slot/subslots do: Slot.Subslot.IOPS := BAD_BY_DEVICE Output Data object subelement value:=Slot.Subslot.Safe State Value Slot.Subslot.IOCS:=GOOD For all involved slot/subslots do: Set Output IOCS.req (Slot,Subslot,IOCS)	CSOUT

#	Current State	Event /Condition =>Action	Next State
9	SOUT	/Submodule Properties.Safe State Behavior == LAST_ VALUE => For all involved slot/subslots do: Slot.Subslot.IOPS := BAD_BY_DEVICE Output Data object subelement value:="local initial value" Slot.Subslot.IOCS:=GOOD For all involved slot/subslots do: Set Output IOCS.req (Slot,Subslot,IOCS)	CSOUT
10	SOUT	/Submodule Properties.Safe State Behavior == ZERO => For all involved slot/subslots do: Slot.Subslot.IOPS := BAD_BY_DEVICE Output Data object subelement value:=ZERO Slot.Subslot.IOCS:=GOOD For all involved slot/subslots do: Set Output IOCS.req (Slot,Subslot,IOCS)	CSOUT
11	CSOUT	Set Output IOCS.cnf(+) /not last one =>	SOUT
12	CSOUT	Set Output IOCS.cnf(+) /last one =>	SIN
13	CSOUT	Set Output IOCS.cnf(-) => ignore (M-Consumer submodule)	SIN
14	SIN	=> For all involved slot/subslots with submodul status GOOD do: Slot := Slot Number; Subslot := Subslot Number IOPS := GOOD Subslot Input Data:=Input Data object subelement value local IOCS:=GOOD For all involved slot/subslots with submodul status BAD do: Slot := Slot Number; Subslot := Subslot Number IOPS := BAD_BY_DEVICE local IOCS:=BAD_BY_DEVICE For all involved slot/subslots do: Set Input.req (Slot,Subslot, IOPS, Subslot Input Data)	CSIN
15	CSIN	Set Input.cnf(+) /not last one => ignore	CSIN
16	CSIN	Set Input.cnf(+) /last one =>	WFAR
17	WFAR	/UserReady => SessionKey:=StoredSessionKey ApplicationReady.req(AREP, SessionKey)	CWFAR
18	CWFAR	ApplicationReady.cnf(+)(AREP,SessionKey) =>	WFID

#	Current State	Event /Condition =>Action	Next State
19	WFID	In_Data.ind(CREP.CPM) New_Data.ind(CREP.CPM, APDU_Status) /not last one => ignore	WFID
20	WFID	In_Data.ind(CREP.CPM) New_Data.ind(CREP.CPM, APDU_Status) /last one (exclude CREPs with Multicast Communication) =>	ID
21	ID	In_Data.ind(CREP.CPM) New_Data.ind(CREP.CPM, APDU_Status) /(CREP with Multicast Communication) && (CHECK_IOPS(CREP) == TRUE) => Alarm Priority:=ALARM_HIGH or ALARM_LOW Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties := "User Specific" ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR Okay Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, User Structure Identifier)	ALWC
22	ID	In_Data.ind(CREP.CPM) New_Data.ind(CREP.CPM, APDU_Status) /(CREP with Multicast Communication) && (CHECK_IOPS(CREP) == FALSE) => Alarm Priority:=ALARM_HIGH or ALARM_LOW Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties := "User Specific" ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR Okay with M-Provider Status Mismatch Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, User Structure Identifier)	ALWC
23	ID	New Output.ind(Slot, Subslot, Watchdog Flag) /Watchdog Flag==WATCHDOG_EXPIRED => "real output values are not modified, possible actions are local matter or for future use with redundancy"	ID
24	ID	M-Provider Communication Stopped.ind(Slot Number, Subslot Number) => Alarm Priority:=ALARM_HIGH or ALARM_LOW Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties := "User Specific" ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out Alarm Notification.req(AREP, API, Alarm Priority, Alarm Type, Slot Number, Subslot Number, Alarm Specifier, Module Ident Number, Submodule Ident Number, User Structure Identifier)	ALWC
25	ALWC	Alarm Notification.cnf(+)(AREP) =>	ID

#	Current State	Event /Condition =>Action	Next State
26	ID	Alarm Ack.ind(AREP, API, Alarm Type, Slot Number, Subslot Number, Alarm Specifier) => ignore	ID
27	ID	New Output.ind(Slot, Subslot) /!Watchdog Flag => Get Output.req(Slot,Subslot)	IDWC
28	IDWC	Get Output.cnf(+)(Slot, Subslot, IOPS,Subslot Output Data,New Flag, IOCS) /IOPS==GOOD => Slot.Subslot.IOPS := GOOD Output Data object subelement value:=Output Data	ID
29	IDWC	Get Output.cnf(+)(Slot, Subslot, IOPS,Subslot Output Data,New Flag, IOCS) /IOPS==BAD_BY_CONTROLLER & Submodule Properties.Safe State Behavior == REPLACEMENT_VALUE => Slot.Subslot.IOPS := BAD_BY_CONTROLLER Output Data object subelement value:=Slot.Subslot.Safe State Value	ID
30	IDWC	Get Output.cnf(+)(Slot, Subslot, IOPS,Subslot Output Data,New Flag, IOCS) /IOPS==BAD_BY_CONTROLLER & Submodule Properties.Safe State Behavior == LAST_VALUE => Slot.Subslot.IOPS := BAD_BY_CONTROLLER	ID
31	IDWC	Get Output.cnf(+)(Slot, Subslot, IOPS,Subslot Output Data,New Flag, IOCS) /IOPS==BAD_BY_CONTROLLER & Submodule Properties.Safe State Behavior == ZERO => Slot.Subslot.IOPS := BAD_BY_CONTROLLER Output Data object subelement value:=ZERO	ID
32	IDWC	Get Output.cnf(+)(Slot, Subslot, IOPS,Subslot Output Data,New Flag, IOCS) /IOPS==BAD_BY_PROVIDER & Submodule Properties.Safe State Behavior == REPLACEMENT_VALUE/LAST_VALUE/ZERO => Slot.Subslot.IOPS := BAD_BY_PROVIDER Output Data object subelement value:=REPLACEMENT_VALUE/LAST_VALUE/ZERO SubmoduleState.CommState := Multicast Provider IOPS bad Alarm Priority:=ALARM_HIGH or ALARM_LOW Alarm Type:=Multicast Communication Mismatch Alarm UserStructureIdentifier := MULTIPLE ChannelNumber := 0x8000 ChannelProperties := "User Specific" ChannelErrorType := Multicast Provider Status Mismatch ExtChannelErrorType := IOPS bad conveyed by Multicast Provider Set_AlarmNotification.ind (AREP, CREP, Alarm Priority, Alarm Type, UserStructureIdentifier, ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType)	ID

#	Current State	Event /Condition =>Action	Next State
33	IDWC	Get Output.cnf(+)(Slot, Subslot, IOPS,Subslot Output Data,New Flag, IOCS) /IOPS==GOOD && SubmoduleState.CommState := Multicast Provider IOPS bad => Slot.Subslot.IOPS := GOOD Output Data object subelement value:=Output Data SubmoduleState.CommState := OK Alarm Priority:=ALARM_HIGH or ALARM_LOW Alarm Type:=Multicast Communication Mismatch Alarm UserStructureIdentifier := MULTIPLE ChannelNumber := 0x8000 ChannelProperties := "User Specific" ChannelErrorType := Multicast Provider Status Mismatch ExtChannelErrorType := Multicast Provider Status Good Set_AlarmNotification.ind (AREP, CREP, Alarm Priority, Alarm Type, UserStructureIdentifier, ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType)	ID
34	ID	Read Output Data.ind(AREP, ARUUID, Slot, Subslot, Length) /ARUUID==AREP.ARUUID&& Slot==AREP.Slot Number&& Subslot==AREP.Subslot Number&& ((Length Output Data+Length IOPS+Length IOCS) <=Length) => AREP:=local AR reference Length:=Slot.Subslot.Length Data+ Slot.Subslot.Length IOCS+Slot.Subslot.Length IOPS Length Data:=Slot.Subslot.Length Data Length IOCS:=Slot.Subslot.Length IOCS Length IOPS:=Slot.Subslot.Length IOPS IOCS:=Slot.Subslot.IOCS IOPS:=Slot.Subslot.IOPS Subslot Output Data:=Output Data object subelement value Read Output Data.res(+) (AREP, Length, Length Data, Length IOCS, Length IOPS, IOCS, IOPS, Subslot Output Data)	ID
35	ID	Read Output Data.ind(AREP, ARUUID, Slot, Subslot, Length) /!(ARUUID==AREP.ARUUID&& Slot==AREP.Slot Number&& Subslot==AREP.Subslot Number&& ((Length Output Data+Length IOPS+Length IOCS) <=Length)) => AREP:=local AR reference Error Decode:=PNIORW Error Code 1:="parameter error" Error Code 2:=vendor specific or 0 Add Data 1:=profile specific or 0 Add Data 2:=vendor specific or 0 Read Output Data.res(-) (AREP, Error Decode, Error Code 1, Error Code 2, Add Data 1, Add Data 2)	ID
36	ID	/input value changed => For all involved slot/subslots do: Slot := Slot Number; Subslot := Subslot Number IOPS := GOOD Subslot Input Data:=Input Data object subelement value For all involved slot/subslots do: Set Input.req (Slot,Subslot, IOPS, Subslot Input Data)	IDWC

#	Current State	Event /Condition =>Action	Next State
37	ID	Read Input Data.ind(AREP, ARUUID, Slot, Subslot, Length) /ARUUID==AREP.ARUUID&& Slot==AREP.Slot Number&& Subslot==AREP.Subslot Number&& ((Length Input Data+Length IOPS+Length IOCS) <=Length) => AREP:=local AR reference Length:=Slot.Subslot.Length Data+ Slot.Subslot.Length IOCS+Slot.Subslot.Length IOPS Length Data:=Slot.Subslot.Length Data Length IOCS:=Slot.Subslot.Length IOCS Length IOPS:=Slot.Subslot.Length IOPS IOCS:=Slot.Subslot.IOCS IOPS:=Slot.Subslot.IOPS Subslot Input Data:=Input Data object subelement value Read Input Data.res(+) (AREP, Length, Length Data, Length IOCS, Length IOPS, IOCS, IOPS, Subslot Input Data)	ID
38	ID	Read Input Data.ind(AREP, ARUUID, Slot, Subslot, Length) /(ARUUID==AREP.ARUUID&& Slot==AREP.Slot Number&& Subslot==AREP.Subslot Number&& ((Length Input Data+Length IOPS+Length IOCS) <=Length)) => AREP:=local AR reference Error Decode:=PNIORW Error Code 1:="parameter error" Error Code 2:=vendor specific or 0 Add Data 1:=profile specific or 0 Add Data 2:=vendor specific or 0 Read Input Data.res(-) (AREP, Error Decode, Error Code 1, Error Code 2, Add Data 1, Add Data 2)	ID

4.3.3.3 Startup of an IO device (RT class 3)

The startup for an IO device shall be done in two phases establishing two ARs in a sequence. The establishment of both connections shall comply with the model described in 4.3.3.2. The first AR shall be connected to interface and port modules, which are involved in the RT class 2 communication. The parameter IR data and Sync data shall be set in the IO device. If the clock synchronization is reached then the second AR is established for the RT class 3 related data exchange.

Both ARs shall use ARUUIDs with a well-defined difference of only 1 nsec to express the sequence and the strong cohesion. The AR of the first phase shall be 1 nsec older.

4.3.3.4 Diagnosis and problem indicator

This subclause describes the application interaction within an IO device concerning diagnosis information.

It includes extracts of the behavior of the following ASEs:

Diagnosis ASE IO Data ASE

There is a means to notify the presence of diagnosis information within IO data transfer referred to as "Problem Indicator" as part of the APDU status. The problem indicator shall be set as long as the Diagnosis ASE contain at least one diagnosis entry that affects the related AR. The IO data ASE provides services to set and get this parameter.

4.3.3.5 User alarms

This subclause describes the application interaction within an IO device concerning alarms that are caused by diagnosis, process, status, update or other manufacturer specific events. The notification of such alarms is manufacturer specific.

It includes extracts of the behavior of the following ASEs:

Alarm ASE

Diagnosis ASE (partly)

The Diagnosis Alarm is a means to notify the event if diagnosis information appears or disappears. It is strongly recommended to use channel diagnosis.

The general behavior of alarms is described in 4.3.2.5.4.

4.3.3.6 Behavior in case of configuration change

4.3.3.6.1 General

The behavior in case of pull or plug of modules and/or submodules, IO supervisor control and release during operation and AR establishment and release is referred to as configuration change.

The behavior in case of configuration change is defined as described below.

If a module is pulled the IOPS and IOCS for each subslot that belongs to the slot shall be set to bad detected by device. Furthermore, the value 0 for the parameter subslot implicitly covers all other subslots that belong to the slot in case of pull. If a submodule is pulled then a dedicated alarm notification shall be issued for the specific submodule. In any case the IO controller shall acknowledge the alarm.

4.3.3.7 State behavior of submodules

This subclause describes the application behavior of an IO device in case of plugging a module or submodule, change of parametrization data, if the IOPS or IOCS changes from BAD to GOOD, or if a submodule is released by an IO supervisor.

If there is more than one IO controller connected to a certain submodule and this module is e.g. released by an IO supervisor then always the first connected IO controller shall get the control back (first serve).

It includes extracts of the behavior of the following ASEs:

Alarm ASE
Context ASE
Record Data ASE
IO Data ASE

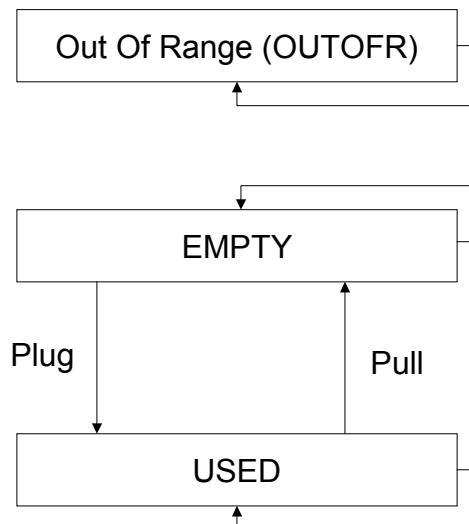


Figure 38 — State diagram for a submodule

Figure 38 shows the state diagram and Table 107 defines the state table for the selected behavior.

Table 107 — State table for a submodule

#	Current State	Event /Condition =>Action	Next State
1	OUTOFR	UserPlugSM.req(RealModulID, RealSubmodulID, API, Slot, Subslot) => UserPlugSM.cnf(-)	OUTOFR
2	EMPTY	UserPlugSM.req(RealModulID, RealSubmodulID, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Lock) => UserPlugSM.cnf(+)	USED
3	EMPTY	UserPlugSM.req(RealModulID, RealSubmodulID, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) => UserPlugSM.cnf(+)	USED
4	EMPTY	UserPlugSM.req(RealModulID, RealSubmodulID, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (WRONG = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => UserPlugWrongAlarm.ind(ANY.AREP, API, Slot, Subslot); UserPlugSM.cnf(+)	USED
5	EMPTY	UserPlugSM.req(RealModulID, RealSubmodulID, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && ((Substitute OK) = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => UserPlugAlarm.ind(ANY.AREP, API, Slot, Subslot); UserPlugSM.cnf(+)	USED
6	USED	UserPlugSM.req(RealModulID, RealSubmodulID, API, Slot, Subslot) => UserPlugSM.cnf(-)	USED
7	OUTOFR	UserPullSM.req(API, Slot, Subslot) => UserPullSM.cnf(-)	OUTOFR
8	EMPTY	UserPullSM.req(API, Slot, Subslot) => UserPullSM.cnf(-)	EMPTY
9	USED	UserPullSM.req(API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Lock) (OK_Queue.FirstEntry = EMPTY) => SubmoduleFlags.State = Disable; SubmoduleFlags.ARInfo = OK; SubmoduleFlags.DiagInfo = Disable; UserPullSM.cnf(+)	EMPTY
10	USED	UserPullSM.req(API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) => SubmoduleFlags.State = Disable; SubmoduleFlags.ARInfo = OK; SubmoduleFlags.DiagInfo = Disable; UserPullAlarm.ind(ANY.AREP, API, Slot, Subslot); UserPullSM.cnf(+)	EMPTY
11	OUTOFR	UserAlarmSend.req(AREP, API, Slot, Subslot) => UserAlarmSend.cnf(-)	OUTOFR
12	EMPTY	UserAlarmSend.req(AREP, API, Slot, Subslot) => UserAlarmSend.cnf(-)	EMPTY
13	USED	UserAlarmSend.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Lock) => UserAlarmSend.cnf(-)	USED
14	OUTOFR	UserSetSMState.req(Command, API, Slot, Subslot) => UserSetSMState.cnf(-)	OUTOFR

#	Current State	Event /Condition =>Action	Next State
15	EMPTY	UserSetSMState.req(Command, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) && (Command.Superordinated = SET) => SubmoduleFlags.Superordinated = Lock UserSetSMState.cnf(+)	EMPTY
16	EMPTY	UserSetSMState.req(Command, API, Slot, Subslot) /(OK_Queue.FirstEntry = ANY) && (Command.Superordinated = SET) => UserSetSMState.cnf(-)	EMPTY
17	EMPTY	UserSetSMState.req(Command, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) && (Command.Superordinated = RESET) => SubmoduleFlags.Superordinated = Free UserSetSMState.cnf(+)	EMPTY
18	EMPTY	UserSetSMState.req(Command, API, Slot, Subslot) /(OK_Queue.FirstEntry = ANY) && (Command.Superordinated = RESET) => SubmoduleFlags.Superordinated = Free UserPullAlarm.ind(ANY.AREP, API, Slot, Subslot); UserSetSMState.cnf(+);	EMPTY
19	EMPTY	UserSetSMState.req(Command, API, Slot, Subslot) /Command <> Superordinated => UserSetSMState.cnf(-)	EMPTY
20	USED	UserSetSMState.req(Command, API, Slot, Subslot) /((SubmoduleFlags.Superordinated = Lock) (OK_Queue.FirstEntry = EMPTY)) && ((Command.State = ARP) (Command.State = Disable) (Command.State = OK)) => UserSetSMState.cnf(-)	USED
21	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (Command.State = ARP) => SubmoduleFlags.ARInfo = ARP; SubmoduleFlags.State = Disable; UserSetIOXSBad.ind(ANY.AREP, API, Slot, Subslot); UserSetSMState.cnf(+)	USED
22	USED	UserSetSMState.req(Command, API, Slot, Subslot) /Command.Diag = SET => SubmoduleFlags.DiagInfo = Diag UserSetSMState.cnf(+)	USED
23	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) && (Command.Superordinated = SET) => SubmoduleFlags.Superordinated = Lock UserSetSMState.cnf(+)	USED
24	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(OK_Queue.FirstEntry = ANY) && (Command.Superordinated = SET) => UserSetSMState.cnf(-)	USED
25	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (SubmoduleFlags.ARInfo = ARP) && (Command.State = OK) => SubmoduleFlags.ARInfo = OK; SubmoduleFlags.State = Enable; UserReleasedAlarm.ind(ANY.AREP, API, Slot, Subslot, UserInfo); UserSetSMState.cnf(+)	USED
26	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (SubmoduleFlags.ARInfo = ARP) && (Command.State = Disable) => SubmoduleFlags.ARInfo = OK; SubmoduleFlags.State = Disable; UserSetSMState.cnf(+)	USED

#	Current State	Event /Condition =>Action	Next State
27	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (SubmoduleFlags.ARInfo = OK) && (SubmoduleFlags.State = Disable) && (Command.State = OK) => SubmoduleFlags.State = Enable; UserReturnOfSMAAlarm.ind(ANY.AREP, API, Slot, Subslot); UserSetSMState.cnf(+)	USED
28	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (SubmoduleFlags.ARInfo = OK) && (SubmoduleFlags.State = OK) && (Command.State = Disable) => SubmoduleFlags.State = Disable; UserSetIOXSBad.ind(ANY.AREP, API, Slot, Subslot); UserSetSMState.cnf(+)	USED
29	USED	UserSetSMState.req(Command, API, Slot, Subslot) /Command.Diag = RESET => SubmoduleFlags.DiagInfo = OK UserSetSMState.cnf(+)	USED
30	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(Command.Superordinated = RESET) && (OK_Queue.FirstEntry = EMPTY) => SubmoduleFlags.Superordinated = Free UserSetSMState.cnf(+)	USED
31	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(Command.Superordinated = RESET) && (OK_Queue.FirstEntry = ANY) && (WRONG = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => SubmoduleFlags.Superordinated = Free UserPlugWrongAlarm.ind(ANY.AREP, API, Slot, Subslot); UserSetSMState.cnf(+)	USED
32	USED	UserSetSMState.req(Command, API, Slot, Subslot) /(Command.Superordinated = RESET) && (OK_Queue.FirstEntry = ANY) && ((Substitute OK) = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => SubmoduleFlags.Superordinated = Free UserReleasedAlarm.ind(ANY.AREP, API, Slot, Subslot); UserSetSMState.cnf(+)	USED
33	OUTOFR	UserRecordReceived.req(AREP, Type, API, Slot, Subslot, Index) => UserRecordReceived.cnf(-)	OUTOFR
34	EMPTY	UserRecordReceived.req(AREP, Type, API, Slot, Subslot, Index) /(Index < 0x8000) (Type = Write) => UserRecordReceived.cnf(-)	EMPTY
35	EMPTY	UserRecordReceived.req(AREP, Type, API, Slot, Subslot, Index) /(Type = Read) && (Index >= 0x8000) => UserRecordReceived.cnf(+)	EMPTY
36	USED	UserRecordReceived.req(AREP, Type, API, Slot, Subslot, Index) /((SubmoduleFlags.Superordinated = Lock) (OK_Queue.FirstEntry = EMPTY) (ANY.AREP <> AREP)) && (Type = Write) => UserRecordReceived.cnf(-)	USED
37	USED	UserRecordReceived.req(AREP, Type, API, Slot, Subslot, Index) / (Type = Read) => UserRecordReceived.cnf(+)	USED

#	Current State	Event /Condition =>Action	Next State
38	USED	UserRecordReceived.req(AREP, Type, API, Slot, Subslot, Index) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) && (Type = Write) => UserRecordReceived.cnf(+)	USED
39	OUTOFR	UserAlarmReceived.req(AREP, API, Slot, Subslot) => UserAlarmReceived.cnf(-)	OUTOFR
40	EMPTY	UserAlarmReceived.req(AREP, API, Slot, Subslot) => UserAlarmReceived.cnf(-)	EMPTY
41	USED	UserAlarmReceived.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Lock) (OK_Queue.FirstEntry = EMPTY) (OK_Queue.FirstEntry.AREP <> (AREP)) => UserAlarmReceived.cnf(-)	USED
42	USED	UserAlarmReceived.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) => UserAlarmReceived.cnf(+)	USED
43	OUTOFR	UserAlarmAck.req(AREP, API, Slot, Subslot) => UserAlarmAck.cnf(-)	OUTOFR
44	EMPTY	UserAlarmAck.req(AREP, API, Slot, Subslot) => UserAlarmAck.cnf(-)	EMPTY
45	USED	UserAlarmAck.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Lock) => UserAlarmAck.cnf(-)	USED
46	USED	UserAlarmAck.req(AREP, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) => UserAlarmAck.cnf(-)	USED
47	USED	UserAlarmAck.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) => UserAlarmAck.cnf(-)	USED
48	USED	UserAlarmAck.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) => UserAlarmAck.cnf(AREP, API, Slot, Subslot)	USED
49	OUTOFR	UserApplReady.req(AREP, ARType, API, Slot, Subslot) => UserApplReady.cnf(-)	OUTOFR
50	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) => UserApplReady.cnf(-)	EMPTY USED
51	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Lock) && (OK_Queue.FirstEntry = ANY) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = SO; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserApplReady.cnf(+)	EMPTY USED

#	Current State	Event /Condition =>Action	Next State
52	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) && (SubmodulFlags.ARP = OK) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = OWN; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserApplReady.cnf(+)	EMPTY USED
53	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) && (SubmodulFlags.ARP = ARP) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = ARP; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserApplReady.cnf(+)	EMPTY USED
54	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (ANY.ARType = IOC) && (AREP.ARType = IOS) => AddInfo = SupervisorTakeOverNotAllowed; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = IOC; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserApplReady.cnf(+)	EMPTY USED
55	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (ANY.ARType = IOC) && (AREP.ARType = IOC) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = IOC; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserApplReady.cnf(+)	EMPTY USED
56	EMPTY USED	UserApplReady.req(AREP, ARType, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (ANY.ARType = IOS) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = IOS; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserApplReady.cnf(+)	EMPTY USED
57	OUTOFR	UserConnectResponse.req(AREP, API, Slot, Subslot) => UserConnectResponse.cnf(-)	OUTOFR
58	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) => UserConnectResponse.cnf(-)	EMPTY USED

#	Current State	Event /Condition =>Action	Next State
59	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Lock) && (OK_Queue.FirstEntry = ANY) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = SO; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserConnectResponse.cnf(+)	EMPTY USED
60	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) && (SubmodulFlags.ARP = OK) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = OWN; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserConnectResponse.cnf(+)	EMPTY USED
61	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) && (SubmodulFlags.ARP = ARP) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = ARP; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserConnectResponse.cnf(+)	EMPTY USED
62	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (ANY.ARTYPE = IOC) && (AREP.ARTYPE = IOS) => AddInfo = SupervisorTakeOverNotAllowed; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = IOC; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserConnectResponse.cnf(+)	EMPTY USED
63	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (ANY.ARTYPE = IOC) && (AREP.ARTYPE = IOC) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = IOC; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserConnectResponse.cnf(+)	EMPTY USED
64	EMPTY USED	UserConnectResponse.req(AREP, API, Slot, Subslot) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (ANY.ARTYPE = IOS) => AddInfo = NIL; DiagInfo = SubmodulFlags.DiagInfo; ARInfo = IOS; UserCreateSubmoduleState.ind(AREP, API, Slot, Subslot, RealModulID, RealSubmodulID, ExpModulID, ExpSubmodulID, ARInfo, DiagInfo, AddInfo); UserConnectResponse.cnf(+)	EMPTY USED

#	Current State	Event /Condition =>Action	Next State
65	OUTOFR	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) => UserConnect.cnf(-)	OUTOFR
66	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated = Lock) && (OK_Queue.FirstEntry = EMPTY) => EnQueue.req(AREP.ARTYPE); OK_Queue.FirstEntry = (AREP); UserConnect.cnf(+)	EMPTY USED
67	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated = Lock) && (OK_Queue.FirstEntry = ANY) && NOT (ANY.ARTYPE IsLessThan ARTYPE) => EnQueue.req(AREP.ARTYPE); UserConnect.cnf(+)	EMPTY USED
68	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated = Lock) && (OK_Queue.FirstEntry = ANY) && (ANY.ARTYPE IsLessThan ARTYPE) && (ANY.ARProperties.SupervisorTakeoverAllowed = True) => EnQueue.req(AREP.ARTYPE); OK_Queue.FirstEntry = (AREP); UserConnect.cnf(+)	EMPTY USED
69	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated = Lock) && (OK_Queue.FirstEntry = ANY) && (ANY.ARTYPE IsLessThan ARTYPE) && (ANY.ARProperties.SupervisorTakeoverAllowed = False) => EnQueue.req(AREP.ARTYPE); UserConnect.cnf(+)	EMPTY USED
70	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.ARTYPE IsLessThan ARTYPE) && (ANY.ARProperties.SupervisorTakeoverAllowed = True) => EnQueue.req(AREP.ARTYPE); OK_Queue.FirstEntry = (AREP); UserControlledAlarm.ind(ANY.AREP, API, Slot, Subslot); UserConnect.cnf(+);	EMPTY USED
71	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated = Free && (OK_Queue.FirstEntry = ANY) && (ANY.ARTYPE IsLessThan ARTYPE) && (ANY.ARProperties.SupervisorTakeoverAllowed = False) => EnQueue.req(AREP.ARTYPE); UserConnect.cnf(+);	EMPTY USED
72	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated=Free) && (OK_Queue.FirstEntry = ANY) && NOT (ANY.ARTYPE IsLessThan ARTYPE) => EnQueue.req(AREP.ARTYPE); UserConnect.cnf(+);	EMPTY USED
73	EMPTY USED	UserConnect.req(AREP, API, Slot, Subslot, ExpModulID, ExpSubmodulID) /(SubmodulFlags.Superordinated=Free) && (OK_Queue.FirstEntry = EMPTY) => EnQueue.req(AREP.ARTYPE); OK_Queue.FirstEntry = (AREP) UserConnect.cnf(+);	EMPTY USED
74	OUTOFR	UserRelease.req(AREP, API, Slot, Subslot) => UserRelease.cnf(-)	OUTOFR
75	EMPTY USED	UserRelease.req(AREP, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) => UserRelease.cnf(-)	EMPTY USED

#	Current State	Event /Condition =>Action	Next State
76	EMPTY USED	UserRelease.req(AREP, API, Slot, Subslot) /(OK_Queue.FirstEntry <> (AREP)) => DeQueue.req(AREP.ARTYPE, AREP) UserRelease.cnf(+);	EMPTY USED
77	EMPTY USED	UserRelease.req(AREP, API, Slot, Subslot) /(OK_Queue.FirstEntry = (AREP)) && ((IOC_Queue.FirstEntry = EMPTY) (IOC_Queue.FirstEntry = (AREP))) && ((IOS_Queue.FirstEntry = EMPTY) (IOS_Queue.FirstEntry = (AREP))) && (IOC_Queue.SecondEntry = EMPTY) && (IOS_Queue.SecondEntry = EMPTY) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = EMPTY UserRelease.cnf(+)	EMPTY USED
78	USED	UserAlarmSend.req(AREP, API, Slot, Subslot) /(OK_Queue.FirstEntry = EMPTY) => UserAlarmSend.cnf(-)	USED
79	USED	UserAlarmSend.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) => UserAlarmSend.cnf(AREP = ANY.AREP, API, Slot, Subslot)	USED
80	EMPTY USED	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Lock && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOS_Queue.FirstEntry UserRelease.cnf(+)	EMPTY USED
81	EMPTY USED	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Lock && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = EMPTY) && (IOC_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOC_Queue.FirstEntry UserRelease.cnf(+)	EMPTY USED
82	EMPTY	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Free && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOS_Queue.FirstEntry UserPullAlarm.ind(ANY.AREP, API, Slot, Subslot); UserRelease.cnf(+);	EMPTY
83	USED	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Free && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (WRONG = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => SubmodulFlags.ARP = OK; DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOS_Queue.FirstEntry UserPlugWrongAlarm.ind(ANY.AREP, API, Slot, Subslot); UserRelease.cnf(+);	USED
84	USED	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Free && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && ((Substitue OK) = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => SubmodulFlags.ARP = OK; DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOS_Queue.FirstEntry UserReleasedAlarm.ind(ANY.AREP, ANY.API, ANY.Slot, ANY.Subslot); UserRelease.cnf(+);	USED

#	Current State	Event /Condition =>Action	Next State
85	EMPTY	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Free && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = EMPTY) && (IOC_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOC_Queue.FirstEntry UserPullAlarm.ind(ANY.AREP, API, Slot, Subslot); UserRelease.cnf(+);	EMPTY
86	USED	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Free && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = EMPTY) && (IOC_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && (WRONG = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOC_Queue.FirstEntry UserPlugWrongAlarm.ind(ANY.AREP, API, Slot, Subslot); UserRelease.cnf(+);	USED
87	USED	UserRelease.req(AREP, API, Slot, Subslot) /SubmoduleFlags.Superordinated = Free && (OK_Queue.FirstEntry = (AREP)) && (IOS_Queue.FirstEntry = EMPTY) && (IOC_Queue.FirstEntry = ANY) && (ANY.AREP <> AREP) && ((Substitute OK) = UserCheckIdent(API, Slot, Subslot, RealModulID, RealSubmodulID, ANY.ExpModulID, ANY.ExpSubmodulID)) => DeQueue.req(AREP.ARTYPE, AREP); OK_Queue.FirstEntry = IOC_Queue.FirstEntry UserReleasedAlarm.ind(ANY.AREP, ANY.API, ANY.Slot, ANY.Subslot); UserRelease.cnf(+);	USED
88	OUTOFR	UserAlarmAckReceived.req(AREP, API, Slot, Subslot) => UserAlarmAckReceived.cnf(-)	OUTOFR
89	EMPTY	UserAlarmAckReceived.req(AREP, API, Slot, Subslot) => UserAlarmAckReceived.cnf(-)	EMPTY
90	USED	UserAlarmAckReceived.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Lock) (OK_Queue.FirstEntry = EMPTY) (OK_Queue.FirstEntry.AREP <> (AREP)) => UserAlarmAckReceived.cnf(-)	USED
91	USED	UserAlarmAckReceived.req(AREP, API, Slot, Subslot) /(SubmoduleFlags.Superordinated = Free) && (OK_Queue.FirstEntry = ANY) && (ANY.AREP = AREP) => UserAlarmAckReceived.cnf(+)	USED

4.3.4 Behavior of an IO controller

4.3.4.1 Overview

Each IO Controller or IO Supervisor shall have a startup machine for establishing an AR.

4.3.4.2 Behavior of an IO controller during Startup

This subclause describes the client application behavior of an IO controller during startup of the user application as an example.

NOTE This subclause provides selected behavior only and is not a comprehensive definition of the client behavior.

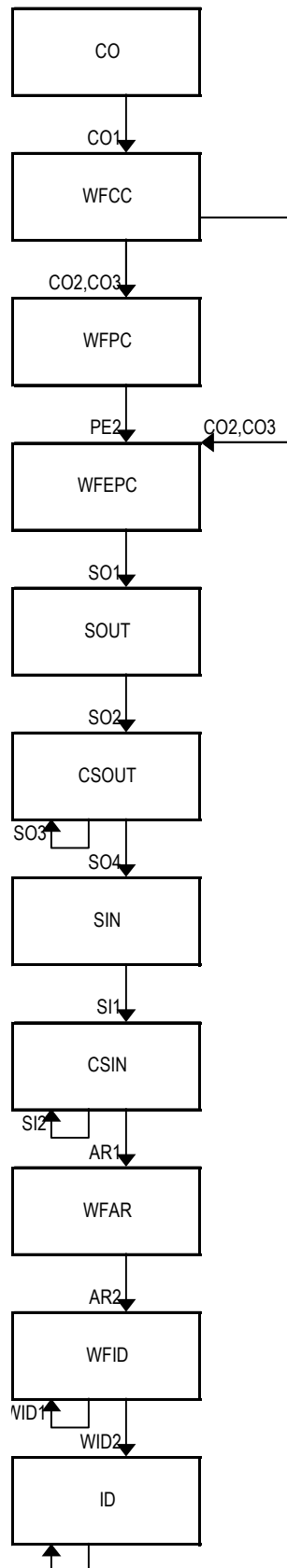


Figure 39 — State diagram client during startup

Figure 39 shows the state diagram and Table 108 defines the state table for the selected behavior.

Table 108 — State table client during startup

#	Current State	Event /Condition =>Action	Next State
1	CO	=> SetConnectRequestData("ExpectedConfiguration") AR Parameter Block.SessionKey:=StoredSessionKey+1 StoredSessionKey=StoredSessionKey+1 Connect.req(AREP, AR Parameter Block, List of IO CR Parameter Blocks, Alarm CR Parameter Block, List of IO Data Parameter Blocks)	WFCC
2	WFCC	Connect.cnf(+)(AREP, AR Response Block, List of IO CR Response Blocks, Alarm CR Response Block, List of IO Data Response Blocks) /CheckResponseBlocks()=Ok && ParameterToSend()=Ok => Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	WFPC
2	WFCC	Connect.cnf(+)(AREP, AR Response Block, List of IO CR Response Blocks, Alarm CR Response Block, List of IO Data Response Blocks) /CheckResponseBlocks()=Ok && ParameterToSend()!=Ok => EndOfParameter.req(AREP, Session Key)	WFPC
2	WFCC	Connect.cnf(+)(AREP, AR Response Block, List of IO CR Response Blocks, Alarm CR Response Block, List of IO Data Response Blocks) /CheckResponseBlocks()!=Ok && ParameterToSend()=Ok => AdaptConfiguration() Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	WFPC
2	WFCC	Connect.cnf(+)(AREP, AR Response Block, List of IO CR Response Blocks, Alarm CR Response Block, List of IO Data Response Blocks) /CheckResponseBlocks()!=Ok && ParameterToSend()!=Ok => AdaptConfiguration() EndOfParameter.req(AREP, Session Key)	WFPC
4	WFPC	Write.cnf(+)(AREP, SeqNumber) /ParameterToSend()!=Ok => Session Key=StoredSessionKey EndOfParameter.req(AREP, Session Key)	WFPC
5	WFPC	EndOfParameter.cnf(AREP, SessionKey) =>	SOUT
8	SOUT	=> For all involved slot/subslots do: Slot.Subslot.IOCS := GOOD Subslot Output Data:=Slot.Subslot.Initial Value For all involved slot/subslots do: Set Output.req (AREP, CREP, Slot Number,Subslot Number, IOPS, Subslot Output Data)	CSOUT
9	CSOUT	Set Output.cnf(+) /not last one =>	CSOUT
10	CSOUT	Set Output.cnf(+) /last one =>	SIN

#	Current State	Event /Condition =>Action	Next State
11	SIN	=> For all involved slot/subslots match expected configuration do: Slot := Slot Number; Subslot := Subslot Number IOCS := GOOD For all involved slot/subslots does not match expected configuration do: Slot := Slot Number; Subslot := Subslot Number IOCS := BAD_BY_CONTROLLER For all involved slot/subslots do: Set Input IOCS.req (AREP, CREP, Slot,Subslot, IOCS)	CSIN
12	CSIN	Set Input IOCS.cnf(+) /not last one =>	CSIN
13	CSIN	Set Input IOCS.cnf(+) /last one =>	WFAR
14	WFAR	ApplicationReady.ind(AREP, SessionKey) /Session Key=StoredSessionKey => ApplicationReady.rsp(AREP, SessionKey)	WFID
15	WFID	In_Data.ind(CREP.CPM) CM_New_Data.ind(CREP.CPM, APDU_Status) /not last one =>	WFID
16	WFID	In_Data.ind(CREP.CPM) CM_New_Data.ind(CREP.CPM, APDU_Status) /last one =>	ID
17	ID	=>	ID

4.3.5 Conformance classes

4.3.5.1 Version 1

Table 109 includes the device conformance classes with its supported basic functions.

Table 109 — Device Conformance

Function	IO Device	IO Controller	IO Supervisor
Context Management ASE – Initiator	O	M	M
Context Management ASE – Responder	M	O	n. a.
IO CR	M	M	M
Alarm CR	M	M	M
Record Data CR	M	M	M
Legend: M Mandatory O Optional n. a. not applicable			

An IO device shall support at least two ARs including two IO CRs, one Alarm CR and one Record Data CR with the following features:

- IO CR: up to 1440 Byte data length
- Alarm CR: up to 1432 Byte data length, minimal 200 Byte data length
- Record Data CR: up to 2^{32} -65 Byte data length, minimal 4176 Byte data length

Furthermore, an IO controller shall support at least one AR for each related IO device including two IO CRs, one Alarm CR and one Record Data CR with the following features:

- IO CR: up to 1440 Byte data length, min. 240 Byte consistency, support of all defined data types
- Alarm CR: up to 1432 Byte data length, minimal 200 Byte data length, capability to queue at least one high prior and one low prior alarm for each related IO device
- Record Data CR: up to 2^{32} -65 Byte data length, minimal 4176 Byte data length, capability to store for each related IO device 16 KByte startup parameter, no parallel access to a record data object

4.3.5.2 Version 2

The following definition shall be used additionally to version 1.

Table 109 includes the device conformance classes with its supported basic functions.

Table 110 — Device Conformance Version 2

Function	IO Device	IO Controller	IO Supervisor
Multicast Provider CR	M	M	M
Multicast Consumer CR	O	M	M
Legend:			
M Mandatory			
O Optional			
n.a. not applicable			

4.3.5.3 Timeout values for name resolution

The allowed values are shown in Table 111.

Table 111 — Timeout values for name resolution

Parameter	Value
ARP-ResponseTimeout	2 s
DCP-IdentifyTimeout	1 s
DCP-Set/GetTimeout	1 s
DNS-RequestTimeout	16 s

4.3.6 Application characteristics

4.3.6.1 Device ident number

The Device Ident Number is necessary for all IO devices. For each type of device an individual Device Ident Number will be needed. This Device Ident Number is not a series number. If a manufacturer has a Device Ident Number for one type of device it is allowed to use this number for all produced devices of the same type. At devices of the same type but with differences in the amount of inputs and outputs it is possible to use the same Device Ident Number for each produced IO Device.

Each device type shall have an individual Device Ident Number which is part of the object UUID.

The Device Ident Number consists of a Vendor ID, which is a globally organizational administered number, and a Device ID, which is administered by the manufacturer.

4.3.6.2 Local Real-time Scheduling Scheme

PROFINET IO provides a mean for RT communication. It is aimed to control the usage of the bandwidth within the system. However, RT Class 1 uses local unsynchronized clock

scheduling at the sending DTE interface. Therefore no special tolerance parameter for the jitter within the system are defined.

PROFINET IO devices shall use a data rate of 100 MBit/s and full duplex mode. In theory, each device could use always wired speed for sending frames. Such behavior applied over a certain amount of time would bring the system out operation. Therefore, PROFINET IO provides a fair mechanism to restrict the sending performance of each device.

As shown in Figure 40, the sending of frames is divided into different cycles at the local interface. Each cycle is defined by $T_{\text{Sendclock}}$ which is between 31,25 μs and 4 ms. At the beginning of cycle at $t_{\text{Sendclock}}$ the cyclic scheduled RT frames that are related to the IO Data ASE. The number of frames depends on the scheduled frames for the current phase and may be also zero. The time for RT frames should not exceed about 50 percent of the bandwidth for each cycle. Subsequently, possible RTA frames of the Alarm Data ASE will be sent. The time for RTA frames should not exceed about 10 percent of the bandwidth for each cycle. Then NRT (Non Realtime) frames (UDP/TCP) will be sent during the rest of available time minus a safety margin ($T_{60\%}$).

The attributes reduction ratio and phase should be used for the distribution of the RT Class 1 frames over the time. The network load is limited and steady by these means.

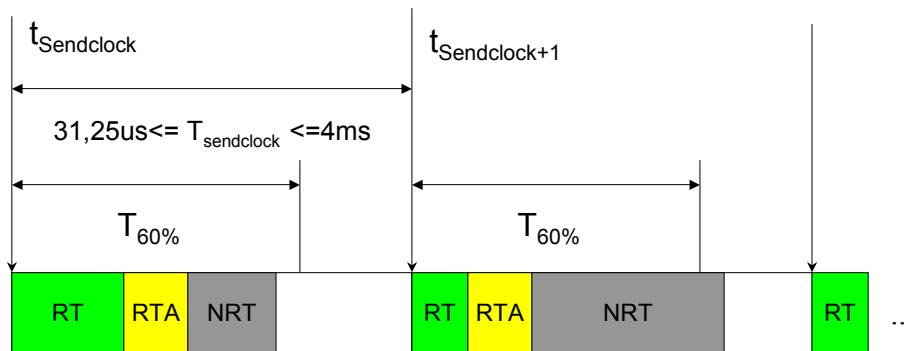


Figure 40 — Example of RT Class 1 behavior at the local interface

For the local scheduling the following rules are applied:

- The local sending process shall drop all RT frames for the whole cycle if at the beginning of the new sending period the first RT frame can not be send because of overload at the local interface.
- In average, not more than 60 percent of bandwidth should be used to avoid overload of the system or parts of the system.

If a NRT frame violates the phase it will be completed before the new RT frame is started (see Figure 41).

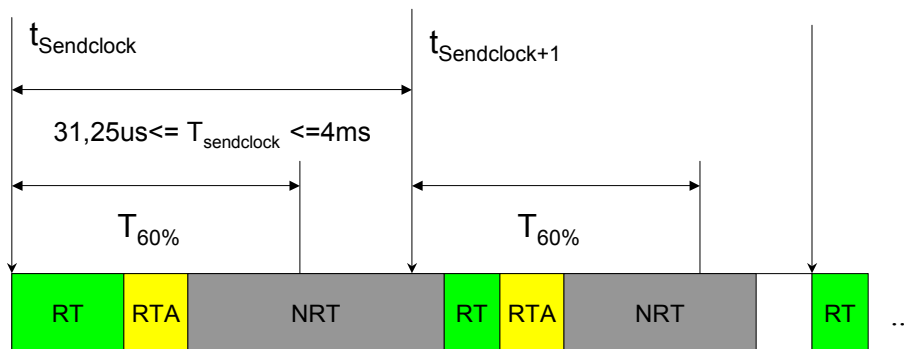


Figure 41 — Example of RT Class 1 behavior at the local interface

Figure 42 illustrates an example for an overload situation at the local sending interface. At $t_{\text{sendclock}+1}$ the provider gets a send error by the MAC because of a temporary overload situation with RT. In this case the provider completely skips the sending of RT frames for the

whole cycle and tries again within the next cycle. This event shall also be logged within the Logbook Data ASE. However, other frames (NRT) may be sent within the cycle if it is possible.

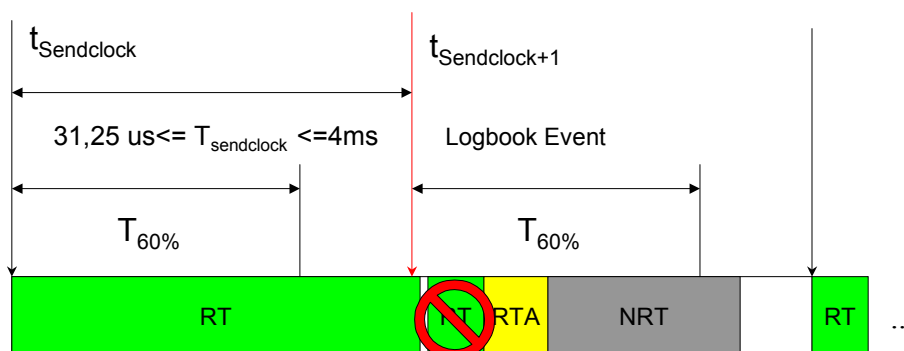


Figure 42 — Example of dropping RT Class 1 frames because of local overload

The priority transport of RT and RTA frames within the network is provided by the usage of VLAN priority tagging.

4.3.6.3 Soft Realtime Network Topology

PROFINET IO makes provisions concerning the network connection of IO controller and IO devices. The DTEs of such devices shall support a bitrate of 100 Mbit/s in full duplex mode. However, some topologies include e.g. wireless segments which do not fulfill these requirements. In such case, the wireless part shall be connected in such a way that only application relationships that use these network path suffer under lower performance. The adaptation can be done by means of the attribute reduction ratio. An example is depicted in Figure 43.

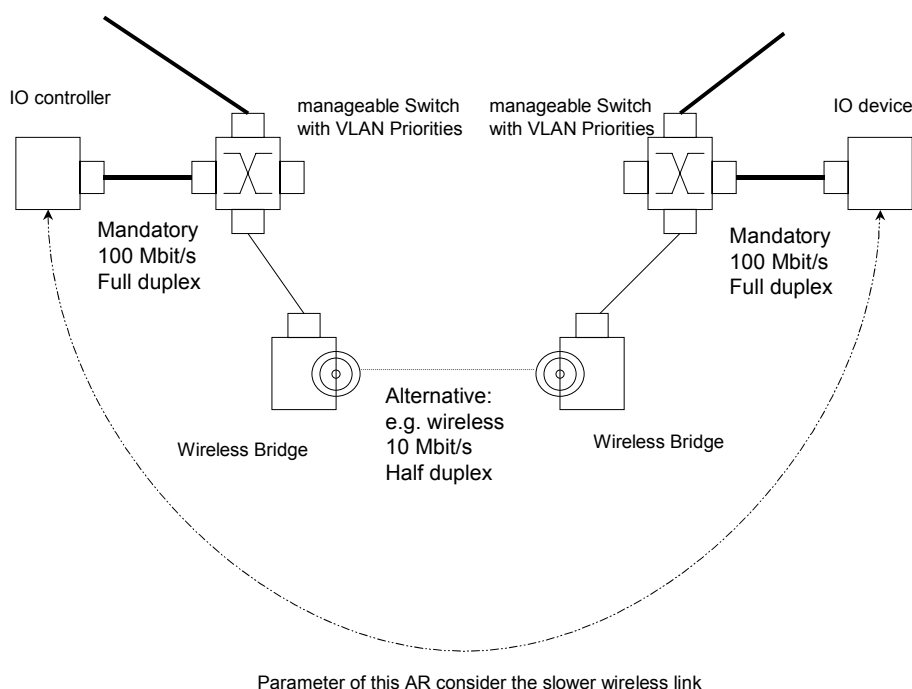


Figure 43 — Example of network topology including slower wireless segments

4.4 DCP service specification

4.4.1 Overview

The "Discovery and basic Configuration Protocol" represents a uniform method of distributing the device parameters (IP address, Name Of Station, ...). In particular, this means that the commissioning engineer requires only one method for all devices. Apart from the uniform configuration procedure, the protocol also allows devices on a subnet to be found and

supplied with an device parameters. As a result, it is therefore possible to supply all devices with device parameters from a central station on Ethernet. Due to its layer 2 based transport protocol the DCP protocol cannot be used between stations in different subnets connected with a router.

The following application scenario shows how DCP can be used to distribute the IP address in a system.

The commissioning engineer, a configuration station and the PROFINET devices are involved in the scenario. The commissioning engineer has a list of the devices in the system. These are uniquely identified (for example, by their MAC addresses or Name Of Station). The commissioning engineer connects the configuration station to the subnet on which the devices are located. The engineer starts the Identify service which is a multicast to all devices on the subnet. All the devices respond to this service. Further information may be read from each device by means of the Get service. The commissioning engineer gets this information displayed, for example, in the form of a list. From this list, the commissioning engineer selects the devices to be configured and sets the required parameters, such as the IP address. Which device receives which IP address is decided by the commissioning engineer based on an unique device identification.

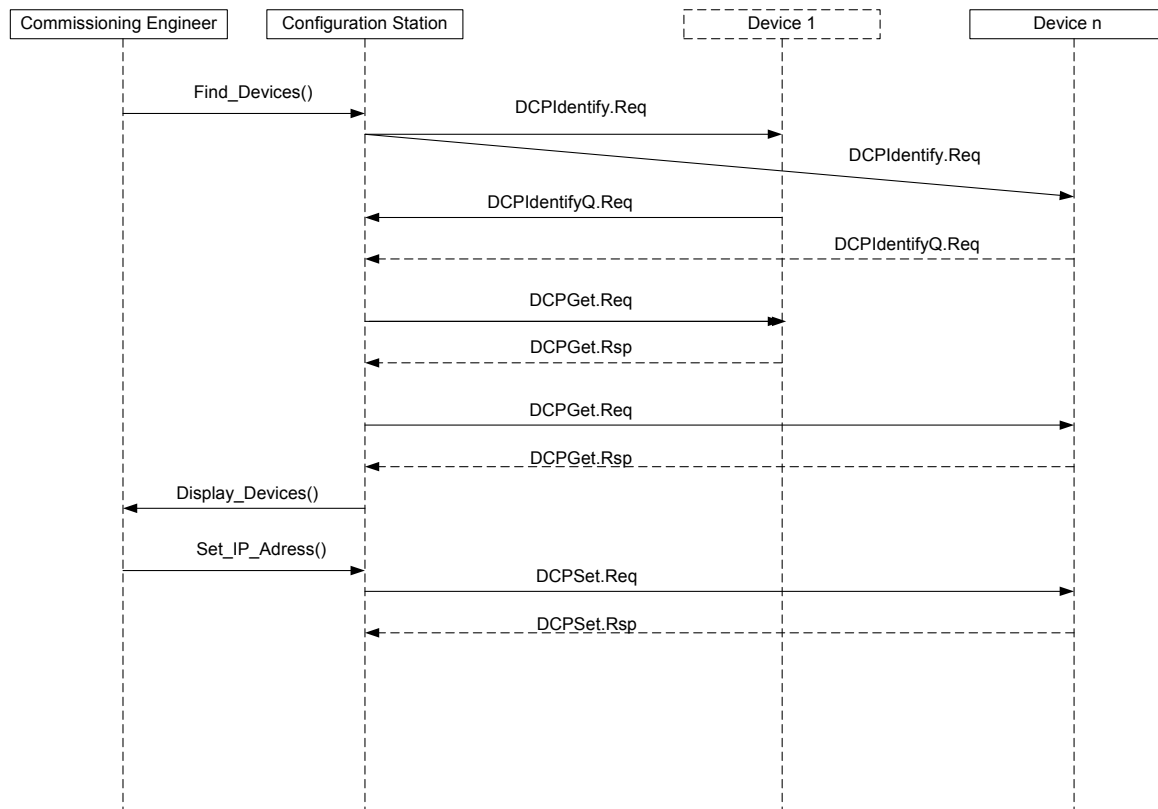


Figure 44 — Sequence diagram for DCP (example)

The DCP protocol defines the following services :

SERVICES:

- 1 (m) OpsService: DCP Get
- 2 (m) OpsService: DCP Set
- 3 (m) OpsService: DCP Identify
- 4 (m) OpsService: DCP Identify Q

4.4.2 DCP Get

The DCP Get service is used to read parameters from the device in terms of options/suboptions.

Table 112 shows the parameter of the service.

Table 112 — DCP Get

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
CREP	M	M(=)		
DA	M			
List Of Selectors	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Result(+)			S	S(=)
CREP			M	M(=)
List Of Data			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
Length			M	M(=)
Status			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
CREP			M	M(=)
ERRCLS			M	M(=)
ERRCODE			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

CREP

This parameter is the local identifier for the desired communication relation.

DA

This parameter is the destination Ethernet MAC Address of the desired communication partner.

List Of Selectors

This parameter contains the requested options/suboptions.

Option

This parameter contains the option that shall be read from the device.

Allowed values shall be according Table 113.

Table 113 — Option

Option
IP options
Device Properties options
DHCP options
LLDP options
Control
manufacturer specific options
AllSelector

Suboption

This parameter contains the suboption that shall be read from the device. Depending on the option only designated suboptions are allowed.

Allowed values for IP options shall be according Table 114.

Table 114 — Suboptions for IP option

Suboption
MAC address
IP-parameter

Allowed values for Control options shall be according Table 115.

Table 115 — Suboptions for control option

Suboption
StartTransaction
EndTransaction
Signal
Response
ResetToFactorySettings

Allowed values for Device Properties options shall be according Table 116.

Table 116 — Suboptions for DeviceProperties options

Suboption
Manufacturer specific
NameOfStation
Device-ID
Device-role
Device-options
Alias NameOfStation

Allowed values for DHCP option shall be according Table 117.

Table 117 — Suboption for DHCP

Suboption	References
Host Name ^{d)}	RFC 2132
Vendor specific information	RFC 2132
Server identifier	RFC 2132
Parameter request list ^{c)}	RFC 2132
Class-identifier	RFC 2132
DHCP client identifier	RFC 2132
FQDN, Fully Qualified Domain Name ^{d)}	
UUID/GUID-based Client (=addressed Station) Identifier	
Control DHCP for address resolution 0: Don't use DHCP 1: Don't use DHCP and reset all DHCP options 2: Use DHCP ^{b)} for address resolution	
a) The reserved options should be used in the same meaning as they are defined in the corresponding RFCs for DHCP. b) This option should be the last in a list of DHCP options in a set request. 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.	

Allowed values for LLDP option shall be according to IEEE P802.1AB.

Allowed values for AllSelector option shall be AllSelector suboption.

Allowed values for Signal option shall be FlashOnce.

Result(+)

This parameter indicates that the service request succeeded. It shall contain the requested values.

List Of Data

This parameter contains the values of the requested options/suboptions.

Option

This parameter contains the option that was read from the device.

Suboption

This parameter contains the suboption that was read from the device. Depending on the option only designated suboptions are allowed.

Length

This parameter contains the length of the data that were read from the device.

Status

This parameter contains the status of the data that were read from the device. Depending on the requested option/suboption the status has different meanings.

Data

This parameter contains the data that were read from the device.

Result(-)

This parameter indicates that the service request failed.

ERRCLS

The parameter ERRCLS contains the error class of the specific error.

Type: Unsigned16

ERRCODE

The parameter ERRCODE contains the error code of the specific error.

Type: Unsigned16

4.4.3 DCP Set

The DCP Set service is used to write parameters to the device in terms of options/suboptions.

Table 118 shows the parameter of the service.

Table 118 — DCP Set

Parameter name	Req	Ind	Rsp	Cnf
Argument	M	M(=)		
CREP	M	M(=)		
DA	M			
List Of Data	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Length	M	M(=)		
Status	M	M(=)		
Data	M	M(=)		
Result(+)			S	S(=)
CREP			M	M(=)
List Of Response			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
Length			M	M(=)
Status			M	M(=)
Result(-)			S	S(=)
CREP			M	M(=)
ERRCLS			M	M(=)
ERRCODE			M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

CREP

This parameter is the local identifier for the desired communication relation.

DA

This parameter is the destination Ethernet MAC Address of the desired communication partner.

List Of Data

This parameter contains the list of options/suboptions and the data to be written.

Option

This parameter contains the option that is written to the device.

Allowed values: see DCP Get

Suboption

This parameter contains the suboption that is written to the server. Depending on the option only designated suboptions are allowed.

Allowed values: see DCP Get

Length

This parameter contains the length of the data that is written to the server.

Status

This parameter contains the status of the data that is written to the server. Depending on the requested option/suboption the status has different meanings.

Data

This parameter contains the data that are written to the server.

Result(+)

This parameter indicates that the service request succeeded. It shall contain the requested values.

List Of Response

This parameter contains the return values of the write operation of the requested option/suboption.

Option

This parameter contains the option that is written to the server.

Suboption

This parameter contains the option that is written to the server.

Length

This parameter contains the length of the response status.

Status

This parameter contains the status of the set operation on the server side.

Result(-)

This parameter indicates that the service request failed.

ERRCLS

The parameter ERRCLS contains the error class of the specific error.

Type: Unsigned16

ERRCODE

The parameter ERRCODE contains the error code of the specific error.

Type: Unsigned16

4.4.4 DCP Identify

The DCP Identify service is used to identify devices. Only devices that fulfill the requested filter are allowed to reply to this request.

Table 119 shows the parameter of the service.

Table 119 — DCP Identify

Parameter name	Req	Ind	Cnf
Argument	M	M(=)	
CREP	M	M(=)	
DA	M		
List Of Filter	M	M(=)	
Option	M	M(=)	
Suboption	M	M(=)	
Length	M	M(=)	
Data	M	M(=)	
Response Delay	M	M(=)	

Parameter name	Req	Ind	Cnf
Result(+) CREP			S(=) M(=)
Result(-) CREP ERRCLS ERRCODE			S(=) M(=) M(=) M(=)

Argument

The argument shall convey the service specific parameters of the service request.

CREP

This parameter is the local identifier for the desired communication relation.

DA

This parameter is the destination Ethernet MAC Address of the desired communication partner.

List Of Filter

This parameter contains the options/suboptions that shall be used to filter the request on the server side.

Option

This parameter contains the value of the option that shall be used to filter the request on the server side.

Suboption

This parameter contains the value of the suboption that shall be used to filter the request on the server side.

Length

This parameter contains the length of the data of the option and suboption that shall be used to filter the request on the server side..

Data

This parameter contains the data of the option and suboption that shall be used to filter the request on the server side.

Response Delay

This parameter contains the value how long the response is delayed if the List Of Filter criteria are fulfilled.

Result(+)

This parameter indicates that the service request succeeded. It shall contain the requested values.

Result(-)

This parameter indicates that the service request failed.

ERRCLS

The parameter ERRCLS contains the error class of the specific error.

Type: Unsigned16

ERRCODE

The parameter ERRCODE contains the error code of the specific error.

Type: Unsigned16

4.4.5 DCP Identify Q

The DCP Identify Q service is used to identify devices. Devices that fulfill the filter delivered with an DCP Identify request are allowed to reply with the DCP Identify Q service.

Table 120 shows the parameter of the service.

Table 120 — DCP Identify Q

Parameter name	Req	Ind
Argument	M	M(=)
CREP	M	M(=)
DA	M	
List Of Data	M	M(=)
Option	M	M(=)
Suboption	M	M(=)
Length	M	M(=)
Status	M	M(=)
Data	M	M(=)

Argument

The argument shall convey the service specific parameters of the service request.

CREP

This parameter is the local identifier for the desired communication relation.

DA

This parameter is the destination Ethernet MAC Address of the desired communication partner.

List Of Data

This parameter contains the values of the options/suboptions.

Option

This parameter contains the option that is delivered from the device.

Suboption

This parameter contains the suboption that is delivered from the device. Depending on the option only designated suboptions are allowed.

Length

This parameter contains the length of the data that are delivered from the device.

Status

This parameter contains the status of the data that are delivered from the device. Depending on the requested option/suboption the status has different meanings.

Data

This parameter contains the data that are delivered from the device.

5 Part 6: Application Layer protocol specification of Type 10 for decentralized periphery

5.1 FAL syntax description

5.1.1 DLPDU abstract syntax reference

Table 121 gives an outline of the abstract syntax of the DLPDU. The encoding and decoding of the fields in Table 121 shall be according to ISO/IEC8802-3 for the DLPDU.

Table 121 — DLPDU syntax

DLPDU name	DLPDU structure
DLPDU	Preamble ^a , Start Frame Delimiter, Destination Address, Source Address, DLSDU ^b , DLPDU_Padding* ^c Frame Check Sequence
NOTE 1 According to ISO/IEC 8802-3 the DLPDU's have a minimum length of 64 octets (excluded Preamble, SFD).	
NOTE 2 For the ISO/IEC 8802-3 frames with VLAN tag is the minimum frame size increased to 68 octets in order to guaranty the minimum frame size of 64 octets after removing the VLAN tag by a bridge.	
^a The field contains at least 7 octets	
^b The minimum DLSDU size 2 octets.	
^c The number of padding octets shall be in the range of 0..46 depending on the DLSDU size. The value shall be set to zero.	

5.1.2 APDU abstract syntax

Table 122 defines the abstract syntax of the PROFINET IO Application Layer PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs. The APDUs of Table 122 shall represent the content of the DLSDU in Table 121.

Table 122 — APDU syntax

APDU name	APDU structure
PTP-RTA-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, PTPRTAHeader, PTPData a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept PTP-RTA-PDU with or without VLAN fields.
PTP-UDP-PDU	[VLAN] ^{a)} , LT(=0x0800), IPHeader, UDPHeader, PTPData a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept PTP-UDP-PDU with or without VLAN fields.
Data-RTC-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, C_SDU, APDU_Status a) The VLAN Field can be omitted in case of optimized RT transportation.
Data-RTA-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, Reference, PDUType (=1), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (1 – 1432), RTA-SDU, RTAPadding* ^{b)} a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept Data-RTA-PDU's with or without VLAN fields. b) The number of padding octets shall be in the range of 0..31 depending on the RTA-SDU size.
ACK-RTA-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, Reference, PDUType (=3), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (0),RTAPadding* ^{b)} a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept ACK-RTA-PDU's with or without VLAN fields. b) The number of padding octets shall be 32.

APDU name	APDU structure
NACK-RTA-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, Reference, PDUType (=2), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (=0), RTAPadding* ^{b)} a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept NACK-RTA-PDU's with or without VLAN fields. b) The number of padding octets shall be 32.
ERR-RTA-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, Reference, PDUType (=4), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (=4), PNIOStatus, RTAPadding* ^{b)} a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept ERR-RTA-PDU's with or without VLAN fields. b) The number of padding octets shall be 28.
CL-RPC-PDU	[VLAN], LT(=0x0800), IPHeader, UDPHeader, RPCHeader, NDRDataRequest ^ NDRDataResponse ^ NDREMapLookupReq ^ NDREMapLookupRes ^ NDREMapLookupFreeReq ^ NDREMapLookupFreeRes ^ NDRAck ^ NDRQuAck ^ NDRQuit ^ NDRFack ^ NDRFault ^ NDRNoCall ^ NDRPing ^ NDRReject
UDP-RTC-PDU	[VLAN] ^{a)} , LT(=0x0800), IPHeader, UDPHeader, DataCyclic a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept UDP-RTC-PDU with or without VLAN fields.
UDP-RTA-PDU	[VLAN] ^{a)} , LT(=0x0800), IPHeader, UDPHeader, DataAcyclic a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept UDP-RTA-PDU with or without VLAN fields.
ICMP-PDU	[VLAN] ^{a)} , LT(=0x0800), IPHeader ^{b)} , ICMPHeader, Data* a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept ICMP-PDU with or without VLAN fields. b) The value of IPHeader.IP_Protocol shall be to 1 to indicate ICMP.
LLDP-PDU	[VLAN] ^{a)} , LT(=0x88CC) ^{b)} , LLDPChassis, LLDPPort, LLDP TTL, LLDP8021*, LLDP8023*, [LLDPOption], LLDPEnd a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept LLDP-PDU with or without VLAN fields. b) The value of LT shall set according to IEEE P802.1AB
MRP-RTA-PDU	[VLAN] ^{a)} , LT(=0x8892), FrameID, MRPData, [Padding] ^{b)} a) The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept MRP-RTA-PDU with or without VLAN fields. b) If the frame is shorter than 64 octets it will be extended with padding to 64 octets.
NOTE 1 According to ISO/IEC 8802-3 the RTA PDU's have a minimum length of 52 octets. It is the sum of the minimum DLSDU size 2, the maximum Padding of 46, and the VLAN tag size of 4.	
NOTE 2 For the ISO/IEC 8802-3 frames with VLAN tag is the minimum frame size increased to 68 octets in order to guaranty the minimum frame size of 64 octets after removing the VLAN tag by a bridge.	

Table 123 defines structures for substitutions of elements of the APDU structures shown in Table 122.

Table 123 — Substitutions

Substitution name	Structure
VLAN	LT(=0x8100), TagControlInformation
C_SDU	{[DataItem], [GAP*]}*, [RTCPadding*] ^{a) b)} a) In case of Data-RTC-PDU the number of padding octets shall be in the range of 0..40 depending on the DataItem size. In case of UDP-RTC-PDU the number of padding octets shall be in the range of 0..12 depending on the DataItem size. b) In case of RT class 3 transportation the number of padding octets is given by the engineering system. The maximum C_SDU size shall be not larger than 1440 octets. Special Case Data-RTC-PDU for IOSAR with ARProperties.DeviceAccess=1: Padding*
DataItem	{[IOCS*], [DataObjectElement*]}*

Substitution name	Structure
DataObjectElement	[Data*], IOPS
APDU_Status	CycleCounter, DataStatus, TransferStatus
Reference	AlarmDstEndpoint, AlarmSrcEndpoint
RTA-SDU	AlarmNotification-PDU ^ AlarmAck-PDU
BlockHeader	BlockType, BlockLength, BlockVersionHigh, BlockVersionLow
AlarmNotification-PDU	BlockHeader, AlarmType, API, SlotNumber, SubslotNumber, ModuleIdentNumber, SubmoduleIdentNumber, AlarmSpecifier, [AlarmItem] The BlockType field within the BlockHeader shall be set to AlarmNotification high or low according Table 167.
AlarmAck-PDU	BlockHeader, AlarmType, API, SlotNumber, SubslotNumber, AlarmSpecifier, PNIOStatus The BlockType field within the BlockHeader shall be set to AlarmAck high or low according Table 167. Special Case "No Error": PNIOStatus(=0x00, 0x00, 0x00, 0x00) Special Case "Alarm Type Not Supported": PNIOStatus(=0xDA, 0x81, 0x3C, 0x00) Special Case "Wrong Submodule State": PNIOStatus(=0xDA, 0x81, 0x3C, 0x01)
AlarmItem	UserStructureIdentifier, Data* ^ ChannelDiagnosisData* ^ DiagnosisData* ExtChannelDiagnosisData* Special case ChannelDiagnosisData: UserStructureIdentifier(=0x8000), ChannelDiagnosisData* Special case Multiple for AlarmType Diagnosis: UserStructureIdentifier(=0x8001), DiagnosisData* Special case ExtChannelDiagnosisData: UserStructureIdentifier(=0x8002), ExtChannelDiagnosisData*
LLDPChassis	LLDP_TLVHeader ^{a)} , LLDP_ChassisIDSubType(7) ^{b)} , LLDP_ChassisID a – b) The encoding of the fields shall be according to IEEE P802.1AB.
LLDPPort	LLDP_TLVHeader ^{a)} , LLDP_PortIDSubType(7) ^{b)} , LLDP_PortID a – b) The encoding of the fields shall be according to IEEE P802.1AB.
LLDPTTL	LLDP_TLVHeader ^{a)} , LLDP_TimeToLive(10) ^{b)} a) The encoding of the fields shall be according to IEEE P802.1AB. b) The encoding of the fields shall be according to IEEE P802.1AB.
LLDPOption	The encoding of the fields shall be according to IEEE P802.1AB.
LLDPEnd	LLDP_TLVHeader(0) ^{a)} a) The encoding of the fields shall be according to IEEE P802.1AB.
LLDP8021	LLDP_TLVHeader ^{a)} , LLDP_OUI(00-80-C2) ^{b)} , LLDP_8021_SubType ^{c)} , Data ^{d)} a – d) The encoding of the fields shall be according to IEEE P802.1AB.
LLDP8023	LLDP_TLVHeader ^{a)} , LLDP_OUI(00-12-0F) ^{b)} , LLDP_8023_SubType ^{c)} , Data ^{d)} a – d) The encoding of the fields shall be according to IEEE P802.1AB.
IPHeader	IP_VersionIHL(0x45), IP_TypeOfService(0), 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 RFC791. a) For UDP-RTC-PDU and UDP-RTA-PDU fragmentation shall not be used. b) The field IP_Options shall be omitted for UDP-RTC-PDU and UDP-RTA-PDU.
ICMPHeader	ICMP_Type, ICMP_Code, ICMP_HeaderChecksum The encoding of the fields shall be according to RFC792.
UDPHeader	UDP_SrcPort, UDP_DstPort, UDP_DataLength, UDP_Checksum ^{a)} The encoding of the fields shall be according to RFC768. a) The UDP_Checksum should be set to zero for UDP-RTC-PDU and UDP-RTA-PDU to improve performance. It is not required for the receiver to check the UDP_Checksum for UDP-RTC-PDU and UDP-RTA-PDU.

Substitution name	Structure
RPCHeader	RPCVersion (4), RPCPacketType, RPCFlags, RPCFlags2, RPCDRep, RPCSerialHigh, RPCObjectUUID ^{a)} , RPCInterfaceUUID ^{b)} , RPCActivityUUID, RPCServerBootTime(1), RPCInterfaceVersion, RPCSequenceNmb, RPCOperationNmb ^{c)} , RPCInterfaceHint, RPCActivityHint, RPCLengthOfBody, RPCFragmentNmb, RPCAuthenticationProtocol, RPCSerialLow a) to identify IO device, IO controller, IO supervisor b) to identify PROFINET IO interface type c) to identify the service type e.g. Connect, Release, Read, Write, Control
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	RPCVersionFack(0), RPCPad1, RPCWindowSize, RPCMaxTsd, RPCMaxFragSize, RPCSerialNumber, RPCSelAckLen, RPCArrayOfSelAck*
NDRNoCall	NULL ^ NDRFack
NDRWorking	NULL
NDRPing	NULL
NDRReject	RPCNCARRejectStatus
DataCyclic	FrameID, C_SDU, APDU_Status
DataAcyclic	(FrameID, Reference, PDUType (=1), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (18 – 1432) ^{a)} , RTA-SDU) ^ (FrameID, Reference, PDUType (=2), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (=0), RTAPadding* ^{b)}) ^ (FrameID, Reference, PDUType (=3), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (0), RTAPadding* ^{c)}) ^ (FrameID, Reference, PDUType (=4), AddFlags, SendSeqNum, AckSeqNum, VarPartLen (=4), PNIOStatus) a) The maximum VarPartLen is calculated to fit in the RTA-Data-PDU and in the UDP-RTA-PDU. b – c) The number of padding octets shall be 4
NDREMapLookup Req	RPCInquiryType, RPCObjectReference(1), RPCObjectUUID, RPCInterfaceReference(2), RPCInterfaceUUID, RPCInterfaceVersionMajor, RPCInterfaceVersionMinor, RPCVersionOption(1), RPCEntryHandleAttribute(0), RPCEntryHandleUUID, RPCMaxEntries
NDREMapLookup Res	RPCEntryHandleAttribute(0), RPCEntryHandleUUID, RPCNumberOfEntries, RPCMaxEntries, RPCEntriesOffset, RPCEntriesCount, [(RPCObjectUUID, RPCTowerReference, RPCAnnotationOffset, RPCAnnotationLength, RPCAnnotation, [RPCGap*], RPCTowerLength, RPCTowerOctetStringLength, [RPCTowerOctetString*], [RPCGap*])*], RPCEMapStatus
RPCTowerOctetString	RPCFloorCount, [RPCFloor*]
RPCFloor	RPLHSByteCount, [RPCProtocolIdentifierString*], RPCRHSByteCount, [RPCRelatedData*]
RPCProtocolIdentifier String	(RPCID, RPCInterfaceUUID, RPCInterfaceVersionMajor) ^ (RPCID, RPCDataRepresentationUUID, RPCInterfaceVersionMajor) ^ RPCProtocolIdentifier ^ RPCServerUDPPort ^ RPCHostAddress
RPCRelatedData	RPCInterfaceVersionMinor ^ RPCPortNumber ^ RPCIPAddress
NDREMapLookup FreeReq	RPCEntryHandleAttribute, RPCEntryHandleUUID
NDREMapLookup FreeRes	RPCEntryHandleAttribute, RPCEntryHandleUUID, RPCEMapStatus
PROFINETIOService ReqPDU	IODConnectReq ^ IODWriteReq ^ IODWriteMultipleReq ^ IODReadReq ^ IODControlReq ^ IODReleaseReq ^ IOXControlReq
PROFINETIOService ResPDU	IODConnectRes ^ IODWriteRes ^ IODWriteMultipleRes ^ IODReadRes ^ IODControlRes ^ IODReleaseRes ^ IOXControlRes
IODConnectReq	ARBlockReq, {IOCRBlockReq*, AlarmCRBlockReq, [ExpectedSubmoduleBlockReq*], [PrmServerBlock], [MCRBlockReq*] ^{a)} } a) The field MCRBlockReq shall only be present if at least one IOCRBlockReq contains the value MULTICAST_CONSUMER_CR as an IO CR Type. Special case: "IOSAR with ARProperties.DeviceAccess=1" ARBlockReq, IOCRBlockReq, IOCRBlockReq, AlarmCRBlockReq
IODConnectRes	ARBlockRes, {IOCRBlockRes*, AlarmCRBlockRes, [ModuleDiffBlock]}

Substitution name	Structure
IODWriteMultipleReq	IODWriteHeader ^{a)} , (IODWriteHeader, 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 IODWriteHeader.
IODWriteReq	IODWriteHeader, RecordDataWrite
IODWriteHeader	BlockHeader, SeqNumber, ARUUID, 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
IODWriteMultipleRes	IODWriteResHeader ^{a)} , (IODWriteResHeader)* a) The value of the parameter index shall be 0xE040.
IODWriteResHeader	BlockHeader, SeqNumber, ARUUID, API, SlotNumber, SubslotNumber, Padding* ^{a)} , Index, RecordDataLength, AdditionalValue1, AdditionalValue2, PNIOStatus ^{b)} , RWPadding* ^{c)} a) The number of padding octets (value=0) in the write res is 2. b) In case of IODWriteRes PNIOStatus (=0,0,0,0). c) The number of padding octets (value=0) in the write res is 16.
IODReadReq	BlockHeader, SeqNumber, ARUUID, API, SlotNumber, SubslotNumber, Padding* ^{a)} , Index, RecordDataLength, [TargetARUUID] ^{b)} , RWPadding* ^{c)} a) The number of padding octets (value=0) in the read request is 2. b) The optional field TargetARUUID shall only be used in conjunction with the value 0 of ARUUID (Implicit AR). c) The number of padding octets (value=0) in the read request is 24 or 8.
IODReadRes	IODReadHeader, RecordDataRead
IODReadHeader	BlockHeader, SeqNumber, ARUUID, 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.
IODControlReq	ControlBlockConnect ^ ControlBlockPlug
IODControlRes	ControlBlockConnect ^ ControlBlockPlug ^ NULL In case of a negative response NULL shall be transmitted.
IODReleaseReq	ReleaseBlock
IODReleaseRes	ReleaseBlock ^ NULL In case of a negative response NULL shall be transmitted.
IOXControlReq	ControlBlockConnect ^ ControlBlockPlug, [ModuleDiffBlock] The field ModuleDiffBlock shall only be present if an Application Ready has to signal an error.
IOXControlRes	ControlBlockConnect ^ ControlBlockPlug ^ NULL In case of a negative response NULL shall be transmitted.
ARBlockReq	BlockHeader, ARType, ARUUID, SessionKey, CMInitiatorMacAdd, CMInitiatorObjectUUID, ARProperties, CMInitiatorActivityTimeoutFactor, UDPCMInitiatorPort, StationNameLength, CMInitiatorStationName The BlockType field within the BlockHeader shall be set to ARBlockReq according Table 167.
ARBlockRes	BlockHeader, ARType, ARUUID, SessionKey, CMResponderMacAdd, UDPCMResponderPort The BlockType field within the BlockHeader shall be set to ARBlockRes according Table 167.

Substitution name	Structure
IOCRBlockReq	BlockHeader, IOCRType, IOCRReference, LT, IOCRProperties, DataLength, FrameID^{a)}, SendClockFactor, ReductionRatio, Phase, Sequence, FrameSendOffset, WatchdogFactor, DataHoldFactor, IOCRTagHeader, IOCRMulticastMACAdd, NumberOfAPIs^{b)}, (API, NumberOfIODataObjects, (SlotNumber, SubslotNumber, IODataObjectFrameOffset)*, NumberOfIOCS, (SlotNumber, SubslotNumber, IOCSFrameOffset)*)* The BlockType field within the BlockHeader shall be set to IOCRBlockReq according Table 167. a) The field FrameID shall be Don't Care if the field IOCRType contains the value Output CR. b) Shall be set to zero if ARProperties.DeviceAccess is set to one. Special Case IOCRType = MULTICAST_CONSUMER_CR The field FrameID shall contain a FrameID from the defined multicast range. Special Case IOCRType = OUTPUT_CR and IOCRProperties = RTClass3 The field FrameID shall contain the FrameID used in the corresponding PDIRData.
MCRBlockReq	BlockHeader, IOCRReference, AddressResolutionProperties, MCTimeoutFactor, StationNameLength, ProviderStationName, [Padding]*
IOCRBlockRes	BlockHeader, IOCRType, IOCRReference, FrameID The BlockType field within the BlockHeader shall be set to IOCRBlockRes according Table 167.
ExpectedSubmoduleBlockReq	BlockHeader, NumberOfAPIs, (API, SlotNumber^{a)}, ModuleIdentNumber, ModuleProperties, NumberOfSubmodules, (SubslotNumber, SubmoduleIdentNumber, SubmoduleProperties^{b)}, (DataDescription, SubmoduleDataLength, LengthIOPS, LengthIOCS)*)* The BlockType field within the BlockHeader shall be set to ExpectedSubmoduleBlockReq according Table 167. a) The value of this parameter shall be identical for all APIs for one ExpectedSubmoduleBlockReq. b) The field SubmoduleProperties.Type determines the number of subsequent data description blocks.
ModuleDiffBlock	BlockHeader, NumberOfAPIs, (API, NumberOfModules, (SlotNumber, ModuleIdentNumber, ModuleState, NumberOfSubmodules, (SubslotNumber, SubmoduleIdentNumber, SubmoduleState)*))* The BlockType field within the BlockHeader shall be set to ModuleDiffBlock according Table 167.
AlarmCRBlockReq	BlockHeader, AlarmCRType, LT, AlarmCRProperties, RTATimeoutFactor, RTARetrieves, LocalAlarmReference, MaxAlarmDataLength, AlarmCRTagHeaderHigh, AlarmCRTagHeaderLow The BlockType field within the BlockHeader shall be set to AlarmCRBlockReq according Table 167.
AlarmCRBlockRes	BlockHeader, AlarmCRType, LocalAlarmReference, MaxAlarmDataLength The BlockType field within the BlockHeader shall be set to AlarmCRBlockRes according Table 167.
PrmServerBlock	BlockHeader, ParameterServerObjectUUID, ParameterServerProperties, CMInitiatorActivityTimeoutFactor, StationNameLength, ParameterServerStationName The BlockType field within the BlockHeader shall be set to PrmServerBlock according Table 167.
ControlBlockConnect	BlockHeader, reserved (2 octets), ARUUID, SessionKey, reserved (2 octets), ControlCommand, ControlBlockProperties The BlockType field within the BlockHeader shall be set to IODBlockReq, IODBlockRes, IOXBlockReq, IOXBlockRes according Table 167.
ControlBlockPlug	BlockHeader, reserved (2 octets), ARUUID, SessionKey, AlarmSequenceNumber, ControlCommand, ControlBlockProperties The BlockType field within the BlockHeader shall be set to IOXBlockReq, IOXBlockRes according Table 167. The field AlarmSequenceNumber shall contain the value of the sub-field AlarmSpecifier.SequenceNumber of the corresponding AlarmNotification-PDU.

Substitution name	Structure
ReleaseBlock	BlockHeader, reserved (2 octets), ARUUID, SessionKey, reserved (2 octets), ControlCommand, ControlBlockProperties The BlockType field within the BlockHeader shall be set to ReleaseBlock according Table 167.
RecordDataRead	DiagnosisData* ^ ExpectedIdentificationData ^ RealIdentificationData* ^ SubstituteValue ^ RecordInputDataObjectElement ^ RecordOutputDataObjectElement ^ Data* ^ ARData ^ IMData ^ LogData ^ ModuleDiffBlock ^ APIData ^ PDPortDataAdjust* ^ PDPortDataCheck* ^ PDPortDataReal* ^ PDIRData* ^ PDSyncData ^ PDevData ^ IsochronousModeData ^ NULL ^{a)} a) NULL shall be used if a requested well-known data record is empty (e.g. Diagnosis).
RecordDataWrite	SubstituteValue ^ Data* ^ IMData ^ PDPortDataAdjust* ^ PDPortDataCheck* ^ PDIRData* ^ PDSyncData ^ IsochronousModeData
DiagnosisData	BlockHeader, ChannelDiagnosis ^ ManufacturerSpecificDiagnosis ^ ExtChannelDiagnosis
ChannelDiagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^{a)} , UserStructureIdentifier(0x8000), ChannelDiagnosisData* a) The field ChannelProperties.Type shall be set to zero. The field ChannelProperties.Direction shall be set to zero. The field ChannelProperties.Specifier shall be set to one if at least one ChannelProperties.Specifier of the ChannelDiagnosisData is set to one.
ChannelDiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType
ManufacturerSpecificDiagnosis	SlotNumber, SubslotNumber, ChannelNumber, ChannelProperties, UserStructureIdentifier, Data* The BlockType field within the BlockHeader shall be set to ManufacturerSpecificDiagnosis according Table 167.
ExtChannelDiagnosis	SlotNumber, SubslotNumber, ChannelNumber(0x8000), ChannelProperties ^{a)} , UserStructureIdentifier(0x8002), ExtChannelDiagnosisData * a) The field ChannelProperties.Type shall be set to zero. The field ChannelProperties.Direction shall be set to zero. The field ChannelProperties.Specifier shall be set to one if at least one ChannelProperties.Specifier of the ChannelDiagnosisData is set to one.
ExtChannelDiagnosisData	ChannelNumber, ChannelProperties, ChannelErrorType, ExtChannelErrorType, ExtChannelAddValue
ExpectedIdentificationData	BlockHeader, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber))*
RealIdentificationData	BlockHeader, NumberOfSlots, (SlotNumber, ModuleIdentNumber, NumberOfSubslots, (SubslotNumber, SubmoduleIdentNumber))*
PDIRData	BlockHeader, NumberOfInterfaces, (SlotNumber, SubslotNumber, PDIRGlobalData, PDIRFrameData)*
PDSyncData	BlockHeader, NumberOfInterfaces, (SlotNumber, SubslotNumber, PTPSubdomainID, IRDataID, ReservedIntervalBegin, ReservedIntervalEnd, PLLWindow, SyncSendFactor, SendClockFactor, SyncProperties, SyncFrameAddress)*
PDevData	BlockHeader, Padding, Padding, [PDPortDataReal*], [Padding*] ^{a)} , [PDPortDataCheck*], [Padding*] ^{a)} , [PDPortDataAdjust*], [Padding*] ^{a)} , [PDIRData], [Padding*] ^{a)} , [PDSyncData] a) The number of padding octets shall be adapted to make the block DWORD aligned.
PDPortDataReal	BlockHeader, NumberOfElements, (SlotNumber, SubslotNumber, LengthOwnPortID, OwnPortID, [Padding] ^{a)} , NumberOfPeers, [LengthPeerPortID, PeerPortID, [Padding] ^{a)} , LengthPeerChassisID, PeerChassisID, [Padding] ^{a)} , PropagationDelayFactor]*, MAUType, [Padding] ^{a)} , DomainBoundary)* a) The number of padding octets shall be adapted to make the block WORD aligned.
TxPortGroup	NumberOfTxPortGroups, TxPortGroupArray
SubstituteValue	BlockHeader, SubstitutionMode, DataItem
RecordInputDataObjectElement	BlockHeader, LengthIOCS, IOCS, LengthIOPS, IOPS, LengthData, Data
RecordOutputDataObjectElement	BlockHeader, SubstituteFlag, LengthIOCS, LengthIOPS, LengthData, DataItem, SubstituteValue
RecordOutputDataSubstituteObjectElement	BlockHeader, LengthIOCS, IOCS, LengthIOPS, IOPS, LengthData, Data, LengthIOCS, IOCS, LengthIOPS, IOPS, LengthSubstituteData, SubstituteData

Substitution name	Structure
ARData	BlockHeader, NumberOfARs, (ARUUID, ARType, ARProperties, CMInitiatorObjectUUID, StationNameLength, CMInitiatorStationName, NumberOfIOCRs, (IOCRType, IOCRProperties, FrameID, APDUStatus ^{a)} , UDPCMInitiatorPort, UDPCMResponderPort)*, AlarmCRType, LocalAlarmReference, RemoteAlarmReference, ParameterServerObjectUUID ^{b)} , StationNameLength ^{c)} , ParameterServerStationName, NumberOfAPIs, API*)* a) The APDUStatus.CycleCounter and APDUStatus.TransferStatus can be zero b) The ParameterServerObjectUUID shall be NIL if not used c) The StationNameLength shall be 0 if not used
APIData	BlockHeader, NumberOfAPIs, API*
LogData	BlockHeader, ActualLocalTimeStamp, NumberOfLogEntries, (LocalTimeStamp, ARUUID, PNIOStatus, EntryDetail)*
CMInitiatorObjectUUID	PROFINETIOConstantValue, InstanceHigh, InstanceLow, DeviceIdentNumber
ParameterServerObject UUID	PROFINETIOConstantValue, InstanceHigh, InstanceLow, DeviceIdentNumber
DeviceIdentNumber	DeviceIDHigh, DeviceIDLow, VendorIDHigh, VendorIDLow
RPCAnnotation	DeviceType, Blank, OrderID, Blank, HWRevision, Blank, SWRevisionPrefix, SWRevision, EndTerm
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
IM_Version	IM_Version_Major, IM_Version_Minor
IM_Software_Revision	SWRevisionPrefix, IM_SWRevision_Functional_Enhancement, IM_SWRevision_Bug_Fix, IM_SWRevision_Internal_Change
PTPData	PTPHeader, PTPSyncDelayReq^PTPFollowUp^PTPDelayRes
PTPRTAHeader	PTP_RTALocalRecvTimeStamps, PTP_RTARemoteSendTimeStamps, PTP_RTADelay, PTP_RTASequenceNumber, PTP_RTAFlags, PTP_RTAReserved
PTPHeader	PTP_Version ^{a)} , PTP_VersionNetwork ^{b)} , PTP_Subdomain ^{c)} , PTP_MessageType ^{d)} , PTP_SourceCommunicationTechnologie ^{e)} , PTP_SourceUUID ^{f)} , PTP_SourcePortID ^{g)} , PTP_SourceSequenceID ^{h)} , PTP_Control ⁱ⁾ , PTP_Reserved ^{j)} , PTP_Flags ^{k)} , PTP_Reserved_1 ^{l)} a – l) The encoding of the fields shall be according to IEEE1588.
PTPSyncDelayReq	PTP_OriginTimeStamp ^{a)} , PTP_EPOCHNumber ^{b)} , PTP_CurrentUTCOffset ^{c)} , PTP_GrandmasterCommunicationTechnology ^{d)} , PTP_GrandmasterClockUUID ^{e)} , PTP_GrandmasterPortID ^{f)} , PTP_GrandmasterSequenceID ^{g)} , PTP_GrandmasterClockStratum ^{h)} , PTP_GrandmasterClockIdentifier ⁱ⁾ , PTP_GrandmasterClockVariance ^{j)} , PTP_GrandmasterPreferred ^{k)} , PTP_GrandmasterIsBoundaryClock ^{l)} , PTP_SyncInterval ^{m)} , PTP_LocalClockVariance ⁿ⁾ , PTP_LocalStepsRemoved ^{o)} , PTP_LocalClockStratum ^{p)} , PTP_LocalClockIdentifier ^{q)} , PTP_ParentCommunicationTechnology ^{r)} , PTP_ParentUUID ^{s)} , PTP_ParentPortField ^{t)} , PTP_EstimateMasterVariance ^{u)} , PTP_EstimateMasterDrift ^{v)} , PTP_UTCReasonable ^{w)} , [PTPOptions] a – w) The encoding of the fields shall be according to IEEE1588.
PTPFollowUp	PTP_AssociatedSequenceID ^{a)} , PTP_PreciseOriginTimeStamp ^{b)} a – b) The encoding of the fields shall be according to IEEE1588.
PTPDelayRes	PTP_DelayReceiveTimeStamps ^{a)} , PTP_RequestedSourceCommunicationTechnology ^{b)} , PTP_RequestedSourceUUID ^{c)} , PTP_RequestedPortID ^{d)} , PTP_RequestedSourceSequenceID ^{e)} a – e) The encoding of the fields shall be according to IEEE1588.
MRPHeader	MRP_TypeLength
MRPBlock	MRPHeader, MRP_Version, MRP_Command, MRP_MAC, MRP_Port, MRP_Info, MRP_Counter, MRP_Transition, MRP_TimeStamp, [MRPPadding*] ^{a)} a) The value is zero. MRPPadding ensures Unsigned16 alignment for the next block
MRPOption	MRPHeader, MRP_OUI, MRP_SubType, Data*, [MRPPadding*] ^{a)} a) The value is zero. MRPPadding ensures Unsigned16 alignment for the next block
MRPEnd	MRP_TypeLength(0)

Substitution name	Structure
MRPData	MRPBlock, [MRPOption*], MRPEnd
PTPBlockHeader	PTP_TypeLength, PTP_OUI, PTP_SubType
PTPOptions	PTPBlockHeader, PTP_Version, PTP_IRDataUUID
PDIRGlobalData	BlockHeader, IRDataID
PDIRFrameData	BlockHeader, NumberOfElements, (FrameSendOffset, DataLength, ReductionRatio, Phase, FrameID, Ethertype, RxPort, FrameDetails, TxPortGroup)*
IsochronousModeData	BlockHeader, TimeDataCycle, TimeOInput, TimeOOutput, TimeOInputValid, TimeOOutputValid, ControllerApplicationCycleFactor, Padding* ^{a)} a) The number of padding octets shall be adapted to make the block DWORD aligned.
PDPortDataAdjust	BlockHeader, NumberOfElements, (SlotNumber, SubslotNumber, [AdjustDomainBoundary], [AdjustMAUType])*
PDPortDataCheck	BlockHeader, NumberOfElements, (SlotNumber, SubslotNumber, [CheckPeers], [CheckPropagationDelayFactor], [CheckMAUType], [CheckDomainBoundary])*
AdjustDomainBoundary	BlockHeader, DomainBoundary, AdjustProperties, [Padding] ^{a)} a) The number of padding octets shall be adapted to make the block WORD aligned.
AdjustMAUType	BlockHeader, MAUType, AdjustProperties
CheckPeers	BlockHeader, NumberOfPeers, (LengthPeerPortID, PeerPortID, LengthPeerChassisID, PeerChassisID)*, [Padding] ^{a)} a) The number of padding octets shall be adapted to make the block WORD aligned.
CheckPropagationDelayFactor	BlockHeader, PropagationDelayFactor
CheckMAUType	BlockHeader, MAUType, [Padding] ^{a)} a) The number of padding octets shall be adapted to make the block WORD aligned.
CheckDomainBoundary	BlockHeader, DomainBoundary

5.1.3 Data types

5.1.3.1 Notation for the Boolean type

Boolean ::= BOOLEAN

```
-- TRUE if the value is non-zero.
```

-- FALSE if the value is zero.

5.1.3.2 Notation for the Integer type

```
Integer8 ::= INTEGER (-128..+127)
```

```
-- range  $-2^7 \leq i \leq 2^7-1$ 
```

```
Integer16 ::= INTEGER (-32768..+32767)
```

```
-- range  $-2^{15} \leq l \leq 2^{15}-1$ 
```

```
Integer32 ::= INTEGER
```

```
-- range  $-2^{31} \leq i \leq 2^{31}-1$ 
```

Integer64 ::= INTEGER

```
-- range  $-2^{63} \leq | \leq 2^{63}-1$ 
```

5.1.3.3 Notation for the Unsigned type

```
Unsigned8 ::= INTEGER (0..255)
```

```
-- range 0 <= l <= 28-1
```

Unsigned16 ::= INTEGER (0..65535)

```
-- range 0 <= | <= 216-1
```

Unsigned32 ::= INTEGER

```
-- range 0 <= i <= 232-1
```

Unsigned64 ::= INTEGER

```
-- range 0 <= l <= 264-1
```

5.1.3.4 Notation for the Floating Point type

Floating32 ::= BIT STRING SIZE (4)

-- IEC 60559 Single precision

Floating64 ::= BIT STRING SIZE (8)

-- IEC 60559 Double precision

5.1.3.5 Notation for the OctetString type

OctetString ::= OCTET STRING

-- For generic use

5.1.3.6 Notation for VisibleString type

VisibleString2 ::= VISIBLE STRING -- For generic use

5.1.3.7 Notation for BinaryDate type

BinaryDate ::= OctetString7

5.1.3.8 Notation for TimeOfDay type

TimeOfDay with date indication ::= OctetString6

TimeOfDay without date indication ::= OctetString4

5.1.3.9 Notation for TimeDifference type

TimeDifference with date indication ::= OctetString6

TimeDifference without date indication ::= OctetString4

5.1.3.10 Notation for Network Time type

Network Time ::= OctetString8

5.1.3.11 Notation for Network Time Difference type

Network Time Difference ::= OctetString8

5.2 Transfer syntax

5.2.1 Coding of basic data types

5.2.1.1 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.

5.2.1.2 Encoding of an Integer value

- The encoding of a fixed-length Integer value of Integer8, Integer16, and Integer32 types shall be primitive, and the ContentsOctets shall consist of exactly one, two, or four 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.

5.2.1.3 Encoding of an Unsigned value

- The encoding of a fixed-length Unsigned value of Unsigned8, Unsigned16, and Unsigned32 types shall be primitive, and the ContentsOctets shall consist of exactly one, two, or four 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.

5.2.1.4 Encoding of a Floating-Point value

- The encoding of a Floating32 value shall be primitive, and the ContentsOctets shall consist of exactly four octets.
- The ContentsOctets shall contain floating-point values defined in conformance with IEC 60559. 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 Floating32.

5.2.1.5 Encoding of a Visible String 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.

5.2.1.6 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.

5.2.1.7 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 45:

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	RSV	RSV	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0...59 min
4	SU	RSV	RSV	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0...23 hours
5	day of week 2 ² 2 ¹ 2 ⁰		day of month 2 ⁴ 2 ³ 2 ² 2 ¹ 2 ⁰						1...7 day of week 1...31 day of month
6	RSV	RSV	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	1...12 months
7	RSV	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				

NOTE To avoid the Y2K problem the values of the year are from 0 to 50 for the years 2000 to 2050 and from 51 to 99 for the years 1951 to 1999. This is according to the British Standards Institution (BSi) DISC PD2000-1:1998 "A Definition of Year 2000 Conformance Requirements" Rule 3 (b).

Figure 45 — Coding of the data type BinaryDate

5.2.1.8 Encoding of a Time Of Day with and without date indication value

- The encoding of a Time Of Day with and without date indication value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the Time Of Day with and without date indication value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 46:

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 ms 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 01.01.84 only with date indication
6	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
msb					lsb				

Figure 46 — Encoding of Time Of Day value

5.2.1.9 Encoding of a Time Difference with and without date indication value

- The encoding of a Time Difference with and without date indication value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the Time Difference with and without date indication value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 47:

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	ms
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 only with date indication
6	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	
msb					lsb				

Figure 47 — Encoding of Time Difference value

5.2.1.10 Encoding of a Network Time value

- The encoding of a Network Time value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the Time Difference value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 48:

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	Seconds since 1.1.1900 0.00,00 or since 7.2. 2036 6.28,16 when time value less than 0x9dff4400.00000000
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 ³¹	2 ³⁰	2 ²⁹	2 ²⁸	2 ²⁷	2 ²⁶	2 ²⁵	2 ²⁴	Fraction Part of seconds one unit is 1/(2 ³²) s
6	2 ²³	2 ²²	2 ²¹	2 ²⁰	2 ¹⁹	2 ¹⁸	2 ¹⁷	2 ¹⁶	
7	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	
8	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	
	msb				lsb				

Figure 48 — Encoding of Network Time value

5.2.1.11 Encoding of a Network Time Difference value

- The encoding of a Network Time Difference value shall be primitive.
- The Length field shall indicate as a binary number the number of octets in the Time Difference value.
- The ContentsOctets shall be equal in value to the octets in the data value, as shown in Figure 49:

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}	Seconds as Integer32
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^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}	2^{24}	Fraction Part of seconds one unit is $1/(2^{32})$ s
6	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	
7	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	
8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
	msb				lsb				

Figure 49 — Encoding of Network Time Difference value

5.2.1.12 Encoding of a NULL value

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

5.2.1.13 Encoding of a NIL value

- The encoding of a NIL value shall be primitive.
- Each ContentsOctet shall be set to 0x00.

5.2.1.14 Encoding of an UUID

- The encoding of this data type shall be according to DCE RPC specification and IEC 61588 for PTP_ParentUUID.

5.2.2 Coding section related to common basic fields

5.2.2.1 Overview

The common fields of this subclause are part of the Data-RTC-PDU, Data-RTA-PDU, ACK-RTA-PDU, NACK-RTA-PDU and ERR-ACK-PDU.

5.2.2.2 Coding of the field LT

This field shall be coded with the values according to ISO/IEC 8802-3 (Unsigned16). This specification uses the values according to Table 124.

Table 124 — LT

Value (hexadecimal)	Meaning
0x0800	IP, UDP, RPC
0x0806	ARP
0x8100	Tag Control Information
0x8892	Data-RTC-PDU, Data-RTA-PDU (RT Class 1 – 3)
0x88CC	LLDP

5.2.2.3 Coding of the field TagControlInformation

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 11: TagControlInformation.VLAN_Id

This field shall be coded according to IEEE802.1Q.

NOTE It is recommended to use the VLAN_Id 0 that means no use of VLANs.

Bit 12: TagControlInformation.CanonicalFormatIdentifier

This field shall be coded according to IEEE802.1Q.

NOTE CFI is constant 0.

Bit 13 – 15: TagControlInformation.Priority

This field shall be coded according to IEEE802.1Q.

This field shall be coded with the values according to Table 125.

Table 125 — TagControlInformation.Priority

Value (hexadecimal)	Meaning
0x00	IP (RPC)
0x01	reserved
0x02	reserved
0x03	reserved
0x04	reserved
0x05	Data-RTA-PDU low, UDP-RTA-PDU low
0x06	Data-RTC-PDU and Data-RTA-PDU high, UDP-RTC-PDU, UDP-RTA-PDU high
0x07	reserved

NOTE If the used IP stack does not provide Tag Control Information then the UDP-RTA-PDU and the UDP-RTC-PDU is send without Tag Control Information.

5.2.2.4 Coding of the field FrameID

This field shall be coded as data type Unsigned16 with the values according to Table 104.

This field identifies the structure and the type of the APDU.

Table 126 — FrameID

Value (hexadecimal)	Meaning	Use
0x0000 – 0x001F	PTP sync	Precision time protocol synchronization
0x0020 – 0x003F	PTP delay request	Precision time protocol synchronization
0x0040 – 0x005F	PTP additional delay request	Precision time protocol synchronization
0x0060 – 0x007F	PTP additional delay response	Precision time protocol synchronization
0x0080 – 0x009F	PTP sync for RT class 3	RT Class 3 Precision time protocol synchronisation
0x00A0 – 0x00FF	reserved	reserved
0x0100 – 0x7FFF	dedicated to RT Class 3 unicast and multicast	IRT
0x8000 – 0xBEFF	dedicated to RT Class 2 unicast	RT
0xBF00 – 0xBFFF	dedicated to RT Class 2 multicast	RT
0xC000 – 0xFAFF	dedicated to RT Class 1 unicast	RT and RTOverUDP
0xFB00 – 0xFBFF	dedicated to RT Class 1 multicast	RT and RTOverUDP
0xFC00	reserved	
0xFC01	Alarm High	RT and RTOverUDP
0xFC02 – 0xFDFF	reserved	
0xFE00	reserved	
0xFE01	Alarm Low	RT and RTOverUDP
0xFE02 – 0xFEFC	reserved	
0xFEFD – 0xFEFF	DCP	RT (see NOTE)
0xFF00 – 0xFF1F	PTP follow up	Precision time protocol synchronization
0xFF20 – 0xFF3F	PTP delay response	Precision time protocol synchronization

Value (hexadecimal)	Meaning	Use
0xFF40- 0xFF5F	PTP additional delay followup request	Precision time protocol synchronization
0xFF60	MRP	Medium redundancy protocol
0xFF61 – 0xFFFF	reserved	reserved
NOTE RTOverUDP may also used		

Table 127 contains the FrameID for PTP sync frame.

Table 127 — FrameID for PTP sync

Value (hexadecimal)	Meaning	Use
0x0000	Clock	Send Clock and phase synchronization
0x0001	Time	time synchronization
0x0002 – 0x001F	reserved	

Table 128 contains the FrameID for PTP delay request frame.

Table 128 — FrameID for PTP delay request

Value (hexadecimal)	Meaning	Use
0x0020	Clock	Send Clock and phase synchronization
0x0021	Time	time synchronization
0x0022 – 0x003F	reserved	

Table 129 contains the FrameID for PTP additional delay request frame.

Table 129 — FrameID for PTP additional delay request

Value (hexadecimal)	Meaning	Use
0x0040	Clock	Send Clock and phase synchronization
0x0041	Time	time synchronization
0x0042 – 0x005F	reserved	

Table 130 contains the FrameID for PTP additional delay response frame.

Table 130 — FrameID for PTP additional delay response

Value (hexadecimal)	Meaning	Use
0x0060	Clock	Send Clock and phase synchronization
0x0061	Time	time synchronization
0x0062 – 0x007F	reserved	

Table 131 contains the FrameID for PTP sync for RT class 3 frame.

Table 131 — FrameID for PTP sync for RT class 3

Value (hexadecimal)	Meaning	Use
0x0080	Clock	Send Clock and phase synchronization
0x0081	Time	time synchronization
0x0082 – 0x009F	reserved	

Table 132 contains the FrameID for PTP follow up frame.

Table 132 — FrameID for PTP follow up

Value (hexadecimal)	Meaning	Use
0xFF00	Clock	Send Clock and phase synchronization
0xFF01	Time	time synchronization
0xFF02 – 0xFF1F	reserved	

Table 133 contains the FrameID for PTP delay response frame.

Table 133 — FrameID for PTP delay response

Value (hexadecimal)	Meaning	Use
0xFF20	Clock	Send Clock and phase synchronization
0xFF21	Time	time synchronization
0xFF22 – 0xFF3F	reserved	

Table 134 contains the FrameID for PTP additional delay followup request frame.

Table 134 — FrameID for PTP additional delay followup request

Value (hexadecimal)	Meaning	Use
0xFF40	Clock	Send Clock and phase synchronization
0xFF41	Time	time synchronization
0xFF42 – 0xFF5F	reserved	

5.2.3 Coding section of PTP-RTA-PDU specific fields

5.2.3.1 Coding of the field PTP_RTALocalReceiveTimeStamp

This field shall be coded as data type Unsigned32. The value contains the clock counter at receive time (resolution of 10ns). This field is relevant for Sync- and DelayReq-Messages. For all other Time-Messages this field shall be set to zero.

5.2.3.2 Coding of the field PTP_RTARemoteSendTimeStamp

This field shall be coded as data type Unsigned32. The value contains the clock counter at send time (resolution of 10ns). This field is relevant for Sync- and DelayReq-Messages. For all other Time-Messages this field shall be set to zero.

5.2.3.3 Coding of the field PTP_RTADelay

This field shall be coded as data type Unsigned32. The value contains the propagation delay time (resolution of 10ns). This field is relevant for Sync- and FollowUp-Messages. For all other Time-Messages this field shall be set to zero.

5.2.3.4 Coding of the field PTP_RTASequenceNumber

This field shall be coded as data type Unsigned16. This field shall be incremented by the TimeMaster with every new Sync-Frame.

5.2.3.5 Coding of the field PTP_RTAFlags

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 3: PTP_RTAFlags.LocalReceiveExtensions

This field contains the clock counter at receive time (resolution of 1ns). This field is relevant for Sync- and DelayReq-Messages. For all other Time-Messages this field shall be set to zero.

Table 135 — PTP_RTAFIags.LocalReceiveExtensions

Value (hexadecimal)	Meaning
0x00	default
0x00 – 0x09	valid
0x0A – 0x0F	reserved

Bit 4 – 7: PTP_RTAFIags.RemoteSendExtensions

This field contains the clock counter at send time (resolution of 1ns). This field is relevant for Sync- and DelayReq-Messages. For all other Time-Messages this field shall be set to zero.

Table 136 — PTP_RTAFIags.RemoteSendExtensions

Value (hexadecimal)	Meaning
0x00	default
0x00 – 0x09	valid
0x0A – 0x0F	reserved

Bit 8 – 11: PTP_RTAFIags.DelayExtensions

This field contains the propagation delay time (resolution of 1ns). This field is relevant for Sync- and FollowUp-Messages. For all other Time-Messages this field shall be set to zero.

Table 137 — PTP_RTAFIags.DelayExtensions

Value (hexadecimal)	Meaning
0x00	default
0x00 – 0x09	valid
0x0A – 0x0F	reserved

Bit 12: PTP_RTAFIags.FollowUp

This field shall be coded with the values according to Table 138 for Sync-Messages. For all other Time-Messages this field shall be set to zero.

Table 138 — PTP_RTAFIags.FollowUp

Value (hexadecimal)	Meaning
0x00	No FollowUp frame
0x01	FollowUp frame

Bit 13: PTP_RTAFIags.DelayMeasure

This field shall be coded with the values according to Table 139 for Sync-Messages. For all other Time-Messages this field shall be set to zero.

Table 139 — PTP_RTAFIags.DelayMeasure

Value (hexadecimal)	Meaning
0x00	No delay measurement
0x01	Sign for start of delay measurement

Bit 14 – 15: reserved

This field shall be set to zero.

5.2.3.6 Coding of the field PTP_IRDataUUID

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)

5.2.3.7 Coding of the field PTP_TypeLength

The coding of this field shall be according to 3.14.2.4. and the individual bits shall have the following meaning:

Bit 0 – 6: PTP_TypeLength.Type

This field shall be coded with the values according to Table 140.

Table 140 — PTP TypeLength.Type

Value (hexadecimal)	Meaning	Use
0x00	EndOfPDU	
0x01 – 0x7E	reserved	
0x7F	Organizationally Specific	optional

Bit 7 – 15: PTP_TypeLength.Length

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

5.2.3.8 Coding of the field PTP_OUI

This field shall be coded as Octet[3] with the Organizationally unique identifier (OUI) as defined ISO/IEC 8802.

5.2.3.9 Coding of the field PTP_SubType

This field shall be coded as Unsigned8. The value is vendor/organizationally specific. For the OUI the parameter PTP_SubType shall be coded as in Table 141.

Table 141 — PTP_SubType

Value (hexadecimal)	Meaning	Usage
0x00	reserved	
0x01	IRDATA	Used for RT Class 3
0x02 – 0xFF	reserved	

5.2.4 Coding section of Data-RTC-PDU specific fields

5.2.4.1 Overview

The field Data and IOxS shall also be used to encode user data for all other PDU types and it is not restricted to the PDU of this subclause.

5.2.4.2 Coding of the field Data

One of the following data types shall be used for each Data field:

- Boolean
- Integer
- Unsigned
- Floating Point
- Visible String
- OctetString
- Binary Date
- Time of Day
- Time-Difference
- Network Time
- Network Time Difference

5.2.4.3 Coding of the field SubstituteData

One of the following data types shall be used for each Data field of substitute data:

- Boolean
- Integer
- Unsigned
- Floating Point
- Visible String
- OctetString
- Binary Date
- Time of Day
- Time-Difference
- Network Time
- Network Time Difference

5.2.4.4 Coding of the field IOxS (IOCS, IOPS)

The coding of these fields shall be according to 3.14.2.3 and the individual bits shall have the following meaning:

Bit 0: IOxS.Extension

This field shall be coded with the values according to Table 142.

Table 142 — IOxS.Extension

Value (hexadecimal)	Meaning
0x00	no IOxS octet follows
0x01	one more IOxS octet follows

Bit 1 to 4: reserved

This field shall be set to zero but not checked at the receiver.

Bit 6 to 5: IOxS.Instance

This field shall be coded with the values according to Table 143 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 143 — IOCS.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 144.

Table 144 — 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
0x01	good, Data field shall contain valid user data

5.2.4.5 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-Data-PDU and 1 ms for UDP-RTC-PDU. Each RTC-Data-PDU contains its local CycleCounter. The receiver of the RTC-Data-PDU shall use this field to detect repetitions, loss of frames, and timeliness of data.

Table 145 defines the action of the consumer according to the difference between the CycleCounter of the last received RTC-Data-PDU or UDP-RTC-PDU and the value of the current received PDU.

Table 145 — CycleCounter Difference

Value (hexadecimal)	Meaning	Use
0x0000-0x0FFF	the consumer votes the received frame as older than the stored frame	frame is dropped
0x1000-0xFFFF	the consumer votes the received frame as newer than the stored frame	frame is processed

The CycleCounter defines a time window between the old and the new frame of 1,92 s to vote as new for RTC-Data-PDUs and 61,44 s to vote as new for UDP-RTC-PDUs.

NOTE If necessary, the local CycleCounter may tightly synchronized global in the network.

A device shall maintain a local 64 bit counter each bit representing 31,25 us. The bits 0 to 15 shall be used for CycleCounter in RTC-Data-PDUs. The bits 5 to 20 shall be used for CycleCounter in UDP-RTC-PDUs.

5.2.4.6 Coding of the field DataStatus

The coding of this field shall be according to 3.14.2.3 and the individual bits shall have the following meaning:

Bit 0: DataStatus.State

This field shall be coded with the values according to Table 146.

Table 146 — DataStatus.State

Value (hexadecimal)	Meaning
0x00	IOCR state is backup
0x01	IOCR state is primary

Bit 1: reserved

This field shall be set to zero but not checked at the receiver.

Bit 2: DataStatus.DataValid

This field shall be coded with the values according to Table 147.

Table 147 — DataStatus.DataValid

Value (hexadecimal)	Meaning
0x00	DataItem invalid
0x01	DataItem valid

Bit 3: reserved, shall be set to zero but not checked at the receiver

Bit 4: DataStatus.ProviderState

This field shall be coded with the values according to Table 148.

Table 148 — DataStatus.ProviderState

Value (hexadecimal)	Meaning
0x00	Stop
0x01	Run

Bit 5: DataStatus.StationProblemIndicator

This field shall be coded with the values according to Table 149.

Table 149 — DataStatus.StationProblemIndicator

Value (hexadecimal)	Meaning
0x00	Problem detected
0x01	Normal Operation

Bit 6: reserved

This Field shall be set to zero but not checked at the receiver.

Bit 7: reserved

This Field shall be set to zero but not checked at the receiver.

5.2.4.7 Coding of the field TransferStatus

This field shall be coded as data type Unsigned8. This field shall be set to zero for RT class 1 and RT Class 2 frames. For RT class 3 the value shall be set according to Table 150.

Table 150 — The bits in the TransferStatus in a RT frame (RT class 3)

Bit No in Transfer state	Name of the error event	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	IRTErrors	0: no IRT error 1: IRT error, i.e. 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, i.e. the bridge received a header, but the data have not arrived on time
other	none	reserved

5.2.5 Coding section related to RTA-PDUs specific fields**5.2.5.1 Coding of the field AlarmDstEndpoint (Reference)**

This field shall be coded as data type Unsigned16.

5.2.5.2 Coding of the field AlarmSrcEndpoint (Reference)

This field shall be coded as data type Unsigned16.

NOTE The value of the field reference for destination is provided by the receiver. The value of the field reference for the source is provided by the sender. The fields are specific for a connection and exchanged during context management operation.

5.2.5.3 Coding of the field PDUType

The coding of this field shall be according to 3.14.2.3 and the individual bits shall have the following meaning:

Bit 0 – 3: PDUType.Type

Shall be set according to the following values to identify the type of the PDU

- Decimal Value (1): means RTA_TYPE_DATA and shall only be used to encode the Data-RTA-PDU
- Decimal Value (2): means RTA_TYPE_NACK and shall only be used to encode the NACK-RTA-PDU
- Decimal Value (3): means RTA_TYPE_ACK and shall only be used to encode the ACK-RTA-PDU
- Decimal Value (4): means RTA_TYPE_ERR and shall only be used to encode the ERR-RTA-PDU

Bit 4 – 7: PDUType.Version

Shall be set to one for this version of the protocol.

Value (1): means Version 1 (RTA_VERS=1)

5.2.5.4 Coding of the field AddFlags

The coding of this field shall be according to 3.14.2.3 and the individual bits shall have the following meaning:

Bit 0 – 3: AddFlags.WindowSize

This field shall be set to 1.

Bit 4: AddFlags.TACK

Shall be set by the sender within a RTA-PDU to control the acknowledge behavior of the receiver.

The value 1 shall be set for immediate acknowledge in a Data-RTA-PDU.

The value 0 shall be set for ERR-RTA-PDU, ACK-RTA-PDU, and NACK-RTA-PDU but the receiver shall not check it.

Bit 5 – 7: reserved

This field shall be set to zero but not checked at the receiver.

5.2.5.5 Coding of the field SendSeqNum

This field shall be coded as data type Unsigned16 with the following values:

Hexadecimal (0x0000 – 0x7FFF)

contains a valid SendSeqNum of a Data-RTA-PDU

This field contains the number of the Data-RTA-PDU. The first issued Data-RTA-PDU shall contain the SendSeqNum 0. The incrementing and comparison of this number is done using modulo 2^{15} operation.

Hexadecimal (0xFFFFE, 0xFFFF)

These values are used to synchronize sender and receiver after establishment of the application relationship. The value 0xFFFF indicates the first Data-RTA-PDU. The value 0xFFFFE indicates that there was no reception of a Data-RTA-PDU before.

NOTE The first Data-RTA-PDU sets SendSeqNum=0xFFFF and AckSeqNum=0xFFFFE. 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.

5.2.5.6 Coding of the field AckSeqNum

This field shall be coded as data type Unsigned16 with the following values:

Hexadecimal (0x0000 – 0x7FFF)

contains a valid AckSeqNum

This field contains the number of the Data-RTA-PDU that is expected to be acknowledged or which is acknowledged.

Hexadecimal (0xFFFF, 0xFFFFE)

contains a initial AckSeqNum

The value 0xFFFFE indicates the there was no Data-RTA-PDU received before. The value 0xFFFF indicates acknowledges the reception of the first Data-RTA-PDU.

5.2.5.7 Coding of the field VarPartLen

This field shall be coded as data type Unsigned16 with the following values:

Decimal (0 – 1432)

contains the number of octets of the following user data

The value 0 shall not be used within a Data-RTA-PDU.

The value shall be 18 if a Data-RTA-PDU contains an Alarm-Ack-PDU.

The value shall be 22 if a Data-RTA-PDU contains an Alarm-Notification-PDU without additional data.

The value shall be in the range from 25 to 1432 if a Data-RTA-PDU contains an Alarm-Notification-PDU with data.

The value 4 shall be used within an ERR-RTA-PDU.

The value 0 shall be used within an ACK-RTA-PDU and a NACK-RTA-PDU.

NOTE Padding octets does not affect the field VarPartLen.

5.2.6 Coding section related to RTA-SDU specific fields

5.2.6.1 Coding of the field AlarmType

This field shall be coded as data type Unsigned16. This field shall be coded with the values according to Table 151.

Table 151 — AlarmType

Value (hexadecimal)	Meaning
0x0000	reserved
0x0001	Diagnosis appears
0x0002	Process
0x0003	Pull
0x0004	Plug
0x0005	Status
0x0006	Update
0x0007	Redundancy
0x0008	Controlled by supervisor
0x0009	Released
0x000A	Plug Wrong Submodule
0x000B	Return of Submodule
0x000C	Diagnosis disappears
0x000D	Multicast Communication Mismatch
0x000E	Port Data Change Notification
0x000F	Sync Data Changed Notification
0x0010	Isochronous Mode Problem Notification
0x0011 – 0x001F	reserved
0x0020 – 0x007F	manufacturer specific
0x0080 – 0x00FF	reserved for profiles
0x0100 – 0xFFFF	reserved

5.2.6.2 Coding of the field AlarmSpecifier

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 10: AlarmSpecifier.SequenceNumber

This field shall be incremented with every AlarmNotification. The allowed range is from 0 to 2047. By means of the SequenceNumber the receiver detects duplications and reflects the value of the field in the AlarmAck.

Bit 11: AlarmSpecifier.ChannelDiagnosis

This field shall be coded with the values according to Table 31. For all other AlarmType's this field shall be set to zero.

Table 152 — AlarmSpecifier.ChannelDiagnosis

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	means that the SubslotNumber contains no ChannelDiagnosis	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch
0x01	means that the SubslotNumber contains at least one ChannelDiagnosis	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch

Bit 12: AlarmSpecifier.ManufacturerSpecificDiagnosis

This field shall be coded with the values according to Table 32. For all other AlarmType's this field shall be set to zero.

Table 153 — AlarmSpecifier.ManufacturerSpecificDiagnosis

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	means that the SubslotNumber contains no ManufacturerSpecificDiagnosis	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch
0x01	means that the SubslotNumber contains at least one ManufacturerSpecificDiagnosis	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch

Bit 13: AlarmSpecifier.SubmoduleDiagnosisState

This field shall be coded with the values according to Table 33. For all other AlarmType's this field shall be set to zero.

Table 154 — AlarmSpecifier.SubmoduleDiagnosisState

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	error free Furthermore, it indicates that all reported diagnosis has been cleared. An individual "disappears" notification can be omitted. However, even in this case a channel diagnosis locally treated as status and/or a Manufacturer Specific Diagnosis treated as status may be present.	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch
0x01	At least one diagnosis exists at the submodul. It could be one or more channel diagnosis and/or one or more Manufacturer Specific Diagnosis.	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch

Bit 14: reserved**Bit 15: AlarmSpecifier.ARDiagnosisState**

This field shall be coded with the values according to Table 155. For all other AlarmType's this field shall be set to zero.

Table 155 — AlarmSpecifier.ARDiagnosisState

Value (hexadecimal)	Meaning	Usage within AlarmType
0x00	error free (related to the AR) Furthermore, it indicates that all reported diagnosis related to the AR has been cleared. An individual "disappears" notification can be omitted. However, even in this case a channel diagnosis locally treated as status and/or a Manufacturer Specific Diagnosis treated as status may be present.	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch
0x01	At least one diagnosis exists at the AR. It could be one or more channel diagnosis and/or one or more Manufacturer Specific Diagnosis.	Diagnosis, Redundancy, Port data change notification, sync data change notification, isochronous mode problem notification, multicast communication mismatch

5.2.7 Coding section related to CL-RPC-PDU**5.2.7.1 Coding of the field RPCVersion**

This field shall be coded as data type Unsigned8. The value shall be 4.

5.2.7.2 Coding of the field RPCPacketType

This field shall be coded as data type Unsigned8.

The values shall be encoded according to Table 156.

Table 156 — 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.

5.2.7.3 Coding of the field RPCFlags

This field shall be coded as data type 3.14.2.3.

The values shall be encoded according to Table 157.

Table 157 — RPCFlags

Bit	Meaning if set to 1
Bit 0	implementation specific, shall be set 0
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 0

5.2.7.4 Coding of the field RPCFlags2

This field shall be coded as data type 3.14.2.3.

The values shall be encoded according to Table 158.

Table 158 — 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 6	reserved
Bit 7	reserved

5.2.7.5 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 3.14.2.3 and the individual bits shall have the meaning defined in Table 159:

Table 159 — RPCDRep.Character- and IntegerEncoding

Bit	Field Name	Value	Meaning
0 – 3	RPCDRep.CharacterEncoding	0	ASCII
		1	EBCDIC
4 – 7	RPCDRep.IntegerEncoding	0	big endian
		1	little endian

NOTE The encoding rules are only applied to the RPCHeader, NDR substitutions and NDREPMMap substitutions within PROFINET IO. It only uses Unsigned8, Unsigned16, or Unsigned32 there. The user data of the PROFINET IO service definitions are not influenced because everything is an opaque OctetString there.

RPCDRep Octet 2

The values of the second octet shall be encoded according to Table 160.

Table 160 — 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.

5.2.7.6 Coding of the field RPCSerialHigh

This field shall be coded as data type Unsigned8. The value contains the high byte of the fragment number of the call.

5.2.7.7 Coding of the field RPCSerialLow

This field shall be coded as data type Unsigned8. The value contains the low byte of the fragment number of the call.

NOTE The value of these fields is incremented which each transmission even with a transmission repetition in distinction to the field FragmentNmb.

5.2.7.8 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)

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 NDREPMMapPDU. If it is part of the NDRDataxxx PDUs then only Big Endian Format is used.

The ordering of octets of Data4 is shown in Table 161:

Table 161 — RPCObjectUUID.Data4

Octet	Meaning
1, 2	value see Table 4
3	Instance High
4	Instance Low
5	DeviceID High
6	DeviceID Low
7	VendorID High
8	VendorID Low

Defined values for PROFINET IO items are shown in Table 4.

Table 162 — RPCObjectUUID – defined values

Value	Description
UUID_IO_ObjectInstance_XYZ DEA00000-6C97-11D1-8271-{xxxxyyyyzzzz}	Identifies a special object instance within a physical device in case there are more than one where
xxxx	represents the instance or node number
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

5.2.7.9 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

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 NDREPMAP PDU. If it is part of the NDRDataxxx PDUs then only Big Endian Format is used.

Defined values for PROFINET IO items are shown in Table 163.

Table 163 — RPCInterfaceUUID – defined values

Value	Description
UUID_IO_DeviceInterface DEA00001-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO device uniquely.
UUID_IO_ControllerInterface DEA00002-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO controller uniquely.
UUID_IO_SupervisorInterface DEA00003-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO supervisor uniquely.
UUID_IO_ParameterServerInterface DEA00004-6C97-11D1-8271-00A02442DF7D	Identifies the interface of an IO parameter server uniquely.
UUID_EPMap_Interface E1AF8308-5D1F-11C9-91A4-08002B14A0FA	Identifies the interface of the endpoint mapper. The RPCInterfaceVersion shall be 0x00030000.

5.2.7.10 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 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 NDREPMAP PDU. If it is part of the NDRDataxxx PDUs then only Big Endian Format is used. The response mirrors the content of the field of the request.

5.2.7.11 Coding of the field RPCServerBootTime

This field shall be coded as data type Unsigned32. It shall be set to a value different from zero.

NOTE Idempotent functions do not need to maintain this value. Only the endpointmapper is allowed to answer with zero.

5.2.7.12 Coding of the field RPCInterfaceVersion

This field shall be coded as data type Unsigned32. The value shall be set to 0x00010000. The representation on the wire is according to the value in the field "RPCDRep.IntegerEncoding".

The major version shall be set to 1 for this version (high word).

The minor version shall be set to 0 for this version (low word).

NOTE Other values may be used for compatible extensions.

5.2.7.13 Coding of the field **RPCSequenceNmb**

This field shall be coded as data type Unsigned32.

5.2.7.14 Coding of the field **RPCOperationNmb**

This field shall be coded as data type Unsigned16. The **RPCOperationNmb** identifies the PROFINET IO service supported by the PROFINET IO interfaces

- IO device,
- IO controller and
- IO supervisor.

The values for PROFINET IO services are defined in Table 164.

Table 164 — RPCOperationNmb (IO device, controller and supervisor)

Value (decimal)	Service	Usage
0	Connect	
1	Release	
2	Read	only valid with ARUUID<>0
3	Write	only valid with ARUUID<>0
4	Control	
5	Read Implicit	only valid with ARUUID=0
6 – 65535	reserved	

The values for endpoint mapper services according to Table 165 shall be used.

Table 165 — 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
7 – 65535		reserved

5.2.7.15 Coding of the field **RPCInterfaceHint**

This field shall be coded as data type Unsigned16. The value shall be set to no hint (0xFFFF) for this version.

5.2.7.16 Coding of the field **RPCActivityHint**

This field shall be coded as data type Unsigned16. The value shall be set to no hint (0xFFFF) for this version.

5.2.7.17 Coding of the field **RPCLengthOfBody**

This field shall be coded as data type Unsigned16. The value shall be set to the number of octets of NDRData of the current frame.

NOTE The conveyance of NDRData may require more than one frame. In this case **RPCLengthOfBody** contains only the number of octets for the current frame.

5.2.7.18 Coding of the field **RPCFragmentNmb**

This field shall be coded as data type Unsigned16. The value shall be set to the number of the current fragment.

5.2.7.19 Coding of the field RPCAuthenticationProtocol

This field shall be coded as data type Unsigned8. The value shall be set to zero for no authentication.

5.2.7.20 Coding of the field RPCCancelVersion

This field shall be coded as data type Unsigned32. The value shall be set to zero.

5.2.7.21 Coding of the field RPCCancelID

This field shall be coded as data type Unsigned32.

5.2.7.22 Coding of the field RPCServerIsAccepting

This field shall be coded as data type Unsigned8.

5.2.7.23 Coding of the field RPCVersionFack

This field shall be coded as data type Unsigned8. The value shall be set to zero.

5.2.7.24 Coding of the field RPCPad1

This field shall be coded as data type Unsigned8.

5.2.7.25 Coding of the field RPCWindowSize

This field shall be coded as data type Unsigned16.

5.2.7.26 Coding of the field RPCMaxTsdU

This field shall be coded as data type Unsigned32.

5.2.7.27 Coding of the field RPCMaxFragSize

This field shall be coded as data type Unsigned32.

5.2.7.28 Coding of the field RPCSerialNumber

This field shall be coded as data type Unsigned16.

5.2.7.29 Coding of the field RPCSelAckLen

This field shall be coded as data type Unsigned16.

5.2.7.30 Coding of the field RPCArrayOfSelAck

This field shall be coded as data type Unsigned32.

5.2.7.31 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

Defined values are shown in Table 166.

Table 166 — RPCDataRepresentationUUID – defined values

Value	Description
UUID_DataRepresentation 8A885D04-1CEB-11C9-9FE8-08002B104860	Identifies the RPC data representation uniquely.

5.2.8 Coding section related to BlockHeader specific fields**5.2.8.1 Overview**

The fields described in this subclause are commonly used in different APDUs to distinguish different user content. The TLV coding scheme is used for this reason.

5.2.8.2 Coding of the field BlockType

This field shall be coded as data type Unsigned16 with the values according to Table 167. This field identifies the PDU or data block type of the PROFINET IO service.

Table 167 — BlockType

Value (hexadecimal)	Meaning	Use
0x0000	reserved	not used
0x0001	AlarmNotification High	AlarmNotification-PDU
0x0002	AlarmNotification Low	AlarmNotification-PDU
0x0008	WriteRecordReq	RecordDataWrite
0x8008	WriteRecordRes	RecordDataWrite
0x0009	ReadRecordReq	RecordDataRead
0x8009	ReadRecordRes	RecordDataRead
0x0010	DiagnosisBlock	RecordDataRead, AlarmNotification-PDU
0x0011	MulticastConsumerInfoBlock	AlarmNotification-PDU
0x0012	ExpectedIdentificationDataBlock	RecordDataRead
0x0013	RealIdentificationData	RecordDataRead
0x0014	SubstituteValue	RecordDataRead, RecordDataWrite
0x0015	RecordInputDataObjectElement	RecordDataRead
0x0016	RecordOutputDataObjectElement	RecordDataRead
0x0017	reserved	
0x0018	ARData	RecordDataRead
0x0019	LogData	RecordDataRead
0x001A	APIData	RecordDataRead
0x0020	I&M0	RecordDataRead
0x0021	I&M1	RecordDataRead
0x0022	I&M2	RecordDataRead
0x0023	I&M3	RecordDataRead
0x0024	I&M4	RecordDataRead
0x8001	Alarm Ack High	AlarmAck-PDU
0x8002	Alarm Ack Low	AlarmAck-PDU
0x0101	ARBlockReq	IODConnectReq
0x8101	ARBlockRes	IODConnectRes
0x0102	IOCRBlockReq	IODConnectReq
0x8102	IOCRBlockRes	IODConnectRes
0x0103	AlarmCRBlockReq	IODConnectReq
0x8103	AlarmCRBlockRes	IODConnectRes
0x0104	ExpectedSubmoduleBlockReq	IODConnectReq
0x8104	ModuleDiffBlock	IODConnectRes, RecordDataRead, IOControlReq, IOSControlReq
0x0105	PrmServerBlockReq	IODConnectReq
0x8105	PrmServerBlockRes	IODConnectRes
0x0106	MCRBlockReq	IODConnectReq
0x0110	IODBlockReq, shall only be used in conjunction with connection establishment phase	IODControlReq (Prm End.req)
0x8110	IODBlockRes, shall only be used in conjunction with connection establishment phase	IODControlRes (Prm End.rsp)
0x0111	IODBlockReq, shall only be used in conjunction with a plug alarm event	IODControlReq (Prm End.req)
0x8111	IODBlockRes, shall only be used in conjunction with a plug alarm event	IODControlRes (Prm End.rsp)
0x0112	IOXBlockReq, shall only be used in conjunction with connection establishment phase	IOControlReq, IOSControlReq (Application Ready.req)

Value (hexadecimal)	Meaning	Use
0x8112	IOXBlockRes, shall only be used in conjunction with connection establishment phase	IOCControlRes IOSControlRes (Application Ready.rsp)
0x0113	IOXBlockReq, shall only be used in conjunction with a plug alarm event	IOCControlReq, IOSControlReq (Application Ready.req)
0x8113	IOXBlockRes, shall only be used in conjunction with a plug alarm event	IOCControlRes IOSControlRes (Application Ready.rsp)
0x0114	ReleaseBlockReq	IODReleaseReq
0x8114	ReleaseBlockRes	IODReleaseRes
0x0200	PDPorDataCheck	RecordDataRead RecordDataWrite
0x0201	PDevData	RecordDataRead
0x0202	PDPorDataAdjust	RecordDataRead RecordDataWrite
0x0203	PDSyncData	RecordDataRead RecordDataWrite
0x0204	IsochronousModeData	RecordDataRead RecordDataWrite
0x0205	PDIRHeaderData	RecordDataRead RecordDataWrite
0x0206	PDIRGlobalData	RecordDataRead RecordDataWrite
0x0207	PDIRFrameData	RecordDataRead RecordDataWrite
0x0208	PTP	PTP-RTA-PDU
0x0209	Sub block for adjusting DomainBoundary	PDPorDataAdjust
0x020A	Sub block for checking Peers	PDPorDataCheck
0x020B	Sub block for checking PropagationDelayFactor	PDPorDataCheck
0x020C	Sub block for checking MAUType	PDPorDataCheck
0x020D	Sub block for checking DomainBoundary	PDPorDataCheck
0x020E	Sub block for adjusting MAUType	PDPorDataAdjust
0x020F	PDPorDataReal	RecordDataRead RecordDataWrite
other	reserved	not used

5.2.8.3 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 Type and Length.

5.2.8.4 Coding of the field BlockVersionHigh

This field shall be coded as data type Unsigned8. The value shall be set to 0x01.

5.2.8.5 Coding of the field BlockVersionLow

This field shall be coded as data type Unsigned8. The value shall be set to 0x00.

5.2.9 Coding section related to common address fields

5.2.9.1 Overview

The fields described in this subclause are commonly used in different APDUs to address user data.

5.2.9.2 Coding of the field API

This field shall be coded as data type Unsigned32. The default AP is addressed by 0. All other values shall not be used and are reserved for profile definitions. Each new AP shall be addressed by x with $1 \leq x \leq 0xFFFFFFFF$.

NOTE The API is assigned by PROFIBUS International (PI).

5.2.9.3 Coding of the field SlotNumber

This field shall be coded as data type Unsigned16. The coding shall be according to Table 168.

Table 168 — 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

5.2.9.4 Coding of the field SubslotNumber

This field shall be coded as data type Unsigned16. The coding shall be according to Table 169.

Table 169 — SubslotNumber

Value (hexadecimal)	Meaning of SubslotNumber
0	Identifies the module itself, It does not address a submodule.
1 – 0x7FFF	The first usable subslot for submodules is one. The last usable subslot for submodules is 0x7FFF. It may contain gaps.
0x8000 – 0x8FFF	Used for 16 interface modules with up to 255 ports; 0x81PP with I counting interfaces and P counting ports
0x9000 – 0xFFFF	reserved

5.2.9.5 Coding of the field Index

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 TargetUUID shall only be present and evaluated in conjunction with the defined indices.

Expression 1:

The rules to evaluate the address parameter shall be applied:

A) RPCOperationNmb == implicit Read

a) ARUUID == NIL (implicit AR)

TargetARUUID == NIL, API, Slot Number, Subslot Number, Index shall be evaluated

TargetARUUID != NIL, API, Slot Number, Subslot Number, Index shall be evaluated

b) ARUUID != NIL (implicit AR)

not allowed

B) RPCOperationNmb == explicit Read / Write

a) ARUUID == NIL (implicit AR)

not allowed

b) ARUUID != NIL (established AR)

API, Slot Number, Subslot Number, Index shall be evaluated

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.

The coding for user specific record data shall be according to Table 170.

Table 170 — Index (user specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, SlotNumber, SubslotNumber and TargetARUUID
0-0x7FFF	user specific RecordData	Expression 1 is applied.

The coding for submodule specific defined record data shall be according to Table 171.

Table 171 — Index (subslot specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUID, SlotNumber, SubslotNumber and TargetARUID
0x8000	ExpectedIdentificationData for one subslot	Expression 1 is applied.
0x8001	RealIdentificationData for one subslot	Expression 1 is applied.
0x8002 – 0x8009	reserved	reserved
0x800A	ChannelDiagnosisData for one subslot	Expression 1 is applied.
0x800B	ManufacturerSpecificDiagnosisData for one subslot with ChannelProperties.Specifier=Error appears	Expression 1 is applied.
0x800C	ManufacturerSpecificDiagnosisData for one subslot (all)	Expression 1 is applied.
0x800D – 0x801D	reserved	reserved
0x801E	SubstituteValues for one subslot	Expression 1 is applied.
0x801F – 0x8027	reserved	reserved
0x8028	RecordInputDataObjectElement for one subslot	Expression 1 is applied.
0x8029	RecordOutputDataObjectElement for one subslot	Expression 1 is applied.
0x802A	PDPortDataReal for one subslot	Expression 1 is applied.
0x802B	PDPortDataCheck for one subslot	Expression 1 is applied.
0x802C	PDIRData for one subslot	Expression 1 is applied.
0x802D	Real PDSyncData for one subslot	Expression 1 is applied.
0x802E	Expected PDSyncData for one subslot	Expression 1 is applied.
0x802F	PDPortDataAdjust for one subslot	Expression 1 is applied.
0x8030	IsochronousModeData for one subslot	Expression 1 is applied.
0x8031 – 0xAFF0	reserved	reserved
0xAFF0	I&M0	Expression 1 is applied.
0xAFF1	I&M1	Expression 1 is applied.
0xAFF2	I&M2	Expression 1 is applied.
0xAFF3	I&M3	Expression 1 is applied.
0xAFF4	I&M4	Expression 1 is applied.
0xAFF5 – 0xAFFF	I&M5 – I&M15	Expression 1 is applied.
0xB000 – 0xBFFF	reserved for profiles	Expression 1 is applied.

The coding for module specific defined record data shall be according to Table 172.

Table 172 — Index (slot specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUID, SlotNumber, SubslotNumber and TargetARUID
0xC000	ExpectedIdentificationData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC001	RealIdentificationData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC002 – 0xC009	reserved	reserved
0xC00A	ChannelDiagnosisData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC00B	ManufacturerSpecificDiagnosisData for one slot with ChannelProperties.Specifier=Error appears	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC00C	ManufacturerSpecificDiagnosisData for one slot (all)	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC00D – 0xC029	reserved	reserved
0xC02A	PDPortDataReal for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC02B	PDPortDataCheck for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC02C	PDIRData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC02D	Real PDSyncData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, SlotNumber, SubslotNumber and TargetARUUID
0xC02E	Expected PDSyncData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC02F	PDPortDataAdjust for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC030	IsochronousModeData for one slot	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.
0xC031 – 0xCFFF	reserved	reserved
0xD000 – 0xDFFF	reserved for profiles	Expression 1 is applied. Additionally, the SubslotNumber shall be ignored.

The coding for AR specific defined record data shall be according to Table 173.

Table 173 — Index (AR specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, SlotNumber, SubslotNumber and TargetARUUID
0xE000	ExpectedIdentificationData for one AR	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE001	RealIdentificationData for one AR	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE002	ModuleDiffBlock for one AR	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE002 – 0xE009	reserved	reserved
0xE00A	ChannelDiagnosisData for one AR	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE00B	ManufacturerSpecificDiagnosisData for one AR with ChannelProperties.Specifier=Error appears	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE00C	ManufacturerSpecificDiagnosisData for one AR (all)	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE00D – 0xE029	reserved	reserved
0xE02A	PDPortDataReal for one AR	Expression 1 is applied.
0xE02B	PDPortDataCheck for one AR	Expression 1 is applied.
0xE02C	PDIRDData for one AR	Expression 1 is applied.
0xE02D	Real PDSyncData for one AR	Expression 1 is applied.
0xE02E	Expected PDSyncData for one AR	Expression 1 is applied.
0xE02F	PDPortDataAdjust for one AR	Expression 1 is applied.
0xE030	IsochronousModeData for one AR	Expression 1 is applied.
0xE031 – 0xE03F	reserved	reserved
0xE040	MultipleWrite	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.
0xE041 – 0xEBFF	reserved	reserved
0xEC00 – 0xEFFF	reserved for profiles	Expression 1 is applied. Additionally, the SlotNumber and the SubslotNumber shall be ignored.

The coding for API specific defined record data shall be according to Table 174 if no AR has been established.

Table 174 — Index (API specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, SlotNumber, SubslotNumber and TargetARUUID
0xF000	RealIdentificationData for one API	Expression 1 is applied. Additionally, the TargetARUUID, SlotNumber and the SubslotNumber shall be ignored.

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, SlotNumber, SubslotNumber and TargetARUUID
0xF001 – 0xF009	reserved	reserved
0xF00A	ChannelDiagnosisData for one API	Expression 1 is applied. Additionally, the TargetARUUID, SlotNumber and the SubslotNumber shall be ignored.
0xF00B	ManufacturerSpecificDiagnosisData for one API with ChannelProperties.Specifier=Error appears	Expression 1 is applied. Additionally, the TargetARUUID, SlotNumber and the SubslotNumber shall be ignored.
0xF00C	ManufacturerSpecificDiagnosisData for one API (all)	Expression 1 is applied. Additionally, the TargetARUUID, SlotNumber and the SubslotNumber shall be ignored.
0xF00C – 0xF01F	reserved	reserved
0xF020	ARData for one API	Expression 1 is applied. Additionally, the TargetARUUID, SlotNumber and the SubslotNumber shall be ignored.
0xF021 – 0xF3FF	reserved	reserved
0xF400 – 0xF7FF	reserved for profiles	Expression 1 is applied. Additionally, the TargetARUUID, SlotNumber and the SubslotNumber shall be ignored.

The coding for device specific defined record data shall be according to Table 175.

Table 175 — Index (device specific)

Value (hexadecimal)	Meaning of index	Meaning of fields ARUUID, SlotNumber, SubslotNumber and TargetARUUID
0xF800 – 0xF81F	reserved	reserved
0xF820	ARData	API, ARUUID, TargetARUUID, SlotNumber and SubslotNumber shall be ignored.
0xF821	APIData	API, ARUUID, TargetARUUID, SlotNumber and SubslotNumber shall be ignored.
0xF822 – 0xF82F	Reserved	reserved
0xF830	LogData	API, ARUUID, TargetARUUID, SlotNumber and SubslotNumber shall be ignored.
0xF831	PDevData	API, ARUUID, TargetARUUID, SlotNumber and SubslotNumber shall be ignored.
0xF832 – 0xFBFF	Reserved	reserved
0xFC00 – 0xFFFF	reserved for profiles	API, ARUUID, TargetARUUID, SlotNumber and SubslotNumber shall be ignored.

5.2.10 Coding section related to IP and UDP

5.2.10.1 Coding of the field UDP_SrcPort

The encoding of the fields shall be according to RFC768. Defined values are shown in Table 176.

Table 176 — UDP_SrcPort

Value (hexadecimal)	Meaning	Usage
0x8892	IANA_PROFINET_IO_UDP_UNICAST_PORT	UDP-RTC-PDU and UDP-RTA-PDU
0x8893	IANA_PROFINET_IO_UDP_MULTICAST_PORT	Reserved
0x8894	IANA_PROFINET_IO_EPM_PORT	NDREMapLookupReq or NDREMapLookupFreeReq

5.2.10.2 Coding of the field UDP_DstPort

The encoding of the fields shall be according to RFC768. Defined values are shown in Table 177.

Table 177 — UDP_DstPort

Value (hexadecimal)	Meaning	Usage
0x8892	IANA_PROFINET_IO_UDP_UNICAST_PORT	UDP-RTC-PDU and UDP-RTA-PDU
0x8893	IANA_PROFINET_IO_UDP_MULTICAST_PORT	Reserved
0x8894	IANA_PROFINET_IO_EPM_PORT	NDREMapLookupReq or NDREMapLookupFreeReq

5.2.10.3 Coding of the field IP_DstIPAddress

The encoding of the fields shall be according to RFC 791 and RFC 1112. Defined values are shown in Table 177.

Table 178 — IP_DstIPAddress

Value (hexadecimal)	Meaning
^a	IANA_PROFINET_IO_MULTICAST_IP_ADDRESS
^a The value will be assigned in the future.	

5.2.11 Coding section related to NDREMapPDU

5.2.11.1 Coding of the field DeviceType

This field shall be coded as data type VisibleString with 25 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 25.

5.2.11.2 Coding of the field OrderID

This field shall be coded as data type VisibleString with 20 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 20.

5.2.11.3 Coding of the field HWRevision

This field shall be coded as data type VisibleString with 5 octets. 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”.

5.2.11.4 Coding of the field SWRevisionPrefix

This field shall be coded as data type VisibleString with 1 octet. 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

5.2.11.5 Coding of the field SWRevision

This field shall be coded as data type VisibleString with 9 octets. 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”.

NOTE As an example SWRevision could be “<Blank><Blank>1<Blank><Blank>0<Blank><Blank>0”. After each group of three octets a “.” should be inserted for visualization.

5.2.11.6 Coding of the field Blank

This field shall be coded as data type OctetString with 1 octet. The value shall be set to blank.

5.2.11.7 Coding of the field RPCInquiryType

This field shall be coded as data type Unsigned32. The coding shall be according to Table 179.

Table 179 — RPCInquiryType

Value (hexadecimal)	Meaning
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

5.2.11.8 Coding of the field RPCObjectReference

This field shall be coded as data type Unsigned32. The value shall be set to 1.

5.2.11.9 Coding of the field RPCInterfaceReference

This field shall be coded as data type Unsigned32. The value shall be set to 2.

5.2.11.10 Coding of the field RPCInterfaceVersionMajor

This field shall be coded as data type Unsigned16.

5.2.11.11 Coding of the field RPCInterfaceVersionMinor

This field shall be coded as data type Unsigned16.

5.2.11.12 Coding of the field RPCVersionOption

This field shall be coded as data type Unsigned32. The value shall be set to 1.

5.2.11.13 Coding of the field RPCEntryHandleAttribute

This field shall be coded as data type Unsigned32. The value shall be set to 0.

5.2.11.14 Coding of the field RPCEntryHandleUUID

This field shall be coded as data type UUID.

5.2.11.15 Coding of the field RPCMaxEntries

This field shall be coded as data type Unsigned32. The value shall be set at least to 1.

5.2.11.16 Coding of the field RPCNumberOfEntries

This field shall be coded as data type Unsigned32.

5.2.11.17 Coding of the field RPCEntriesOffset

This field shall be coded as data type Unsigned32. The value shall be set to 0.

5.2.11.18 Coding of the field RPCEntriesCount

This field shall be coded as data type Unsigned32.

5.2.11.19 Coding of the field RPCTowerReference

This field shall be coded as data type Unsigned32. The value shall be set to 3.

5.2.11.20 Coding of the field RPCAnnotationOffset

This field shall be coded as data type Unsigned32. The value shall be set to 0.

5.2.11.21 Coding of the field RPCAnnotationLength

This field shall be coded as data type Unsigned32. The value shall be set from 0 to 64.

5.2.11.22 Coding of the field EndTerm

This field shall be coded as data type Unsigned8. The value shall be set to 0.

5.2.11.23 Coding of the field RPCGap

This field shall be coded as data type Unsigned8.

5.2.11.24 Coding of the field RPECTowerLength

This field shall be coded as data type Unsigned32.

5.2.11.25 Coding of the field RPECTowerOctetStringLength

This field shall be coded as data type Unsigned32.

5.2.11.26 Coding of the field RPCEPMapStatus

This field shall be coded as data type Unsigned32. The coding shall be according to Table 180.

Table 180 — RPCEPMapStatus

Value (hexadecimal)	Meaning
0x00000000	RPC okay
0x16C9A0D6	Endpoint not registered
others	reserved

5.2.11.27 Coding of the field RPCFloorCount

This field shall be coded as data type Unsigned16. The value shall be set to 5. The encoding shall be independently of the field RPCDRep always little endian.

5.2.11.28 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.

5.2.11.29 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.

5.2.11.30 Coding of the field RPCID

This field shall be coded as data type Unsigned8. The value shall be set to 0x0D.

5.2.11.31 Coding of the field RPCProtocolIdentifier

This field shall be coded as data type Unsigned8. The value shall be set to 0x0A.

5.2.11.32 Coding of the field RPCServerUDPPort

This field shall be coded as data type Unsigned8. The value shall be set to 0x08.

5.2.11.33 Coding of the field RPCHostAddress

This field shall be coded as data type Unsigned8. The value shall be set to 0x09.

5.2.11.34 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.

5.2.11.35 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.

5.2.12 Coding section related to NDRData

5.2.12.1 Coding of the field ArgsMaximum

This field shall be coded as data type Unsigned32. The value shall be set to the maximum buffer size available for the response.

5.2.12.2 Coding of the field ArgsLength

This field shall be coded as data type Unsigned32. The value shall be set to the number of octets within the PROFINETIOServiceReqPDU or PROFINETIOServiceResPDU.

5.2.12.3 Coding of the field MaximumCount

This field shall be coded as data type Unsigned32. 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.

5.2.12.4 Coding of the field Offset

This field shall be coded as data type Unsigned32. The value shall be set to zero.

5.2.12.5 Coding of the field ActualCount

This field shall be coded as data type Unsigned32. The value shall be set to the same value as in the field ArgsLength.

5.2.12.6 Coding of the field PNIOStatus

This field shall be coded as data type Unsigned32. The content is defined in 5.2.13.74. The PNIOStatus shall be calculated according the following equation.

$$\begin{aligned}
 \text{PNIOStatus} = & \text{ErrorCode} & * & 16777216 & + \\
 & \text{ErrorDecode} & * & 65536 & + \\
 & \text{ErrorCode1} & * & 256 & + \\
 & \text{ErrorCode2} & & &
 \end{aligned} \quad (1)$$

5.2.13 Coding section related to PROFINET IO Service**5.2.13.1 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.

5.2.13.2 Coding of the field SeqNumber

This field shall be coded as data type Unsigned16.

5.2.13.3 Coding of the field ARType

This field shall be coded as data type Unsigned16 with the values according to Table 181.

Table 181 — ARType

Value (hexadecimal)	Meaning	Use
0x0000	reserved	
0x0001	IOCARSsingle	
0x0002	reserved	
0x0003	IOCARCIR	for future versions
0x0004	IOCAR_IOControllerRedundant	for future versions
0x0005	IOCAR_IODeviceRedundant	for future versions
0x0006	IOSAR	the supervisor AR is a special form of the IOCARSsingle
0x0007 – 0xFFFF	reserved	

5.2.13.4 Coding of the field SessionKey

This field shall be coded as data type Unsigned16. The value of the field SessionKey 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 0 for the implicit AR.

5.2.13.5 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 ISO/IE C8802-3 MAC address.

If the field ARProperties.CRTransport contains the value 0 this field shall be do not care.

5.2.13.6 Coding of the field IOCRMulticastMACAdd

This field shall be coded as data type OctetString[6]. The value of the field IOCRMulticastMACAdd shall be according to ISO/IEC 8802-3 MAC address.

Shall be set according to the Table 182.

Table 182 — IOCRMulticastMACAdd

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-00-00	DCP
01-0E-CF	00-00-01	MRP frames
01-0E-CF	00-00-02	MRP frames
01-0E-CF	00-00-03 to 00-00-FF	reserved for other applications (e.g. DCP)
01-0E-CF	00-01-00	RT Class 2 multicast communication address
01-0E-CF	00-01-01	RT Class 3 destination address
01-0E-CF	00-01-02	RT Class 3 PTP sync multicast address
01-0E-CF	00-01-03 to 00-01-FF	reserved for further multicast addresses within the PROFINET IO context
01-0E-CF	00-02-00 to 00-02-1F	PTP sync multicast address
01-0E-CF	00-02-20 to 00-02-3F	PTP follow up multicast address
01-0E-CF	00-02-40	PTP delay multicast address and PTP additional delay followup multicast address
01-0E-CF	00-02-41 to FF-FF-FF	reserved

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

The IEEE Organizationally Unique Identifier for PROFINET is 00-0E-CF.

The PTP sync multicast address shall be set according to the Table 183.

Table 183 — PTP sync multicast address

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-02-00	clock
01-0E-CF	00-02-01	time
01-0E-CF	00-02-02 to 00-02-1F	reserved

The PTP follow up multicast address shall be set according to the Table 184.

Table 184 — PTP follow up multicast address

Value OUI (Multicast) (hexadecimal)	Value ExtensionIdentifier (hexadecimal)	Meaning
01-0E-CF	00-02-20	clock
01-0E-CF	00-02-21	time
01-0E-CF	00-02-22 to 00-02-3F	reserved

It shall be set according to the Table 185.

Table 185 — PROFINET OUI

Value for OUI (hexadecimal)	Meaning
00-0E-CF	global administered individual unicast

Value for OUI (hexadecimal)	Meaning
01-0E-CF	global administered group (multicast) address
02-0E-CF	local administered individual unicast
03-0E-CF	local administered group (multicast) address

5.2.13.7 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 ISO/IEC 8802-3 MAC address.

If the field ARProperties.CRTransport contains the value 0 this field shall be do not care.

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

5.2.13.8 Coding of the field ARProperties

The coding of this field shall be according to 3.14.2.5 and the individual bits shall have the following meaning:

Bit 0 – 2: ARProperties.State

Shall be set according to the Table 186.

Table 186 — ARProperties.State

Value (hexadecimal)	Meaning
0x00	Backup
0x01	Primary
0x02 – 0x07	reserved

Bit 3: ARProperties.SupervisorTakeoverAllowed

Shall be set according to Table 187.

Table 187 — ARProperties.SupervisorTakeoverAllowed

Value (hexadecimal)	Meaning
0x00	not allowed
0x01	allowed

Bit 4: ARProperties.ParametrizationServer

Shall be set according to Table 188.

Table 188 — ARProperties. ParametrizationServer

Value (hexadecimal)	Meaning
0x00	External PrmServer
0x01	CM Initiator

Bit 5 – 6: ARProperties.DataRate

Shall be set according to Table 189.

Table 189 — ARProperties.DataRate

Value (hexadecimal)	Meaning
0x00	at least 100 MB/s or more
0x01	100 MB/s
0x02	1 GB/s
0x03	10 GB/s

Bit 7: ARProperties.reserved_1

Shall be set to 0.

Bit 8: ARProperties.DeviceAccess

Shall be set according to Table 190.

Table 190 — ARProperties.DeviceAccess

Value (hexadecimal)	Meaning
0x00	Only the submodules from the ExpectedSubmoduleBlock are accesible
0x01	Submodule access is controlled by Device application

Bit 9 – 31: ARProperties.reserved

Shall be set to 0.

5.2.13.9 Coding of the field IOCRProperties

The coding of this field shall be according to 3.14.2.5 and the individual bits shall have the following meaning:

Bit 0 – 3: IOCRProperties.RTClass

Shall be set according to the Table 191.

Table 191 — IOCRProperties.RTClass

Value (hexadecimal)	Meaning	Use
0x00	reserved	
0x01	RTClass1	Data-RTC-PDU
0x02	RTClass2	Data-RTC-PDU
0x03	RTClass3	Data-RTC-PDU
0x04	RTClass1	UDP-RTC-PDU
0x05 – 0x07	reserved	

Bit 4 – 5: IOCRProperties.InterfaceType

Shall be set to zero in this version.

Bit 6-8: IOCRProperties.SendClockSynchronisation

Shall be set to zero in this version.

Bit 9 – 10: IOCRProperties.AddressResolution

Shall be set to zero in this version.

Bit 11: IOCRProperties.MProviderDataStatus

Shall be set according to the Table 193.

Table 192 — IOCRProperties.MProviderDataStatus

Value	Meaning
0	Data status of multicast provider frame is invalid in start up phase till Application Ready
1	Data status of multicast provider frame is valid in start up phase

Bit 12 – 31: IOCRProperties.reserved

Shall be set to 0.

5.2.13.10 Coding of the field NumberOfIODataObjects

This field shall be coded as data type Unsigned16.

5.2.13.11 Coding of the field NumberOfIOCS

This field shall be coded as data type Unsigned16.

5.2.13.12 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 IOCBLOCKReq and IOCBLOCKRes to reference the DataItem. Furthermore, it is used in PDIRData of the Physical Device ASE.

5.2.13.13 Coding of the field IOCRTagHeader

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0-11: IOCRTagHeader.IOCRVLANID

Shall be set according to the Table 193.

Table 193 — IOCRTagHeader.IOCRVLANID

Value (hexadecimal)	Meaning
0x000	no VLAN
0x001	default VLAN
0x002 – 0x7FF	according IEEE 802.1Q

Bit 12: IOCRTagHeader.reserved

Shall be set to 0.

Bit 13 – 15: IOCRTagHeader.IOUserPriority

Shall be set according to the Table 194.

Table 194 — IOCRTagHeader.IOUserPriority

Value (hexadecimal)	Meaning
0x00-0x05	reserved
0x06	IO CR Priority
0x07	reserved

5.2.13.14 Coding of the field IOCRTYPE

This field shall be coded as data type Unsigned16 with the values according to Table 195.

Table 195 — IOCRTYPE

Value (hexadecimal)	Meaning
0x0000	reserved
0x0001	Input CR
0x0002	Output CR
0x0003	Multicast Provider CR
0x0004	Multicast Consumer CR
0x0005 – 0xFFFF	reserved

5.2.13.15 Coding of the field CMInitiatorActivityTimeoutFactor

This field shall be coded as data type Unsigned16 with the values according to Table 196.

Table 196 — CMInitiatorActivityTimeoutFactor

Value (decimal)	Meaning	Use
0	reserved	
1 – 1000	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
1001 – 65535	reserved	

5.2.13.16 Coding of the field StationNameLength

This field shall be coded as data type Unsigned16.

5.2.13.17 Coding of the field CMInitiatorStationName

This field shall be coded as data type OctetString with 1 to 240 octets.

The following syntax applies:

- 1 or more labels, separated by [.]
- total length is 1-240
- label-length is 1-63
- labels consist of [a-z0-9-]
- labels do not start with [-]
- labels do not end with [-]
- labels do not have the form “port-xyz” with x, y, z = 0...9
- station-names do not have the form n.n.n.n, n = 0...999
- labels do only start with ‘xn—’ if RFC 3490 is applied

Furthermore, the definition of RFC 1034, and RFC 3490 shall be applied.

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

EXAMPLE 2 “device-1.bögeholz” is coded as “device-1.xn—bgeholz-90a”

NOTE The field CMLInitiatorStationName is not terminated by zero.

5.2.13.18 Coding of the field ParameterServerStationName

This field shall be coded as data type OctetString with 1 to 240 octets.

The following syntax applies:

- 1 or more labels, separated by [.]
- total length is 1-240
- label-length is 1-63
- labels consist of [a-z0-9-]
- labels do not start with [-]
- labels do not end with [-]
- labels do not have the form “port-xyz” with x, y, z = 0...9
- station-names do not have the form n.n.n.n, n = 0...999
- labels do only start with ‘xn—’ if RFC 3490 is applied

Furthermore, the definition of RFC 1034, and RFC 3490 shall be applied.

NOTE The field ParameterServerStationName is not terminated by zero.

5.2.13.19 Coding of the field ProviderStationName

This field shall be coded as data type OctetString with 1 to 240 octets.

The following syntax applies:

- 1 or more labels, separated by [.]
- total length is 1-240
- label-length is 1-63
- labels consist of [a-z0-9-]
- labels do not start with [-]
- labels do not end with [-]
- labels do not have the form “port-xyz” with x, y, z = 0...9
- station-names do not have the form n.n.n.n, n = 0...999
- labels do only start with ‘xn—’ if RFC 3490 is applied

Furthermore, the definition of RFC 1034, and RFC 3490 shall be applied.

NOTE The field ProviderStationName is not terminated by zero.

5.2.13.20 Coding of the field LLDP_ChassisID

This field shall be coded as data type OctetString with 1 to 240 octets.

The following syntax applies:

- 1 or more labels, separated by [.]
- total length is 1-240
- label-length is 1-63
- labels consist of [a-z0-9-]
- labels do not start with [-]
- labels do not end with [-]
- labels do not have the form “port-xyz” with x, y, z = 0...9
- station-names do not have the form n.n.n.n, n = 0...999
- labels do only start with ‘xn—’ if RFC 3490 is applied

Furthermore, the definition of RFC 1034, and RFC 3490 shall be applied.

NOTE The field LLDP_ChassisID is not terminated by zero.

5.2.13.21 Coding of the field LLDP_PortID

This field shall be coded as data type OctetString with 8 or 14 octets with 1 or 2 labels, separated by [.]

This field contains the name of the local port which shall be used as Port ID subtype locally assigned compliant with IEEE P802-1AB. The value shall be “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 value “port-001” or “port-001.00000” shall be used for the first port submodule for an interface.

Furthermore, the definition of RFC 1034, and RFC 3490 shall be applied.

NOTE The field LLDP_PortID is not terminated by zero.

5.2.13.22 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 1439. The maximum number of used octets shall not exceed the maximum C_SDU size.

5.2.13.23 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 1439. The maximum number of used octets shall not exceed the maximum C_SDU size.

5.2.13.24 Coding of the field LengthIOCS

This field shall be coded as data type Unsigned8. Shall be set according to the Table 197.

Table 197 — LengthIOCS

Value (hexadecimal)	Meaning
0x00	reserved
0x01	one octet for IOCS
0x02 – 0xFF	reserved

5.2.13.25 Coding of the field LengthIOPS

This field shall be coded as data type Unsigned8. Shall be set according to the Table 198.

Table 198 — LengthIOPS

Value (hexadecimal)	Meaning
0x00	reserved
0x01	one octet for IOPS
0x02-0xFF	reserved

5.2.13.26 Coding of the field LengthData

This field shall be coded as data type Unsigned16. The values shall be in the range from 0 to 1439.

5.2.13.27 Coding of the field LengthSubstituteData

This field shall be coded as data type Unsigned16. The values shall be in the range from 0 to 1439.

5.2.13.28 Coding of the field NumberOfAPIs

This field shall be coded as data type Unsigned16. The values shall be in the range from 1 to 0xFFFF.

5.2.13.29 Coding of the field AlarmCRProperties

The coding of this field shall be according to 3.14.2.5 and the individual bits shall have the following meaning:

Bit 0: AlarmCRProperties.Priority

Shall be set according to the Table 199.

Table 199 — AlarmCRProperties.Priority

Value (hexadecimal)	Meaning
0x00	user priority (default), the priority given by the user is used and two alarm resources are available
0x01	use only low priority, user priority is ignored and only one alarm resource is available

Bit 1: AlarmCRProperties.Transport

Shall be set according to Table 200.

Table 200 — AlarmCRProperties.Transport

Value (hexadecimal)	Meaning
0x00	Alarm CR uses Data-RTA-PDU
0x01	Alarm CR uses UDP-RTA-PDU

Bit 2 – 31: AlarmCRProperties.reserved

Shall be set to 0.

5.2.13.30 Coding of the field AlarmCRTagHeaderHigh

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0-11: AlarmCRTagHeaderHigh.AlarmCRVLANID

Shall be set according to the Table 201.

Table 201 — AlarmCRTagHeaderHigh.AlarmCRVLANID

Value (hexadecimal)	Meaning
0x000	no VLAN
0x001	default VLAN
0x002 – 0x7FF	according IEEE 802.1Q

Bit 12: AlarmCRTagHeaderHigh.reserved

Shall be set to 0.

Bit 13 – 15: AlarmCRTagHeaderHigh.AlarmUserPriority

Shall be set according to the Table 202.

Table 202 — AlarmCRTagHeaderHigh.AlarmUserPriority

Value (hexadecimal)	Meaning
0x00 – 0x05	reserved
0x06	Alarm CR Priority High

Value (hexadecimal)	Meaning
0x07	reserved

5.2.13.31 Coding of the field AlarmCRTagHeaderLow

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0-11: AlarmCRTagHeaderLow.AlarmCRVLANID

Shall be set according to the Table 203.

Table 203 — AlarmCRTagHeaderLow.AlarmCRVLANID

Value (hexadecimal)	Meaning
0x000	no VLAN
0x001	default VLAN
0x002-0x7FF	according IEEE 802.1Q

Bit 12: AlarmCRTagHeaderLow.reserved

Shall be set to 0.

Bit 13 – 15: AlarmCRTagHeaderLow.AlarmUserPriority

Shall be set according to the Table 204.

Table 204 — AlarmCRTagHeaderLow.AlarmUserPriority

Value (hexadecimal)	Meaning
0x00-0x04	reserved
0x05	Alarm CR Priority Low
0x06-0x07	reserved

5.2.13.32 Coding of the field AlarmSequenceNumber

This field shall be coded as data type Unsigned16 with the values according to Table 205.

Table 205 — 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	

5.2.13.33 Coding of the field AlarmCRType

This field shall be coded as data type Unsigned16 with the values according to Table 206.

Table 206 — AlarmCRType

Value (hexadecimal)	Meaning
0x0000	reserved
0x0001	Alarm CR
0x0002 – 0xFFFF	reserved

5.2.13.34 Coding of the field RTATimeoutFactor

This field shall be coded as data type Unsigned16 with the values according to Table 207. The time base is 100 ms. The RTATimeout is calculated:

$$\text{RTATimeout} = \text{RTATimeoutFactor} \times 100 \text{ ms} \quad (1)$$

Table 207 — RTATimeoutFactor

Value (hexadecimal)	Meaning	Use
0x0000	reserved	
0x0001 – 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.

5.2.13.35 Coding of the field RTARetries

This field shall be coded as data type Unsigned16. The allowed value range shall be from 3 to 15 retries.

5.2.13.36 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.

5.2.13.37 Coding of the field AddressResolutionProperties

The coding of this field shall be according to 3.14.2.5 and the individual bits shall have the following meaning:

Bit 0 – 2: AddressResolutionProperties.Protocol

Shall be set according to the Table 208.

Table 208 — 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 – 15: reserved

Shall be set to zero in this version.

Bit 16 – 31: AddressResolutionProperties.Factor

This field shall be coded with the values according to Table 209. The time base is one second. The AddressResolutionInterval is calculated:

$$\text{AddressResolutionInterval} = \text{AddressResolutionProperties.Factor} \times 1 \text{ s} \quad (2)$$

Table 209 — AddressResolutionProperties.Factor

Value (hexadecimal)	Meaning
0x0000	reserved
0x0001 – 0x0064	mandatory
0x0065 – 0xFFFF	optional

5.2.13.38 Coding of the field MCITimeoutFactor

This field shall be coded as data type Unsigned16. The time base shall be 100 ms. The allowed value range shall be from 0 to 100. The MCIMonitoringInterval is calculated:

$$\text{MCIMonitoringInterval} = \text{MCITimeoutFactor} \times 100 \text{ ms} \quad (3)$$

5.2.13.39 Coding of fields related to Instance, DeviceID, VendorID**5.2.13.39.1 General**

The Unsigned16 fields Instance, DeviceID, and VendorID are divided into two Unsigned8 fields to define the order of the bytes inside the ObjectUUID OctetString.

5.2.13.39.2 Coding of the field InstanceLow

This field shall be coded as data type Unsigned8.

5.2.13.39.3 Coding of the field InstanceHigh

This field shall be coded as data type Unsigned8.

NOTE The value 0 for both InstanceHigh and InstanceLow is recommended for single instance devices.

5.2.13.39.4 Coding of the field DeviceIDLow

This field shall be coded as data type Unsigned8.

5.2.13.39.5 Coding of the field DeviceIDHigh

This field shall be coded as data type Unsigned8.

NOTE The value 0 for both DeviceIDHigh and DeviceIDLow is reserved.

5.2.13.39.6 Coding of the field VendorIDLow

This field shall be coded as data type Unsigned8.

5.2.13.39.7 Coding of the field VendorIDHigh

This field shall be coded as data type Unsigned8. The value 0xFF is reserved for profiles.

NOTE The value 0 for both VendorIDHigh and VendorIDLow is reserved for IEC 61158-6 Type 3 devices.

5.2.13.40 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 210.

Table 210 — ModuleIdentNumber

Value (hexadecimal)	Meaning
0x00000000	reserved
0x00000001 – 0xFFFFFFFF	manufacturer specific

5.2.13.41 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 211.

Table 211 — SubmoduleIdentNumber

Value (hexadecimal)	Meaning
0x00000000	assigned to the subslot zero only, and manufacturer specific for subslots > 0
0x00000001 – 0xFFFFFFFF	manufacturer specific

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.

5.2.13.42 Coding of the field NumberOfARs

This field shall be coded as data type Unsigned16.

5.2.13.43 Coding of the field NumberOfIOCRs

This field shall be coded as data type Unsigned16.

5.2.13.44 Coding of the field SubmoduleDataLength

This field shall be coded as data type Unsigned16. The range shall be between 0 and 1439.

5.2.13.45 Coding of the field NumberOfModules

This field shall be coded as data type Unsigned16.

5.2.13.46 Coding of the field NumberOfSlots

This field shall be coded as data type Unsigned16.

5.2.13.47 Coding of the field NumberOfSubmodules

This field shall be coded as data type Unsigned16.

5.2.13.48 Coding of the field NumberOfSubslots

This field shall be coded as data type Unsigned16.

5.2.13.49 Coding of the field NumberOfRelatedIOCS

This field shall be coded as data type Unsigned16.

5.2.13.50 Coding of the field NumberOfRelatedIODataObjects

This field shall be coded as data type Unsigned16.

5.2.13.51 Coding of the field ARUUID

This field shall be coded as data type UUID. The value NIL indicates the usage of the implicit AR. All other values identify established ARs.

5.2.13.52 Coding of the field TargetARUUID

This field shall be coded as data type UUID. The value NIL indicates the usage of the implicit AR. All other values identify established ARs.

5.2.13.53 Coding of the field ActualLocalTimeStamp

This field shall be coded as data type Unsigned64. The value contains the current cycle count.

5.2.13.54 Coding of the field LocalTimeStamp

This field shall be coded as data type Unsigned64. The value contains the cycle count.

5.2.13.55 Coding of the field NumberOfLogEntries

This field shall be coded as data type Unsigned16. The value contains the number of log entries. The minimum number shall be 16 for an IO device and 4Kbyte an IO controller.

5.2.13.56 Coding of the field EntryDetail

This field shall be coded as data type Unsigned32 according to protocol machine behavior. The value zero shall be used for no detail.

5.2.13.57 Coding of the field AdditionalValue1 and AdditionalValue2

This field shall be coded as data type Unsigned16. The values shall contain additional user information within negative responses. The value zero indicates no further information.

5.2.13.58 Coding of the field ControlBlockProperties

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 15: ControlBlockProperties.reserved

Shall be set to zero.

5.2.13.59 Coding of the field ControlCommand

The coding of this field shall be according to 3.14.2.4. Only one of the following bits shall be set. The individual bits shall have the following meaning:

Bit 0: ControlCommand.PrmEnd

Shall be set according to the Table 212.

Table 212 — 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

Shall be set according to the Table 213.

Table 213 — 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.	IOCControlReq, IOSControlReq

Bit 2: ControlCommand.Release

Shall be set according to the Table 214.

Table 214 — ControlCommand.Release

Value (hexadecimal)	Meaning	Use
0x00	no Release	
0x01	The IO controller terminates the AR.	IODReleaseReq

Bit 3: ControlCommand.Done

Shall be set according to the Table 215.

Table 215 — ControlCommand.Done

Value (hexadecimal)	Meaning	Use
0x00	no Done	
0x01	An acknowledge is sent.	IODReleaseRes, IOXControlRes, IODControlRes

Bit 4 to 15: ControlCommand.reserved

Shall be set to zero.

5.2.13.60 Coding of the field DataDescription

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: DataDescription.Type

Shall be set according to the Table 216.

Table 216 — DataDescription.Type

Value (hexadecimal)	Meaning
0x00	reserved
0x01	Input
0x02	Output
0x03	reserved

Bit 2-15: DataDescription.reserved

Shall be set to 0.

5.2.13.61 Coding of the field DataLength

This field shall be coded as data type Unsigned16. The values shall contain the length of the C_SDU. The range is from 40 to 1440.

The field DataLength is in minimum the sum of all DataItems. It can also be larger.

5.2.13.62 Coding of the field SubstituteDataLength

This field shall be coded as data type Unsigned16. The values shall contain the length of the C_SDU. The range is from 40 to 1440.

The field DataLength is in minimum the sum of all DataItems. It can also be larger.

5.2.13.63 Coding of the field GAP

This field shall be coded as data type Unsigned8. The values shall be set to zero.

5.2.13.64 Coding of the field Padding

This field shall be coded as data type Unsigned8. The values shall be set to zero.

5.2.13.65 Coding of the field RTCPadding

This field shall be coded as data type Unsigned8.

5.2.13.66 Coding of the field RTAPadding

This field shall be coded as data type Unsigned8.

5.2.13.67 Coding of the field RWPadding

This field shall be coded as data type Unsigned8. The values shall be set to zero.

5.2.13.68 Coding of the field SendClockFactor

These fields shall be coded as data type Unsigned16. The time base is 31,25 us. The value range is from 1 to 128. It is mandatory to support the value 32.

NOTE The SendClockTime corresponds with a value range from 31,25 µs to 4000 µs. At a data rate of 100Mb/s a maximum Ethernet frame contains 1518 or 1522 Bytes. To transfer this kind of frame the SendClockTime should be greater than 122,24µs.

5.2.13.69 Coding of the field ReductionRatio

These fields shall be coded as data type Unsigned16. The value range is from 1 to 16384 in discrete steps.

The usage is defined in Table 217.

Table 217 — Values of ReductionRatio

Value(decimal)	Meaning	PDU	Use
1	mandatory	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values
2	mandatory	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values
4	mandatory	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values
8	mandatory	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values
16	mandatory	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values
32	mandatory, optional for RTClass 3	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values

64	mandatory, optional for RTClass 3	Data-RTC-PDU, optional for UDP-RTC-PDU	with all SendClockFactor values
128	mandatory, optional for RTClass 3	Data-RTC-PDU, UDP-RTC-PDU	with all SendClockFactor values
256	mandatory, optional for RTClass 3	Data-RTC-PDU, UDP-RTC-PDU	for UDP-RTC-PDU with all SendClockFactor values, for Data-RTC-PDU with SendClockFactor up to 64
512	mandatory, optional for RTClass 3	Data-RTC-PDU, UDP-RTC-PDU	for UDP-RTC-PDU with all SendClockFactor values, for Data-RTC-PDU with SendClockFactor up to 32
1024	mandatory	UDP-RTC-PDU	with all SendClockFactor values
2048	mandatory	UDP-RTC-PDU	with all SendClockFactor values
4096	mandatory	UDP-RTC-PDU	with all SendClockFactor values
8192	mandatory	UDP-RTC-PDU	only with SendClockFactor values less or equal 64
16384	mandatory	UDP-RTC-PDU	only with SendClockFactor values less or equal 32
16385 – 65535	reserved		
other	optional		

5.2.13.70 Coding of the field Phase

These fields shall be coded as data type Unsigned16. It shall be less or equal to the related ReductionRatio.

The usage is defined in Table 218.

Table 218 — Values of Phase

Value(decimal)	Meaning	Use
0	reserved	
1 – 16384	mandatory	but shall not exceed the value of ReductionRatio
16385 – 65535	reserved	

5.2.13.71 Coding of the field Sequence

These fields shall be coded as data type Unsigned16. The value range is from 0 to 0xFFFF.

The usage is defined in Table 219.

Table 219 — Values of Sequence

Value(decimal)	Meaning	Use
0	mandatory, sequence undefined	provider protocol machines defines the sequence for each phase
1-0xFFFF	optional, sequence defined	provider protocol machines uses the given sequence for each phase

5.2.13.72 Coding of the field DataHoldFactor and WatchdogFactor

These fields shall be coded as data type Unsigned16. The time base is SendClockFactor multiplied with ReductionRatio of the monitored consumer. The WatchdogTime and the DataHoldTime are calculated:

$$\text{WatchdogTime} = \text{WatchdogFactor} * \text{SendClockFactor} * \text{ReductionRatio} * 31,25 \mu\text{s} \quad (2)$$

$$\text{DataHoldTime} = \text{DataHoldFactor} * \text{SendClockFactor} * \text{ReductionRatio} * 31,25 \mu\text{s} \quad (3)$$

The value range for Data-RTC-PDUs is from 0x0003 to 0x1E00. The DataHoldTime and WatchdogTime shall equal or less 1,92 s.

The value range for UDP-RTC-PDUs is from 0x0003 to 0xF000. The DataHoldTime and WatchdogTime shall equal or less 61,44 s.

The usage of the DataHoldFactor is defined in Table 220.

Table 220 — DataHoldFactor

Value(decimal)	Meaning	Description
0x0000	reserved	
0x0001 – 0x0002	optional	An expiration of the time leads to a AR termination.
0x0003 – 0x000F	mandatory	An expiration of the time leads to a AR termination.
0x0010 – 0x1E00	optional	An expiration of the time leads to a AR termination.
0x1E01 – 0xFFFF	reserved	

The usage of the WatchdogFactor is defined in Table 221.

Table 221 — WatchdogFactor

Value(decimal)	Meaning	Description
0x0000	reserved	
0x0001 – 0x0002	optional	A value “1” leads to a AR termination for one missed Frame.
0x0003 – 0x000F	mandatory	An expiration of the time leads to a check of the corresponding DataHoldFactor.
0x0010 – 0x1E00	optional	An expiration of the time leads to a check of the corresponding DataHoldFactor.
0x1E01 – 0xFFFF	reserved	

5.2.13.73 Coding of the field FrameSendOffset

These fields shall be coded as data type Unsigned32. The time base is 1 ns. The value of the field FrameSendOffset shall be less than SendClockFactor * 31,25 µs.

The usage is defined in Table 222.

Table 222 — Values of FrameSendOffset

Value(decimal)	Meaning	Use
0 – 0x003D08FF	optional, relative send offset to the start of the related reduction ratio and phase	optimized scheduling
0x003D0900 – 0xFFFFFFFFE	reserved	not used
0xFFFFFFFFF	mandatory, best effort	provider protocol machines sends the frame as soon as possible

5.2.13.74 Coding section related to PNIOStatus

5.2.13.74.1 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=0 shall be responded.

NOTE An illegal address combination is for example TargetUUID=NULL and ARUUID=NULL in ReadExpectedIdentification service.

5.2.13.74.2 Coding of the field ErrorCode

This field shall be coded as data type Unsigned8. The usage is defined in Table 223.

Table 223 — Values of ErrorCode for negative responses

Value(hexadecimal)	Meaning	Use
0 to 0x80	Reserved	
0x81	PNIO	LogData
0x82 to 0xCE	Reserved	
0xCF	RTA error	within the ERR-RTA-PDU and UDP-RTA-PDU
0xD0 to 0xD9	Reserved	
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, IOCControlRes, IOSControlRes	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 to 0xEF	Reserved	
0xF0 to 0xFF	Reserved	

5.2.13.74.3 Coding of the field ErrorDecode

This field shall be coded as data type Unsigned8. The usage is defined in Table 224.

Table 224 — Values of ErrorDecode

Value(decimal)	Meaning	Use
0x00 to 0x7F	reserved	
0x80	PNIORW ^{a)} a) equivalent to DPV1	used in context with user error codes of the services Read, Write, Read Input Data, Read Output Data, Read Device Diagnosis, Read AR Data, Read Logbook, Read Expected Identification, Read Real Identification
0x81	PNIO	used in context with other services or internal e.g. RPC errors
0x82 to 0xFF	reserved	

5.2.13.74.4 Coding of the field ErrorCode1 and ErrorCode2

These fields shall be coded as data type Unsigned8.

The field ErrorDecode defines the coding and meaning of ErrorCode1 and ErrorCode2.

The ErrorDecode value PNIORW indicate that the parameters ErrorCode1 shall be set according Table 225.

Table 225 — Coding of ErrorCode1 with ErrorDecode PNIORW

ErrorClass (decimal)	Meaning	ErrorCode (decimal)
0 to 9	not specified	not specified *)
10	application	0 = read error 1 = write error 2 = module failure 3 to 6 = not specified 7 = busy 8 = version conflict 9 = feature not supported 10 to 15 = User specific

ErrorClass (decimal)	Meaning	ErrorCode (decimal)
11	access	0 = invalid index 1 = write length error 2 = invalid slot/subslot 3 = type conflict 4 = invalid area 5 = state conflict 6 = access denied 7 = invalid range 8 = invalid parameter 9 = invalid type 10 = backup 11 to 15 = User specific
12	resource	0 = read constrain conflict 1 = write constrain conflict 2 = resource busy 3 = resource unavailable 4 to 7 = not specified 8 to 15 = User specific
13 to 15	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 indicate that the parameter ErrorCode2 shall be set user specific.

The ErrorDecode value PNIO indicate that the parameter ErrorCode1 shall be set according to Table 226.

Table 226 — Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO

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
		...	...
		15	Error in Parameter StationName
2	Connect Parameter Error Faulty IOCRBlockReq	16 – 255	reserved
		0	Error in Parameter BlockType
		...	...
3	Connect Parameter Error Faulty ExpectedSubmoduleBlockReq	23	Error in Parameter IOCSFrameOffset production
		24-255	reserved
		0	Error in Parameter BlockType
4	Connect Parameter Error Faulty AlarmCRBlockReq	...	...
		7	Error in Parameter SubmoduleLength production
		8 – 255	reserved
5	Connect Parameter Error Faulty PrmServerBlockReq	0	Error in Parameter BlockType
		...	...
		8	Error in Parameter ParameterServerStationName
6	Connect Parameter Error Faulty MCRBlockReq	9 – 255	reserved
		0	Error in Parameter BlockType
		...	...
7 – 19	reserved	8	Error in Parameter ProviderStationName
		9 – 255	reserved
		0 – 255	reserved

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
20	IODControl Parameter Error Faulty ControlBlockConnect	0	Error in Parameter BlockType
		7	Error in Parameter ControlBlockProperties
		8 – 255	reserved
21	IODControl Parameter Error Faulty ControlBlockPlug	0	Error in Parameter BlockType
		7	Error in Parameter ControlBlockProperties
		8 – 255	reserved
22	IOXControl Parameter Error Faulty ControlBlock after a connection establishment	0	Error in Parameter BlockType
		7	Error in Parameter ControlBlockProperties
		8 – 255	reserved
23	IOXControl Parameter Error Faulty ControlBlock a plug alarm	0	Error in Parameter BlockType
		7	Error in Parameter ControlBlockProperties
		8 – 255	reserved
24 – 39	Reserved	0 – 255	reserved
40	Release Parameter Error Faulty ReleaseBlock	0	Error in Parameter BlockType
		7	Error in Parameter ControlBlockProperties
		8 – 255	reserved
41 – 59	reserved	0 – 255	reserved
60	AlarmAck Error Codes	0	Alarm Type Not Supported
		1	Wrong Submodule State
		2 – 255	reserved
61	CMDEV	0 – 255	usage see protocol machines and stored as Logbook entries
62	CMCTL	0 – 255	usage see protocol machines and stored as Logbook entries
63	NRPM	0 – 255	usage see protocol machines and stored as Logbook entries
64	RMPM	0 – 255	usage see protocol machines and stored as Logbook entries
65	ALPMI	0 – 255	usage see protocol machines and stored as Logbook entries
66	ALPMR	0 – 255	usage see protocol machines and stored as Logbook entries
67	LMPM	0 – 255	usage see protocol machines and stored as Logbook entries
68	MMAC	0 – 255	usage see protocol machines and stored as Logbook entries
69	RPC	0 – 255	see Table 227
70	APMR	0 – 255	usage see protocol machines and stored as Logbook entries
71	APMS	0 – 255	usage see protocol machines and stored as Logbook entries
72	CPM	0 – 255	usage see protocol machines and stored as Logbook entries
73	PPM	0 – 255	usage see protocol machines and stored as Logbook entries
74	Used by DCPUCS	0 – 255	usage see protocol machines and stored as Logbook entries
75	Used by DCPUCR	0 – 255	usage see protocol machines and stored as Logbook entries
76	Used by DCPMCS	0 – 255	usage see protocol machines and stored as Logbook entries

Value ErrorCode1 (decimal)	Meaning	Value ErrorCode2 (decimal)	Meaning/Usage
77	Used by DCPMCR	0 – 255	usage see protocol machines and stored as Logbook entries
78	FSPM	0 – 255	usage see protocol machines and stored as Logbook entries
79 – 252	reserved	0 – 255	reserved
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)
		3 – 255	reserved
254	reserved	0 – 255	reserved
255	User specific	0 – 255	User specific

Table 227 — Values of ErrorCode2 for ErrorCode1=RPC

Value (decimal)	Definition	Meaning
0	reserved	
1	CLRPC_ERR_REJECTED	epm or server did reject the call, for further details see “nca_status”
2	CLRPC_ERR_FAULTED	server had a fault while executing the call, see “nca_status”
3	CLRPC_ERR_TIMEOUT	epm or server did not respond within (rtx-max * rtx-timeout)
4	CLRPC_ERR_IN_ARGS	broadcast or maybe “hdr_data” too large, see channel-details
5	CLRPC_ERR_OUT_ARGS	server sent back more than “alloc_len”
6	CLRPC_ERR_DECODE	result of epm-query/break 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	

5.2.13.75 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 228.

Table 228 — ModuleState

Value (hexadecimal)	Meaning	Use
0x0000	no module	e.g.module not plugged
0x0001	wrong module	e.g. ModuleIdentNumber wrong
0x0002	proper module	module is okay but at least one submodule is locked, wrong or missing
0x0003	substitute	module is not the same as requested – but compatible. The IO device was able to adapt by its own decision
0x0004-0xFFFF	reserved	

5.2.13.76 Coding of the field SubmoduleState

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

5.2.13.76.1 Coding if SubmoduleState.FormatIndicator == 1**Bit 0 – 5: SubmoduleState.AddInfo**

Shall be set according to the Table 229.

Table 229 — SubmoduleState.AddInfo

Value (hexadecimal)	Meaning	Description
0x00	None	
0x01	takeover is not allowed	This Submodul is not available for takeover by IOSAR.
0x02 – 0x3F	reserved	

Bit 6: SubmoduleState.DiagInfo

Shall be set according to the Table 230.

Table 230 — SubmoduleState.DiagInfo

Value (hexadecimal)	Meaning	Description
0x00	No DiagnosisData available	There is no DiagnosisData available/stored for this submodule.
0x01	DiagnosisData available	There is no DiagnosisData available for this submodule: It can be read with the corresponding records.

Bit 7 – 10: SubmoduleState.ARInfo

Shall be set according to the Table 41.

Table 231 — 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 the it is blocked. I.e. 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 an other IOAR
0x04	Locked By IO Supervisor (IOS)	This AR is not owner of the submodule. It is owned by an other IOSAR
0x05 – 0x0F	Reserved	Reserved

Bit 11 – 14: SubmoduleState.IdentInfo

Shall be set according to the Table 42.

Table 232 — SubmoduleState.IdentInfo

Value (hexadecimal)	Meaning
0x00	OK
0x01	Substitute (SU)
0x02	Wrong (WR)
0x03	NoSubmodule (NO)
0x04 – 0x0F	reserved

Bit 15: SubmoduleState.FormatIndicator

Shall be set according to the Table 233.

Table 233 — SubmoduleState.FormatIndicator

Value (hexadecimal)	Meaning
0x00	Coding uses SubmoduleState.Detail
0x01	Coding uses SubmoduleState.IdentInfo, .ARInfo, .DiagInfo and .AddInfo

5.2.13.76.2 Coding if SubmoduleState.FormatIndicator == 0

Bit 0 – 14: SubmoduleState.Detail

Shall be set according to the Table 40.

Table 234 — 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 compatible and the IO device was able to adapt In this case the IO device has to 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 ExpectedSubmoduleRes.

Bit 15: SubmoduleState.FormatIndicator

Shall be set according to the Table 235.

Table 235 — SubmoduleState.FormatIndicator

Value (hexadecimal)	Meaning
0x00	Coding uses SubmoduleState.Detail
0x01	Coding uses SubmoduleState.IdentInfo, .ARInfo and .AddInfo

5.2.13.77 Coding of the field SubmoduleProperties

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 1: SubmoduleProperties.Type

Shall be set according to the Table 236.

Table 236 — 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

Shall be set according to Table 237.

Table 237 — SubmoduleProperties.SharedInput

Value (hexadecimal)	Meaning	Use
0x00	IO controller	IO controller Could be set with all possible values of SubmoduleProperties.Type.
0x01	IO controller shared	Only the shared IO controller shall use this value. The shared IO controller is does not receive alarms and is restricted to read only. Could only be set with all possible values without SubmoduleProperties.Type==0x02.

Bit 3: SubmoduleProperties.ReduceInputSubmoduleDataLength

Shall be set according to Table 238.

Table 238 — 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

Shall be set according to Table 239.

Table 239 — 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 – 15: SubmoduleProperties.reserved

Shall be set to 0.

5.2.13.78 Coding of the field ModuleProperties

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 15: ModuleProperties.reserved

Shall be set to 0.

5.2.13.79 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 240.

Table 240 — 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	may be used in profile specification
0x0200 – 0xFFFF	reserved	

5.2.13.80 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 241.

Table 241 — 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	

5.2.13.81 Coding of the field UDPCMInitiatorPort

This field shall be coded as data type Unsigned16. This field shall have the value 0x8892.

5.2.13.82 Coding of the field UDPCMResponderPort

This field shall be coded as data type Unsigned16. This field shall have the value 0x8892.

5.2.13.83 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. This field shall contain the value of the reference of the IO device within AlarmCRBlockRes and ARData.

5.2.13.84 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 within ARData.

5.2.13.85 Coding of the field MaxAlarmDataLength

This field shall be coded as data type Unsigned16. This field shall contain the maximal value of the AlarmNotification-PDU. The allowed range shall be from 200 to 1432.

5.2.13.86 Coding of the field ParameterServerProperties

This field shall be coded as data type Unsigned32. This field is reserved for future use.

5.2.13.87 Coding of the I&M Records**5.2.13.87.1 Coding of the field IM_Serial_Number**

This field shall be coded as data type VisibleString with 16 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 16.

5.2.13.87.2 Coding of the field IM_Hardware_Revision

This field shall be coded as data type Unsigned16 with the values according to Table 242.

Table 242 — IM_Hardware_Revision

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Hardware revision

5.2.13.87.3 Coding of the field IM_SWRevision_Functional_Enhancement

This field shall be coded as data type Unsigned8 with the values according to Table 243.

Table 243 — IM_SWRevision_Functional_Enhancement

Value (hexadecimal)	Meaning
0x00 – 0xFF	Functional Enhancement

5.2.13.87.4 Coding of the field IM_SWRevision_Bug_Fix

This field shall be coded as data type Unsigned8 with the values according to Table 244.

Table 244 — IM_SWRevision_Bug_Fix

Value (hexadecimal)	Meaning
0x00 – 0xFF	Bug Fix

5.2.13.87.5 Coding of the field IM_SWRevision_Internal_Change

This field shall be coded as data type Unsigned8 with the values according to Table 245.

Table 245 — IM_SWRevision_Internal_Change

Value (hexadecimal)	Meaning
0x00 – 0xFF	Internal Change

5.2.13.87.6 Coding of the field IM_Revision_Counter

This field shall be coded as data type Unsigned16 with the values according to Table 246.

Table 246 — IM_Revision_Counter

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Revision Counter

5.2.13.87.7 Coding of the field IM_Profile_ID

This field shall be coded as data type Unsigned16 with the values according to Table 247.

Table 247 — IM_Profile_ID

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Using shall be defined by profiles

5.2.13.87.8 Coding of the field IM_Profile_Specific_Type

This field shall be coded as data type Unsigned16 with the values according to Table 248.

Table 248 — IM_Profile_Specific_Type

Value (hexadecimal)	Meaning
0x0000 – 0xFFFF	Using shall be defined by profiles and will be treated as a subversion of the profile

5.2.13.87.9 Coding of the field IM_Version_Major

This field shall be coded as data type Unsigned8 with the values according to Table 249.

Table 249 — IM_Version_Major

Value (hexadecimal)	Meaning
0x00	reserved
0x01	Shall set in this version
0x02 – 0xFF	reserved

5.2.13.87.10 Coding of the field IM_Version_Minor

This field shall be coded as data type Unsigned8 with the values according to Table 250.

Table 250 — IM_Version_Minor

Value (hexadecimal)	Meaning
0x01	Shall set in this version
0x00, 0x02 – 0xFF	reserved

5.2.13.87.11 Coding of the field IM_Supported

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0: IM_Supported.Profil_specific

This field shall be set related to the profiles.

Bit 1: IM_Supported.I&M1

This field shall be set if the I&M1 record contains data.

Bit 2: IM_Supported.I&M2

This field shall be set if the I&M2 record contains data.

Bit 3: IM_Supported.I&M3

This field shall be set if the I&M3 record contains data.

Bit 4: IM_Supported.I&M4

This field shall be set if the I&M4 record contains data.

Bit 5: IM_Supported.I&M5

This field shall be set if the I&M5 record contains data.

Bit 6: IM_Supported.I&M6

This field shall be set if the I&M6 record contains data.

Bit 7: IM_Supported.I&M7

This field shall be set if the I&M7 record contains data.

Bit 8: IM_Supported.I&M8

This field shall be set if the I&M8 record contains data.

Bit 9: IM_Supported.I&M9

This field shall be set if the I&M9 record contains data.

Bit 10: IM_Supported.I&M10

This field shall be set if the I&M10 record contains data.

Bit 11: IM_Supported.I&M11

This field shall be set if the I&M11 record contains data.

Bit 12: IM_Supported.I&M12

This field shall be set if the I&M12 record contains data.

Bit 13: IM_Supported.I&M13

This field shall be set if the I&M13 record contains data.

Bit 14: IM_Supported.I&M14

This field shall be set if the I&M14 record contains data.

Bit 15: IM_Supported.I&M15

This field shall be set if the I&M15 record contains data.

5.2.13.87.12 Coding of the field IM_Tag_Function

This field shall be coded as data type VisibleString with 32 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 32.

5.2.13.87.13 Coding of the field IM_Tag_Location

This field shall be coded as data type VisibleString with 22 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 22.

5.2.13.87.14 Coding of the field IM_Date

This field shall be coded as data type VisibleString with 16 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 16.

String format := "YYYY-MM-DD HH:MM", in accordance with the data format in ISO 8601.

5.2.13.87.15 Coding of the field IM_Descriptor

This field shall be coded as data type VisibleString with 54 octets. The value shall be set manufacturer specific and shall be filled with blanks if it is shorter than 54.

5.2.13.87.16 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.

5.2.14 Coding section related to RPC (NCA Codes)

5.2.14.1 Coding section of the field RPCNCAFaultStatus

The RPC specific NCA error codes shall be coded as Unsigned32 with the values according to Table 251.

Table 251 — 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
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

5.2.14.2 Coding section of the field RPCNCAResjectStatus

The RPC specific NCA error codes shall be coded as Unsigned32 with the values according to Table 252.

Table 252 — Values of NCAResjectStatus

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
0x1C010006	nca_wrong_boot_time

Value (hexadecimal)	Definition
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

5.2.15 Coding section related to Alarm and Diagnosis PDUs

5.2.15.1 Coding of the field UserStructureIdentifier

This field shall be coded as data type Unsigned16 with the values according to Table 35. This field identifies the structure of the field Data of the AlarmNotification and structure of the field Data of the alarm data.

Table 253 — UserStructureIdentifier

Value (hexadecimal)	Meaning	Usage
0x0000 – 0x7FFF	ManufacturerSpecific	In conjunction with alarm type Diagnosis appears/disappears ManufacturerSpecific Diagnosis in AlarmNotification and Diagnosis Data. In conjunction with other alarm types the usage is manufacturer specific.
0x8000	ChannelDiagnosis	Shall only be used in conjunction with ChannelDiagnosis in AlarmNotification and Diagnosis Data.
0x8001	Multiple	Shall only be used in conjunction with data, which comply to the “(BlockHeader, Data*)” structure. Furthermore, the BlockType shall always correspond to the used AlarmType.
0x8002	ExtChannelDiagnosisData	Shall only be used in conjunction with ChannelDiagnosisWithAddInfo in AlarmNotification and Diagnosis Data.
0x8002 – 0x8FFF	reserved	
0x9000 – 0x9FFF	reserved for profiles	
0xA000 – 0xFFFF	reserved	

5.2.15.2 Coding of the field ChannelErrorType

This field shall be coded as data type Unsigned16 with the values according to Table 254.

Table 254 — 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 – 0x000F	reserved	Unknown error
0x0010	manufacturer specific recommended for “parametrization fault”	parametrization fault
0x0011	manufacturer specific recommended for “power supply fault”	power supply fault

Value (hexadecimal)	Meaning	Assigned text
0x0012	manufacturer specific recommended for "fuse blown / open"	fuse blown / open
0x0013	manufacturer specific	manufacturer specific
0x0014	manufacturer specific recommended for "ground fault"	ground fault
0x0015	manufacturer specific recommended for "reference point lost"	reference point lost
0x0016	manufacturer specific recommended for "process event lost / sampling error"	process event lost / sampling error
0x0017	manufacturer specific recommended for "threshold warning"	threshold warning
0x0018	manufacturer specific recommended for "output disabled"	output disabled
0x0019	manufacturer specific recommended for "safety event"	safety event
0x001A	manufacturer specific recommended for "external fault"	external fault
0x001B – 0x001F	manufacturer specific	manufacturer specific
0x0020 – 0x00FF	reserved	unknown error
0x0100 – 0x7FFF	manufacturer specific	manufacturer specific
0x8000	Data Transmission Impossible	Data Transmission Impossible
0x8001	Remote Mismatch	Remote Mismatch
0x8002	Medium Redundancy Mismatch	Medium Redundancy Mismatch
0x8003	Sync Mismatch	Sync Mismatch
0x8004	IsochronousMode Mismatch	IsochronousMode Mismatch
0x8005	Multicast CR Mismatch	Multicast CR Mismatch
0x8006	Multicast Provider Status Mismatch	Multicast Provider Status Mismatch
0x8007 – 0x8FFF	reserved	unknown error
0x9000 – 0x9FFF	reserved for profiles	profile specific
0xA000 – 0xFFFF	reserved	Unknown error

5.2.15.3 Coding of the field ChannelNumber

This field shall be coded as data type Unsigned16 with the values according to Table 255.

Table 255 — ChannelNumber

Value (hexadecimal)	Meaning
0x0000 – 0x7FFF	manufacturer specific
0x8000	submodule
0x8001 – 0xFFFF	reserved

NOTE The ChannelNumber 0x8000 references the submodule itself.

5.2.15.4 Coding of the field ChannelProperties

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 7: ChannelProperties.Type

Shall be set according to the Table 256.

Table 256 — ChannelProperties.Type

Value (hexadecimal)	Meaning
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
0x02	2 Bit

Value (hexadecimal)	Meaning
0x03	4 Bit
0x04	8 Bit
0x05	16 Bit
0x06	32 Bit
0x07	64 Bit
0x08 – 0xFF	reserved

Bit 8 – 10: reserved

Bit 11 – 12: ChannelProperties.Specifier

This field shall be coded with the values according to Table 257. The field ChannelProperties.Specifier is related to the addressed channel.

Table 257 — ChannelProperties.Specifier

Value (hexadecimal)	Meaning	Usage
0x00	reserved	never
0x01	error appears	DiagnosisAlarm and Read Device Diagnosis
0x02	error disappears and error free	DiagnosisAlarm and Read Device Diagnosis
0x03	error disappears but other error remain	only DiagnosisAlarm

Bit 13 – 15: ChannelProperties.Direction

Shall be set according to Table 258.

Table 258 — ChannelProperties.Direction

Value (hexadecimal)	Meaning
0x00	manufacturer specific
0x01	Input
0x02	Output
0x03	Input/Output
0x04 – 0x07	reserved

5.2.15.5 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 according to Table 259.

Table 259 — ExtChannelErrorType

ChannelErrorType	Value (hexadecimal)	Meaning
0x8000	0x0001	Link Ready
0x8000	0x0002	Link Fail
0x8000	0x0003	Link OK
0x8000	0x0004	Port Disable
0x8000	0x0005	Port Discarding
0x8000	0x0006	Port Learning
0x8000	0x0007	Port Forwarding
0x8001	0x0001	Remote System Data Mismatch
0x8001	0x0002	Remote System Data Match
0x8001	0x0003	Data Rate Mismatch
0x8001	0x0004	Data Rate Okay
0x8001	0x0005	Duplex Mode Mismatch
0x8001	0x0006	Duplex Mode Okay
0x8001	0x0007	Propagation Delay Mismatch
0x8001	0x0008	Propagation Delay Okay
0x8001	0x0009	PMD Type Mismatch
0x8001	0x000A	PMD Type Okay
0x8002	0x0001	Medium Redundancy Lost

ChannelErrorType	Value (hexadecimal)	Meaning
0x8002	0x0002	Medium Redundancy Okay
0x8003	0x0001	No Sync Message Received
0x8003	0x0002	Wrong PTPSubDomainID
0x8003	0x0003	Wrong IRDataID
0x8003	0x0004	Jitter out of Boundary
0x8003	0x0005	Sync Okay
0x8004	0x0001	Output Time Failure
0x8004	0x0002	Output Time Failure Disappears
0x8004	0x0003	Input Time Failure
0x8004	0x0004	Input Time Failure Disappears
0x8005	0x0001	Multicast Consumer CR timed out
0x8005	0x0002	AddressResolutionFailed
0x8005	0x0003	Multicast Consumer CR Okay
0x8005	0x0004	Multicast Consumer CR Okay with M-Provider Status Mismatch
0x8006	0x0001	Provider state changed to value bad
0x8006	0x0002	Provider state changed to value good

5.2.15.6 Coding of the field ExtChannelAddValue

This field shall be coded as data type Unsigned32.

5.2.16 Coding section related to Physical Device Port Data

5.2.16.1 Coding of the field OwnPortID

This field shall be coded as data type OctetString[8] or OctetString[14].

This field contains the name of the local port which shall be used as Port ID subtype locally assigned compliant with IEEE P802-1AB. The value shall be “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 value “port-001” or “port-001.00000” shall be used for the first port submodule for an interface.

5.2.16.2 Coding of the field PeerPortID

This field shall be coded as data type OctetString[255].

5.2.16.3 Coding of the field LengthPeerChassisID

This field shall be coded as data type Unsigned8.

5.2.16.4 Coding of the field PeerChassisID

This field shall be coded as data type OctetString[1..240].

5.2.16.5 Coding of the field PropagationDelayFactor

This field shall be coded as data type Unsigned32. The time base shall be one ns.

5.2.17 Coding section related to Physical Device IR Data

5.2.17.1 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 260.

Table 260 — RxPort

Value (hexadecimal)	Meaning
0x00	local injection
0x01	Port 1
0x02	Port 2
...	...
0xFF	Port 255

5.2.17.2 Coding of the field NumberOfTxPortGroups

This field shall be coded as data type Unsigned8. Allowed values are 1, 3, 5, 7, 9, 11, 13, 15, ... 31, 33. This field shall only count the succeeding TxPortGroupArray entries.

5.2.17.3 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

$$N = M_{\text{highest}} \text{ DIV } 8 + 1 \quad (4)$$

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).

The last TxPortGroup octet (octet N) may not contain all 8 TxPort entries. If M_{highest} MOD 8 ≠ 7 the octet N is not fully filled and the remaining bits shall be set to zero referred to as Padding Bits.

NOTE The term DIV stands for division without rest. The term MOD stands for the rest of the division.

The TxPortGroup of the device's TxPorts shall be structured in ascending order without gaps. The coding of this field shall be according to 3.14.2.3 and the individual bits shall have the following meaning:

Bit 0: TxPortEntry_0

This bit shall be set with the values according to Table 261 if the TxPort with the number that meets m MOD 8 = 0 is present. A padding bit shall be used if no TxPort is present.

Table 261 — TxPortEntry

Value (hexadecimal)	Meaning
0x00	transmission off
0x01	transmission on

The TxPort of local injection is always placed in TxPortEntry_1 of the TxPortGroup octet number one.

Bit 1: TxPortEntry_1

This bit shall be set with the values according to Table 261 if the TxPort with the number that meets m 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 261 if the TxPort with the number that meets m 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 261 if the TxPort with the number that meets m 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 261 if the TxPort with the number that meets m 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 261 if the TxPort with the number that meets m MOD 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 261 if the TxPort with the number that meets m MOD 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 261 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.

where m is the number of the current TxPort with $1 \leq m \leq N$.

5.2.17.4 Coding of the field FrameDetails

This field shall be coded according to 3.14.2.3 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 78.

Table 262 — FrameDetails.SyncFrame

Value (hexadecimal)	Meaning
0x00	no sync frame
0x01	primary sync frame
0x02	secondary sync frame
0x03	reserved

Bit 2 – 3: FrameDetails.MeaningFrameSendOffset

This field shall be coded with the values according to Table 79.

Table 263 — FrameDetails.MeaningFrameSendOffset

Value (hexadecimal)	Meaning
0x00	Field FrameSendOffset specifies the point of time for receiving or transmitting a frame
0x01	Field FrameSendOffset specifies the beginning of the RT Class 3 interval within a phase
0x02	Field FrameSendOffset specifies the ending of the RT Class 3 interval within a phase
0x03	reserved

Bit 4 – 7: reserved**5.2.17.5 Coding of the field AdjustProperties**

This field shall be coded as data type Unsigned16. The individual bits shall have the following meaning:

Bit 0: AdjustProperties.StorageMode

This field shall be coded with the values according to Table 83.

Table 264 — AdjustProperties.StorageMode

Value (hexadecimal)	Meaning
0x00	set data temporary, no remanent storage
0x01	set data and store remanent

Bit 1 – 15: reserved**5.2.17.6 Coding of the field MAUType**

This field shall be coded as data type Unsigned16 with the values according to:

Table 265 — MAUType

Value (hexadecimal)	Meaning
0x0000 – 0x000F	reserved
0x0010	100BASETXFD (Default)
0x0011	reserved
0x0012	100BASEFXFD
0x0013 – 0x0015	reserved
0x0016	1000BASEXFD
0x0017	reserved
0x0018	1000BASELXFD

Value (hexadecimal)	Meaning
0x0019	reserved
0x001A	1000BASEXFD
0x001B – 0x001D	reserved
0x001E	1000BASETFD
0x001F	10GigBASEFX
0x0020 – 0xFFFF	reserved

5.2.17.7 Coding of the field DomainBoundary

This field shall be coded as data type Unsigned32. The individual bits shall be coded with the values according to Table 266.

Table 266 — DomainBoundary

Value	Meaning
0	block Sync/Time telegram of this PTPSubdomain
1	do not block Sync/Time telegram of this PTPSubdomain

5.2.17.8 Coding of the field IRDataID

This field shall be coded as data type UUID.

5.2.17.9 Coding of the field Ethertype

This field shall be coded as described in 5.2.2.2. In addition to this, the only allowed value is 0x8892.

5.2.18 Coding section related to Physical Sync Data

5.2.18.1 Coding of the field PTPSubdomainID

This field shall be coded as data type UUID. The value NULL indicates no synchronisation within the Read Real Sync Data service.

5.2.18.2 Coding of the field SyncProperties

This field shall be coded according to 3.14.2.4 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 82.

Table 267 — SyncProperties.Role

Value (hexadecimal)	Meaning
0x00	local sync
0x01	external sync.: RoleClock Slave
0x02	Clock Master
0x03	reserved

Bit 2 – 7: reserved

Bit 8-15: SyncProperties.SyncClass

This field shall be coded with the values according to Table 83.

Table 268 — SyncProperties.SyncClass

Value (hexadecimal)	Meaning
0x00	Test Stratum
0x01	Primary (GPS, atom clock)
0x02	Secondary (directly connected to a primary clock)
0x03	Connected to an external time signal (over boundary clocks)
0x04	Not connected to an external time signal
0x05 – 0xFE	reserved
0xFF	No clock synchronization

5.2.18.3 Coding of the field ReservedIntervalBegin

This field shall be coded as data type Unsigned32. The time base is one ns.

5.2.18.4 Coding of the field ReservedIntervalEnd

This field shall be coded as data type Unsigned32. The time base is one ns.

5.2.18.5 Coding of the field SyncSendFactor

This field shall be coded as data type Unsigned16. The time base is one ns.

5.2.18.6 Coding of the field SyncFrameAddress

This field shall be coded as data type Unsigned8.

5.2.18.7 Coding of the field PLLWindow

This field shall be coded as data type Unsigned32. The time base is one ns. The value 0 indicates no PLLWindow defined within the Read Real Sync Data service.

5.2.19 Coding section related to Isochrone Mode Data**5.2.19.1 Coding of the field TimeDataCycle**

This field shall be coded as data type Unsigned16. The time base is 31,25 µs.

5.2.19.2 Coding of the field TimeIOInput

This field shall be coded as data type Unsigned32. The time base is ns.

5.2.19.3 Coding of the field TimeIOOutput

This field shall be coded as data type Unsigned32. The time base is ns.

5.2.19.4 Coding of the field TimeIOInputValid

This field shall be coded as data type Unsigned32. The time base is ns.

5.2.19.5 Coding of the field TimeIOOutputValid

This field shall be coded as data type Unsigned32. The time base is ns.

5.2.19.6 Coding of the field ControllerApplicationCycleFactor

This field shall be coded as data type Unsigned16.

5.2.20 Coding section MRP-RTA-PDU specific fields**5.2.20.1 Coding of the field MRP_TypeLength**

The coding of this field shall be according to 3.14.2.4 and the individual bits shall have the following meaning:

Bit 0 – 6: MRP_TypeLength.Type

This field shall be coded with the values according to Table 269.

Table 269 — MRP_Type

Value (hexadecimal)	Meaning	Usage
0x00	EndOfPDU	
0x01	Command	mandatory
0x02 – 0x7E	reserved	
0x7E	Organizationally Specific	optional

Bit 7 – 15: MRP_TypeLength.Length

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

5.2.20.2 Coding of the field MRP_Version

This field shall be coded as data type Unsigned16. This field shall be set to one.

5.2.20.3 Coding of the field MRP_Command

This field shall be coded as Unsigned16 with the values according to Table 270.

Table 270 — MRP_Command

Value (hexadecimal)	Meaning	Usage
0x0000	reserved	
0x0001	Testing	RedundancyMaster is testing ring state
0x0002 – 0x001F	reserved	
0x0020	RedundancyMaster_LinkDownAck	detecting/acknowledging link down at ring port
0x0021	RedundancyMaster_LinkUpAck	detecting/acknowledge link up at ring port
0x0022 – 0x002F	reserved	
0x0030	RedundancySlave_LinkDown	detecting link down at ring port
0x0031	RedundancySlave_LinkUp	detecting link up at ring port
0x0032 – 0xFFFF	reserved	

5.2.20.4 Coding of the field MRP_MAC

This field shall be coded as data type OctetString[6]. The value of the field MRP_MAC shall be according to ISO/IEC 8802-3 MAC address.

5.2.20.5 Coding of the field MRP_Port

This field shall be coded as Unsigned16 with the values according to Table 271.

Table 271 — MRP_Port

Value (hexadecimal)	Meaning	Usage
0x0000	reserved	
0x0001	Port-001	Port 1 as defined in the PhyDev ASE
0x0002	Port-002	Port 2 as defined in the PhyDev ASE
...		
0x00FE	Port-254	Port 254 as defined in the PhyDev ASE
0x00FF	Port-255	Port 255 as defined in the PhyDev ASE
0x0100 – 0xFFFF	reserved	

5.2.20.6 Coding of the field MRP_Info

This field shall be coded as Unsigned16 with the values according to Table 272.

Table 272 — MRP_Info

Value (hexadecimal)	Meaning	Usage
0x0000	reserved	
0x0001	Active	Redundancy Master is in active state
0x0002	Passive	Redundancy Master is in passive state
0x0003 – 0xFFFF	reserved	

5.2.20.7 Coding of the field MRP_Counter

This field shall be coded as Unsigned16 with the values according to Table 273.

Table 273 — MRP_Counter

Value (hexadecimal)	Meaning	Usage
0x0000 – 0x007F	count down for next topologie change event (in ms)	Shows time when redundancy devices do next topologie change. Shall count down in 20ms steps. Only RedundancyMaster should send a zero value!
0x0080 – 0xFFFF	reserved	

5.2.20.8 Coding of the field MRP_Transistion

This field shall be coded as Unsigned16 with the values according to Table 274.

Table 274 — MRP_Transition

Value (hexadecimal)	Meaning	Usage
0x0000 – 0xFFFF	number of transitions between active/passive	used for monitoring this value via a packet sniffer station.

5.2.20.9 Coding of the field MRP_TimeStamp

This field shall be coded as Unsigned32 with the values according to Table 275.

Table 275 — MRP_TimeStamp

Value (hexadecimal)	Meaning	Usage
0x00000000 – 0xFFFFFFFF	actual counter value of 1ms counter	The value is used to determine the maximum travel time of the test packets.

5.2.20.10 Coding of the field MRP_OUI

This field shall be coded as Octet[3] with the Organizationally unique identifier (OUI) as defined ISO/IEC 8802.

5.2.20.11 Coding of the field MRP_SubType

This field shall be coded as Unsigned8. The value is vendor / organizationally specific.

5.2.21 PDU code checking rules**5.2.21.1 Overview**

The rules specified in 5.2.19.1 shall be applied to check FAL PDUs at the receiver.

5.2.22 IODConnectReq**5.2.22.1 ArgsLength**

This field shall be checked according to Table 276.

Table 276 — ArgsLength check

	Parameter	Checking rules	Behaviour on Mismatch
RPCHeader	RPCOperationNmb	== 0 (Connect)	okay, check IODConnectReq-PDU
	RPCOperationNmb	== 1 (Release)	
NDRDataRequest	ArgsLength	== Sum of all BlockLength + Number of Blocks * 4	okay
	ArgsLength	!= Sum of all BlockLength + Number of Blocks * 4	RSP-, 0xDB, 0x81, 0x01, 0x04
IODConnectReq		General: If one OR more unknown blocks are contained in the request	RSP-, 0xDB, 0x81, 0x01, 0x04

5.2.22.2 ARBlockReq

This field shall be checked according to Table 277.

Table 277 — ARBlockReq check

Parameter	Checking rules	Behaviour on Mismatch
BlockType	!= 0x0101	RSP-, 0xDB, 0x81, 0x01, 0
BlockLength	!= 54 + StationNameLength	RSP-, 0xDB, 0x81, 0x01, 1
BlockVersionHigh	≤ GSDML-AR_BlockVersion[high]	okay
BlockVersionHigh	> GSDML-AR_BlockVersion[high]	RSP-, 0xDB, 0x81, 0x01, 2
BlockVersionLow	≤ GSDML-AR_BlockVersion[low]	okay
BlockVersionLow	> GSDML-AR_BlockVersion[low]	RSP-, 0xDB, 0x81, 0x01, 3
ARType	== 1 OR equal to one in the GSDML	okay
ARType	!= 1 OR not equal to one in the GSDML	RSP-, 0xDB, 0x81, 0x01, 4
ARUUID	== NIL	RSP-, 0xDB, 0x81, 0x01, 5
ARUUID	!= NIL	okay
SessionKey	not checked	okay
CMInitiatorMacAdd	If ARProperties.CRTransport == 0 (UDP-Transport)	ignore
CMInitiatorMacAdd	If ARProperties.CRTransport == 1 AND (CMInitiatorMacAdd != multicast address)	okay
CMInitiatorMacAdd	If ARProperties.CRTransport == 1 AND (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	okay
ARProperties.State	If ARType == (IOCARSingle OR IOCARShared OR IOSAR) AND ARProperties.State == Primary	okay
ARProperties.State	If ARType == (IOCARSingle OR IOCARShared OR IOSAR) AND ARProperties.State != Primary	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.State	If ARType != (IOCARSingle OR IOCARShared OR IOSAR) AND ARProperties.State == (Primary OR Backup)	okay
ARProperties.SupervisorTake overAllowed	If ARType == (IOCARSingle OR IOCARShared OR IOCARCIR) AND ARProperties.SupervisorTakeoverAllowed == (allowed OR not allowed)	okay
ARProperties.SupervisorTake overAllowed	If ARType != (IOCARSingle OR IOCARShared OR IOCARCIR) AND ARProperties.SupervisorTakeoverAllowed == (allowed)	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.Parametrization Server	== External PrmServer	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.Parametrization Server	== CM Initiator	okay
ARProperties.DataRate	==0x0 OR 0x01	okay
ARProperties.DataRate	==0x02 OR 0x03 AND not supported in GSDML	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.DataRate	==0x02 OR 0x03 AND supported in GSDML	okay
ARProperties.CRTransport	== 1 (Data-RTC-PDU) Comment: Data-RTC-PDU is mandatory	okay

Parameter	Checking rules	Behaviour on Mismatch
ARProperties.CRTransport	== 0 (UDP-RTC-PDU) AND supported in the GSDML Comment: UDP-RTC-PDU is optional	okay
ARProperties.CRTransport	== 0 and not supported in the GSDML Comment: UDP is optional	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.DeviceAccess	== 1 AND ARType == IOSAR	okay
ARProperties.DeviceAccess	== 1 AND ARType != IOSAR	RSP-, 0xDB, 0x81, 0x01, 9
ARProperties.reserved	== 0	okay
ARProperties.reserved	!= 0	RSP-, 0xDB, 0x81, 0x01, 9
CMInitiatorActivityTimeoutFactor	== 1 to 1000 (Base 100ms) OR Part 6: 0xFFFF	okay
CMInitiatorActivityTimeoutFactor	> 1000 (Base 100ms)	RSP-, 0xDB, 0x81, 0x01, 10
UDPCMInitiatorPort	== PN-Konstante	okay
UDPCMInitiatorPort	!= PN-Konstante	RSP-, 0xDB, 0x81, 0x01, 11
StationNameLength	== 1 to 255 AND consistent with BlockLength	okay
StationNameLength	== 0 OR > 255	RSP-, 0xDB, 0x81, 0x01, 12
CMInitiatorStationName	visible string, according to RFC 1034 AND RFC 3490	okay
CMInitiatorStationName	visible string, not according to RFC 1034 AND RFC 3490	RSP-, 0xDB, 0x81, 0x01, 13

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 50. The general structure is according to the IEC 61158-6 protocol machine model.

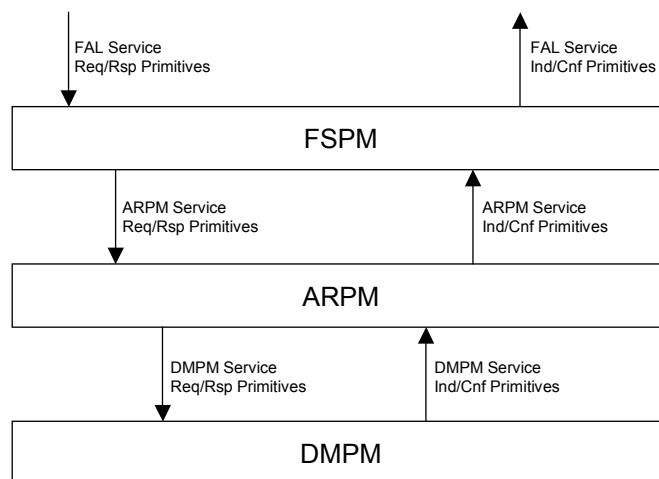


Figure 50 — Relationship among Protocol Machines

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 PROFINET IO FAL provides two kinds of protocol machine architectures, one for IO controller and one for IO devices. The architectures are specified in subclauses 5.3.2.1 and 5.3.3.

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 to protocol machines, the LMPM and the MAC protocol machines.

5.3.1.2 Fieldbus Service Protocol Machines (FSPM)

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 in this machine that allows to collaborate two AR into a single entity with higher availability.

For this task, the following support shall be offered by an FSPM:

- Redundant communication interface (RedCom) to a peer FSPM (only for redundant devices)
- Coordination of the Primary/Backup APDU_Status(information)
- Issuing switchover Procedures (signal Primary failures in case of an IO-Device)
- Collaboration of the set and get data services of the redundant AR

NOTE A switchover will be initiated in case of a communication problem (indicated by CPM via a NoData) if a switchover can be completed before Data hold time expires the system will continue otherwise the consumer will use Substitute Data for further processing.

5.3.1.3 IO controller to IO device (IO AR)

The PPM, CPM, ALPMI, ALPMR, APMR, APMS, CMDEV, CMCTL, RMPM State Machines are responsible for the cyclic, acyclic and alarm data transfer between an IO controller and an IO device.

5.3.1.4 IO supervisor to IO device (Supervisor AR)

The PPM, CPM, ALPMI, ALPMR, APMR, APMS, CMDEV, CMCTL, RMPM State Machines are responsible for the cyclic, acyclic and alarm data transfer between an IO supervisor and an IO device.

NOTE Within this application relationship, the PPM, CPM, APMR, and APMS protocol machines convey their APDUs directly over RPC and not over the LMPM.

5.3.1.5 DLL Mapping Protocol Machines (DMPM)

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 ISO/IEC 15802 – 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 a explicit confirmation for the MA_UNITDATA request service primitive indicating that the DLPDU has been transmitted.

5.3.2 Assignment of state machines to devices

Table 278 shows the assignment of State Machines to devices. Some State Machines are always required and thus marked with 'M' (mandatory) others are marked with 'O' (optional). The remarks show options in mandatory machines.

Table 278 — Assignment of state machines

Device type	Machine	mandatory/ optional	Remarks
IO device	FSPMDEV	M	Fieldbus Service Protocol Machine Device
	CMDEV	M	Context Management Device
	RMPM	M	Ressource Manager Protocol Machine
	PPM	M	Provider Protocol Machine
	CPM	M	Consumer Protocol Machine
	ALPMR	M	Alarm Protocol Machine Responder
	ALPMI	M	Alarm Protocol Machine Initiator
	APMS	M	Acyclic Protocol Machine Sender
	APMR	M	Acyclic Protocol Machine Receiver
	LMPM	M	Link Mapping Protocol Machine
IO controller	FSPMCTL	M	Fieldbus Service Protocol Machine Controller
	CMCTL	M	Context Management Controller
	NRPM	M	Name Resolution Protocol Machine
	PPM	M	Provider Protocol Machine
	CPM	M	Consumer Protocol Machine
	ALPMR	M	Alarm Protocol Machine Responder
	ALPMI	M	Alarm Protocol Machine Initiator
	APMS	M	Acyclic Protocol Machine Sender
	APMR	M	Acyclic Protocol Machine Receiver
	LMPM	M	Link Layer Mapping Protocol Machine

5.3.2.1 Overview IO controller

Figure 51 illustrates the general structure of the AL of an IO controller by showing its state machines and the services used by them.

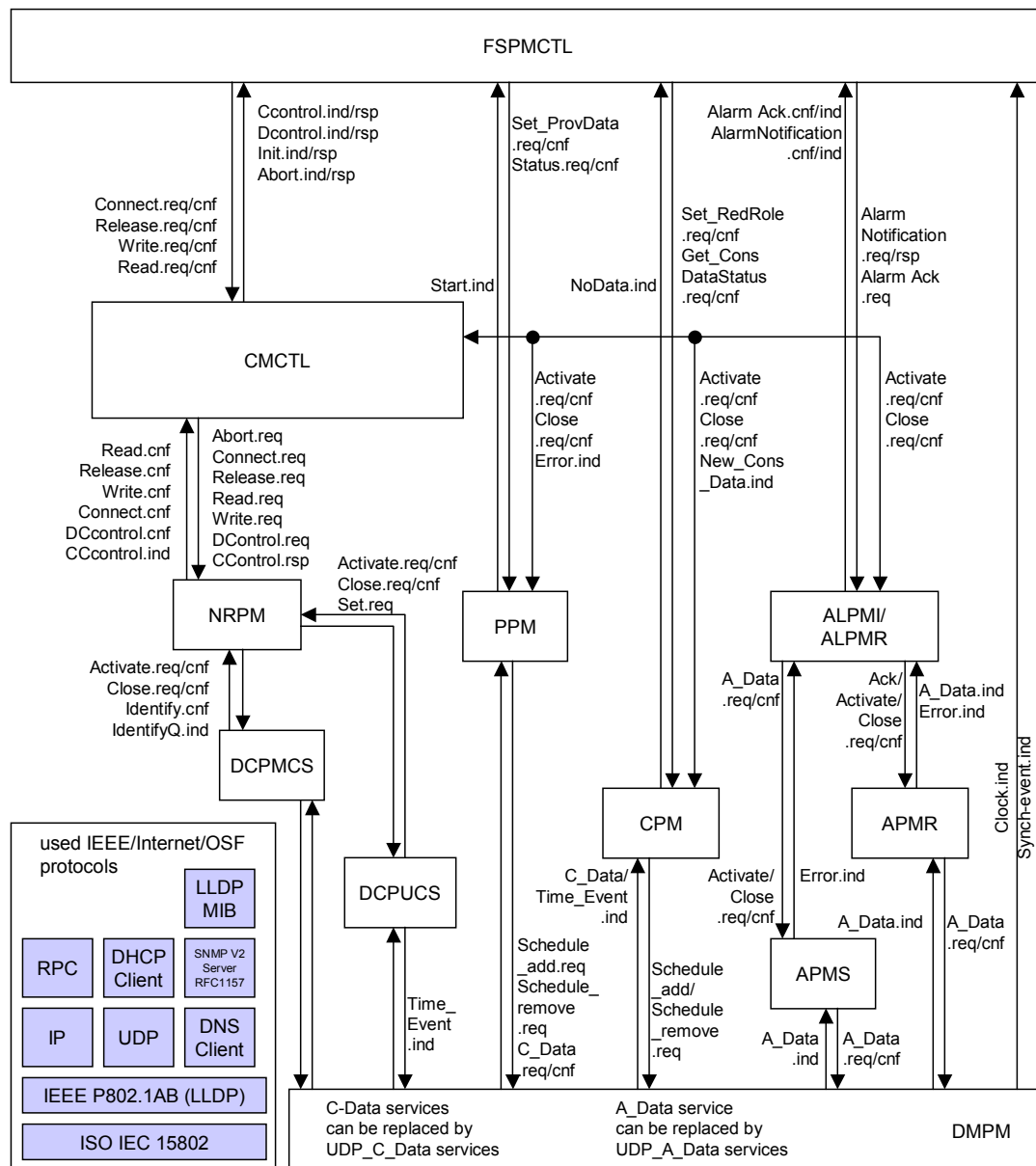


Figure 51 — Structuring of the protocol machines and adjacent layers in a IO controller

IO controllers include the following protocol machines:

CMCTL	Context Management of the IO controller (one per AR)
NRPM	Name Resolution protocol machine (one for each related IO device)
DCPUCS	Sender (Unicast) for DCP protocol (one)
DCPMCS	Sender (Multicast) for DCP protocol (one)
DCPUCR	Receiver (Unicast) for DCP protocol (one)
DCPMCR	Receiver (Multicast) for DCP protocol (one)
DHCPCL	Client for DHCP protocol (one)
DNSCL	Client for DNS protocol (one)
PPM	Provider Protocol Machine (one per IOCR)
CPM	Consumer Protocol Machine (one per IOCR)
ALPMR	Alarm Protocol Machine Responder (one or two per AR)
APMS	Acyclic Sender Protocol Machine (one or two per AR)
APMR	Acyclic Receiver Protocol Machine (one or two per AR)

RPC	Connectionless DCE RPC (one)
UDP	IETF User Datagram Protocol (one)
IP	IETF Internet Protocol (one)
LLDP	IEEE P802.1AB Link Layer Discovery Protocol (one)
LMPM	Link Mapping Protocol Machine (one)
GenSync	Generic Synchronization (Receive Control / Time Master Management)
Relay	Forwarding of frames (excluding RT class 3, synchronization frames)
IR-FW	Forwarding of RT class 3 frames
SY-FW	Forwarding of synchronization frames

5.3.3 Overview IO device

Figure 52 illustrates the general structure of the AL of an IO device by showing its state machines and the services used by them.

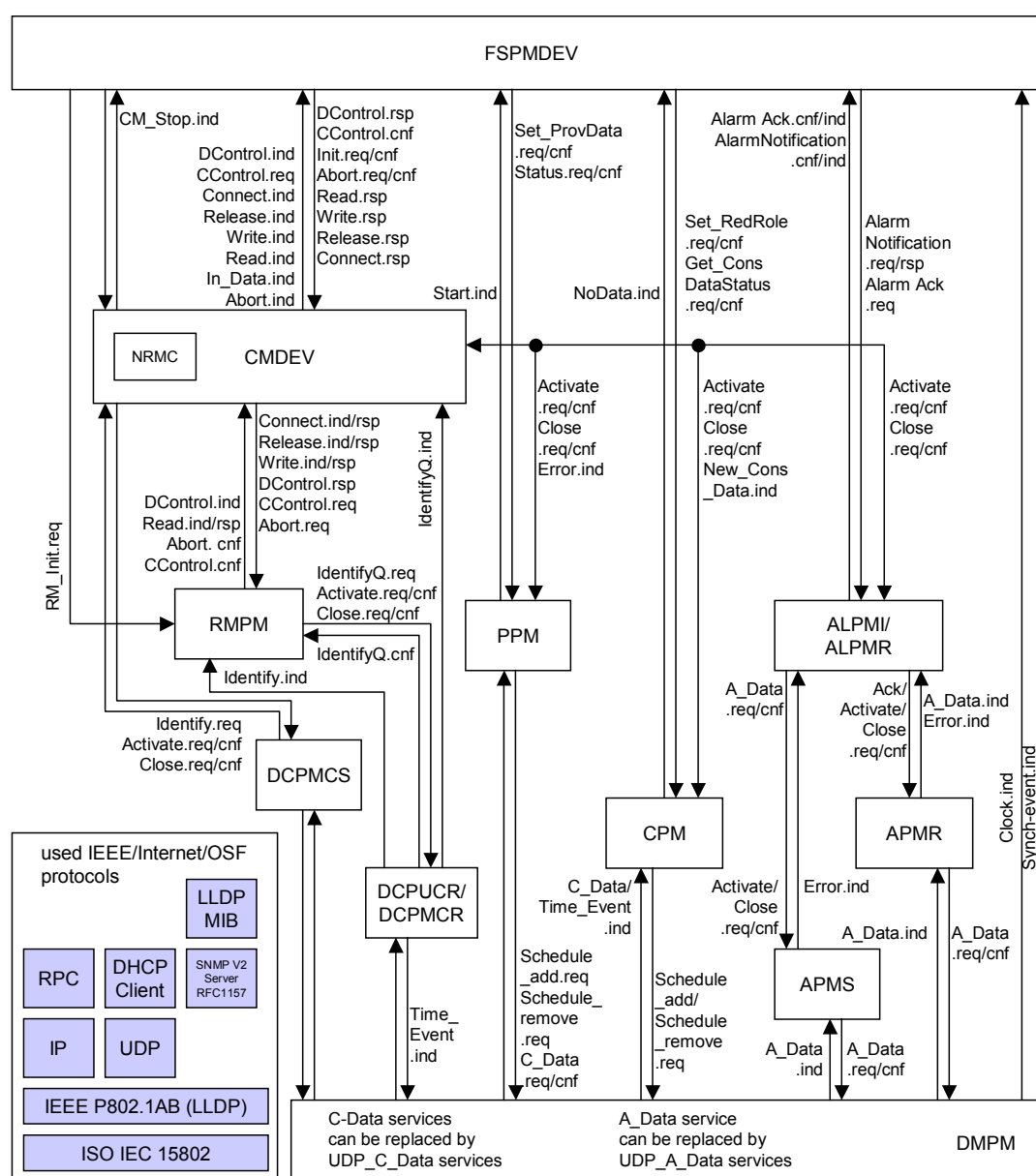


Figure 52 — Structuring of the protocol machines and adjacent layers in a IO device

The IO device include the following protocol machines:

CMDEV	Context Management of the IO device (one per AR)
RMPM	Ressource Manager protocol machine (one per device)
NRMC	Name Resolution Multicast (one per multicast consumer)
DCPUCR	Receiver (Unicast) for DCP protocol (one)
DCPMCR	Receiver (Multicast) for DCP protocol (one)
DHCPCL	Client for DHCP protocol (one)
PPM	Provider Protocol Machine (one per IOCR)
CPM	Consumer Protocol Machine (one per IOCR)
ALPMI	Alarm Protocol Machine Initiator (one or two per AR)
APMS	Acyclic Sender Protocol Machine (one or two per AR)
APMR	Acyclic Receiver Protocol Machine (one or two per AR)
RPC	Connectionless DCE RPC (one)
UDP	IETF User Datagram Protocol (one)
IP	IETF Internet Protocol (one)
LLDP	IEEE P802.1AB Link Layer Discovery Protocol (one), if released
LMPM	Link Mapping Protocol Machine (one)
GenSync	Generic Synchronization (Receive Control/ Time Master Management)
Relay	Forwarding of frames (excluding RT class 3, synchronization frames)
IR-FW	Forwarding of RT class 3 frames
SY-FW	Forwarding of synchronization frames

5.3.4 Overview IO supervisor

The IO supervisor contains in principle the same protocol machines as an IO controller. The PPM, CPM, ALPMR, APMS, and APMR protocol machines may be omitted if the IO supervisor only supports read functions. These protocol machine are necessary if the IO supervisor also supports control functions.

5.3.5 Application relationship monitoring

5.3.5.1 Monitoring of the IO – AR

5.3.5.1.1 General

In industrial control systems there is a need for checking the correct functioning of each individual part. In such a system a medium failure or a defect in an IO controller shall not cause errors in the peripheral devices. In the above mentioned cases, IO devices shall switch to a safe state. Additionally, the User has to be informed about transmission breakdowns after a certain amount of time.

5.3.5.1.2 Monitoring intervals of the Consumer

Watchdog Timer

This timer monitors the activity of the producer. The value is CR attributes SendClockFactor multiplied with ReductionRatio multiplied with the attribute Watchdog Factor. It is at least three times the sending time of the producer. It is restarted with each reception of a Data-RTC-PDU.

Data Hold Timer

This timer monitors the activity of the producer too. The last received valid data remain valid within this interval. The value should be equal to Watchdog Timer for non-redundant connection relationships. If the timer expires the data shall be set to the selected substitute value.

5.3.5.1.3 Intervals of the Producer

SendClock

This timer contains the basic send cycle.

ReductionRatio

The timer contains the actual send cycle of the IO CR.

Phase and Sequence

These attributes determine the actual sending window based on SendClockFactor and ReductionRatio and the position within the sending list.

5.3.5.2 Monitoring of services on Record CR (QQCB-CO) and Implicit CR (QQCB-CL)

The objective of the connection monitoring of DCE RPC based CRs is to detect the failure of the communication partner in case of an outstanding service. There is no explicit idle service for connection monitoring.

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 The ping service is used if the request is completely send and the response is still pending.

Timeout FRAG

The value shall be 2 s.

NOTE The transmission of a fragment will be repeated up to three times after two seconds if there is no acknowledge.

Timeout Cancel

The value shall be 1 s.

NOTE The Cancel service will be repeated up to three times after one second if there is no reaction.

Timeout Resend, Timeout Ack, Timeout Broadcast, Timeout IDLE, Timeout WAIT

The values are don't care.

NOTE The above described parameter are defined with their values in DCE RPC.

5.4 AP-Context state machine

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

NOTE The AP Context state machine is part of the IEC 61158-6 model.

5.5 FAL Service Protocol Machines (FSPMs)**5.5.1 Overview**

This type specifies one 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 FSPMDEV**5.5.2.1 Overview**

The FSPMDEV is responsible for the startup of the underlying protocol machines. They shall be started in the following order:

- 1) LMPM
- 2) LLDP
- 3) DHCP/DCP
- 4) RMPM (part of address resolution, the RMPM starts all other machines-IP, UDP, RPC, RM (other parts), CMDEV, PPM, CPM ...)

5.5.2.2 Primitive definitions**5.5.2.2.1 Primitives exchanged between FSPMDEV and AP-Context**

Table 279 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 279 — Primitives issued by AP-Context (FAL user) to FSPMDEV

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read.rsp(+)	FAL User	AREP Seq Number Add Data 1 Add Data 2 Length Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RSP+
Read.rsp(-)	FAL User	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 Input Data.rsp(+)	FAL User	AREP Seq Number Length Length Data Length IOCS Length IOPS IOCS IOPS Subslot Input Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_INPD_RSP+
Read Input Data.rsp(-)	FAL User	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_INPD_RSP-
Read Output Data.rsp(+)	FAL User	AREP Seq Number Length Length IOCS Length IOPS Length Output Data IOCS IOPS Output Data Substitute Mode Substitute Active Flag Output Substitute Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_OUTPD_RSP+
Read Output Data.rsp(-)	FAL User	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_OUTPD_RSP-
Read Logbook.rsp(+)	FAL User	AREP Seq Number Length Current Local Time Stamp Number Of Log Entries List of Entries Local Time Stamp AR UUID PNIO Status Entry Detail	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_LOG_RSP+

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read Logbook.rsp(-)	FAL User	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_LOG_RSP-
Read Device Diagnosis.rsp(+)	FAL User	AREP Seq Number Length List of Diagnosis Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_DIAG_RSP+
Read Device Diagnosis.rsp(-)	FAL User	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_DIAG_RSP-
Read Expected Identification.rsp(+)	FAL User	AREP Seq Number Length List of Slots	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_EID_RSP+
Read Expected Identification.rsp(-)	FAL User	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_EID_RSP-
Read Real Identification.rsp(+)	FAL User	AREP Seq Number Length Number of Slots List of Slots	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RID_RSP+
Read Real Identification.rsp(-)	FAL User	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_RID_RSP-
Read Identification Difference.rsp(+)	FAL User	AREP Seq Number Length Number of APIs List of APIs	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_IDDIFF_RSP+
Read Identification Difference.rsp(-)	FAL User	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_IDDIFF_RSP-

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read Real Port Data.rsp(+)	FAL User	AREP Seq Number Length Real List of Ports	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RPD_RSP+
Read Real Port Data.rsp(-)	FAL User	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_RPD_RSP-
Read Expected Port Data.rsp(+)	FAL User	AREP Seq Number Length Expected List of Ports	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_EPD_RSP+
Read Expected Port Data.rsp(-)	FAL User	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_EPD_RSP-
Read Adjusted Port Data.rsp(+)	FAL User	AREP Seq Number Length Adjusted List of Ports	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_APD_RSP+
Read Adjusted Port Data.rsp(-)	FAL User	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_APD_RSP-
Read IR Data.rsp(+)	FAL User	AREP Seq Number Length IR Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_IRD_RSP+
Read IR Data.rsp(-)	FAL User	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_IRD_RSP-
Read Real Sync Data.rsp(+)	FAL User	AREP Seq Number Length Sync Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RSYN_RSP+

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read Real Sync Data.rsp(-)	FAL User	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_RSYN_RSP-
Read Expected Sync Data.rsp(+)	FAL User	AREP Seq Number Length Sync Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_ESYN_RSP+
Read Expected Sync Data.rsp(-)	FAL User	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_ESYN_RSP-
Read PDev Data.rsp(+)	FAL User	AREP Seq Number Real List of Ports Expected List of Ports Adjusted List of Ports IR Data Sync Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_PDEV_RSP+
Read PDev Data.rsp(-)	FAL User	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_PDEV_RSP-
Read AR Data.rsp(+)	FAL User	AREP Seq Number Length Number Of ARs List of ARs	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_ARD_RSP+
Read AR Data.rsp(-)	FAL User	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_ARD_RSP-
Read IsoM Data.rsp(+)	FAL User	AREP Seq Number Length IsoM Data	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_ISOM_RSP+

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read IsoM Data.rsp(-)	FAL User	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_ISOM_RSP-
Write.rsp(+)	FAL User	AREP Multiple Seq Number Add Data 1 Add Data 2	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2)	MAP_WRITE_RSP+
Write.rsp(-)	FAL User	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_RSP-
Write Expected Port Data.rsp(+)	FAL User	AREP Multiple Seq Number	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2)	MAP_WRITE_EPD_RSP+
Write Expected Port Data.rsp(-)	FAL User	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_EPD_RSP-
Write Adjusted Port Data.rsp(+)	FAL User	AREP Multiple Seq Number	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2)	MAP_WRITE_APD_RSP+
Write Adjusted Port Data.rsp(-)	FAL User	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_APD_RSP-
Write IR Data.rsp(+)	FAL User	AREP Multiple Seq Number	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2)	MAP_WRITE_EPD_RSP+
Write IR Data.rsp(-)	FAL User	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_EPD_RSP-
Set Input.req	FAL User	AREP CREP Slot Number Subslot Number IOPS Subslot Input Data	PPM_Set_Prov_Data.req (CREP,Data)	MAP_SIN_REQ

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Set Output IOCS.req	FAL User	AREP CREP Slot Number Subslot Number IOCS	PPM_Set_Prov_Data.req (CREP,Data)	MAP_OIOCS_REQ
Set Input APDU Data Status.req	FAL User	AREP DataValid Flag ARState Flag ProviderState Flag ProblemIndicator Flag	PPM_Set_Prov_Status.req (CREP,D_Status)	MAP_SAS_REQ
Get Input IOCS.req	FAL USER	AREP CREP Slot Number Subslot Number	CPM_Get_Cons_Status.req (CREP)	MAP_GINIOCS_REQ
Get Output.req	FAL User	AREP CREP Slot Number Subslot Number	CPM_Get_Cons_Data.req (CREP), CPM_Get_Cons_Status.req (CREP)	CREP=CREP
Alarm Notification.req	FAL User	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 Ack.req	FAL User	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier	ALPMR_Alarm_Ack.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_AA_REQ
Connect.rsp(+)	FAL User	AREP AR Response Block List of IO CR Response Blocks Alarm CR Response Block Module Diff Block	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	MAP_CON_RSP+
Connect.rsp(-)	FAL User	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-
Release.rsp(+)	FAL User	AREP Session Key	CM_Release.rsp(+)(AREP, ControlBlock)	MAP_REL_RSP+
Release.rsp(-)	FAL User	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_REL_RSP-
Abort.req	FAL User	AREP	CM_Abort.req (AREP)	AREP=AREP

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
End Of Parameter.rsp(+)	FAL User	AREP Session Key Alarm Sequence Number	CM_DCControl.rsp(+) (AREP, ControlBlock)	MAP_EOP_RSP+
End Of Parameter.rsp(-)	FAL User	AREP Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2	CM_DCControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_EOP_RSP-
Application ready.req	FAL User	AREP Session Key Alarm Sequence Number Module Diff Block	CM_CCControl.req(AREP, ControlBlock, ModuleDiffBlock)	MAP_AREADY_REQ
local action			CM_Init.req	none
local action			IFW_Schedule_add.req(Port, Sched_list)	none
local action			IFW_Schedule_remove.req (Port)	none
local action			IFW_SetFWState.req(Port, FWState)	none
local action			IFW_SetLEState.req(Port, LEState)	none
local action			TMM_Set_Schedule_Item.req (Port, Sched_list)	none
local action			TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status)	none
local action			TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status)	none
local action			RCTL_Stop.req	none
local action			SYN_Adelay_Req.req (Port, DA, SA, ADQ_list, ADQ_status)	none
local action			SYN_Adelay_Req.req (Port, DA, SA, ADQ_list, ADQ_status)	none

Table 280 shows the service primitives including their associated parameters issued by the FSPMDEV and received by the AP-Context (FAL user).

Table 280 — Primitives issued by FSPMDEV to AP-Context (FAL user)

Primitive mapped from	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IND	Read.ind	FSPM	AREP API Target ARUUIID Slot Number Subslot Number Index Seq Number Length

Primitive mapped from	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_INPD_IND	Read Input Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_OUTPD_IND	Read Output Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_LOG_IND	Read Logbook.ind	FSPM	AREP Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_DIAG_IND	Read Device Diagnosis.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Diagnosis Item Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_EID_IND	Read Expected Identification.ind	FSPM	AREP Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_RID_IND	Read Real Identification.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IDDIFF_IND	Read Identification Difference.ind	FSPM	AREP Target ARUUID Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_RPD_IND	Read Real Port Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_EPD_IND	Read Expected Port Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length

Primitive mapped from	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_APD_IND	Read Adjusted Port Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IRD_IND	Read IR Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_RSYNC_IND	Read Real Sync Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_ESYNC_IND	Read Expected Sync Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_PDEV_IND	Read PDev Data.ind	FSPM	AREP Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_ARD_IND	Read AR Data.ind	FSPM	AREP Target ARUUID Seq Number Length
CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_ISOM_IND	Read IsoM Data.ind	FSPM	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length
CM_Write.ind(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_IND	Write.ind	FSPM	AREP API Slot Number Subslot Number Index Multiple Seq Number Length Data
CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	MAP_WRITE_OSUBD_IND	Write Output Substitute Data.ind	FSPM	AREP API Slot Number Subslot Number Multiple Seq Number Substitute Mode Length Output Data Output Substitute Data

Primitive mapped from	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	MAP_WRITE_EPD_IND	Write Expected Port Data.ind	FSPM	AREP API Slot Number Subslot Number Multiple Seq Number Length Expected List of Ports
CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	MAP_WRITE_APD_IND	Write Adjusted Port Data.ind	FSPM	AREP API Slot Number Subslot Number Multiple Seq Number Length Adjusted List of Ports
CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	MAP_WRITE_IRD_IND	Write IR Data.ind	FSPM	AREP API Slot Number Subslot Number Multiple Seq Number Length IR Data
CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	MAP_WRITE_ESYNC_IND	Write Sync Data.ind	FSPM	AREP API Slot Number Subslot Number Multiple Seq Number Length Sync Data
CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	MAP_WRITE_ISOM_IND	Write IsoM Data.ind	FSPM	AREP API Slot Number Subslot Number Multiple Seq Number Length IsoM Data
PPM_Set_Prov_Data.cnf(+) (CREP)	MAP_SIN_CNF	Set Input.cnf(+)	FSPM	AREP
PPM_Set_Prov_Data.cnf(-) (CREP,ERRCLS,ERRCODE)	MAP_SIN_CNF	Set Input.cnf(-)	FSPM	AREP
PPM_Set_Prov_Status.cnf(+) (CREP)	MAP_SAS_CNF	Set Input APDU Data Status.cnf(+)	FSPM	AREP
PPM_Set_Prov_Status.cnf(-) (CREP,ERRCLS,ERRCODE)	MAP_SAS_CNF	Set Input APDU Data Status.cnf(-)	FSPM	AREP
PPM_Set_Prov_Data.cnf(+) (CREP)	MAP_OIOCS_CNF	Set Output IOCS.cnf(+)	FSPM	AREP
PPM_Set_Prov_Data.cnf(-) (CREP,ERRCLS,ERRCODE)	MAP_OIOCS_CNF	Set Output IOCS.cnf(-)	FSPM	AREP
CPM_Get_Cons_Data.cnf(+) (CREP, Data, New_Flag)	MAP_GOUID_CNF+	Get Ouput.cnf(+)	FSPM	AREP IOPS Subslot Output Data New Flag IOCS
CPM_Get_Cons_Data.cnf(-) (CREP,ERRCLS,ERRCODE)	MAP_GOUID_CNF-	Get Ouput.cnf(-)	FSPM	AREP
CPM_Get_Cons_Status.cnf(+) (CREP, Status, RecvCounter)	MAP_GINIOCS_CNF+	Get Input IOCS.cnf(+)	FSPM	AREP IOCS

Primitive mapped from	Parameter mapped from	Primitive name	Source	Associated parameters
CPM_Get_Cons_Status.cnf(-) (CREP,ERRCLS,ERRCODE)	MAP_GINIOCS_CNF-	Get Input IOCS.cnf(-)	FSPM	AREP
CPM_Set_RedRole.cnf (CREP)	local action			
CM_Init.cnf	local action			
CPM_Stop.ind	local action			
CM_In_Data.ind(CREP.CPM)	MAP_INDATA_IND	New Output.ind	FSPM	AREP CREP Slot Number Subslot Number InData Flag
CM_New_Data.ind(CREP.CP M, APDU_Status)	MAP_NEWDATA_IND	New Output.ind	FSPM	AREP CREP Slot Number Subslot Number
CM_New_Data.ind(CREP.CP M, APDU_Status)	MAP_NEWSTATUS_ IND	New Output APDU Data Status.ind	FSPM	AREP CREP Data Flag AR State Flag Provider State Flag Problem Indicator Flag
CPM_NoData.ind(CREP)	MAP_NODATA_IND	New Output.ind	FSPM	AREP CREP Slot Number Subslot Number Watchdog Flag
ALPMI_Alarm_Notification.cnf (+)(CREP)	MAP_AN_CNF+	Alarm notification.cnf(+)	FSPM	AREP
ALPMI_Alarm_Notification.cnf (-)(CREP)	MAP_AN_CNF-	Alarm notification.cnf(-)	FSPM	AREP Status
ALPMI_Alarm_Ack.ind(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_AA_IND	Alarm Ack.ind	FSPM	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier
CM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmodule BlockReq)	MAP_CON_IND	Connect.ind	FSPM	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
CM_Release.ind(AREP, ControlBlock)	MAP_REL_IND	Release.ind	FSPM	AREP Session Key
CM_Abort.ind	AREP	Abort.ind	FSPM	AREP
CM_Abort.cnf(AREP)	AREP=AREP	Abort.cnf	FSPM	AREP
CM_DControl.ind(AREP, ControlBlock)	MAP_EOP_IND	End Of Parameter.ind	FSPM	AREP Session Key Alarm Sequence Number
CM_CControl.cnf(+)(AREP, ControlBlock)	MAP_AREADY_CNF+	Application Ready.cnf(+)	FSPM	AREP Session Key Alarm Sequence Number
CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_AREADY_CNF-	Application Ready.cnf(-)	FSPM	AREP Error Decode Error Code 1 Error Code 2

Primitive mapped from	Parameter mapped from	Primitive name	Source	Associated parameters
LMPM_Time_Event.ind(CREP, Cycle)	MAP_SYNCH_IND	SYNCH Event.ind	FSPM	Slot Subslot Global Cycle Counter Phase Status
RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status)	MAP_SYSTI_IND	Sync State Info.ind	FSPM	Real Sync Data Slot Number Subslot Number PTP Subdomain ID Project ID Send Clock Factor PLL Window Sync Properties Role Stratum Sync Error Status Slot Number Subslot Number
DiagnosisEvent.ind(AREP, CREP, API, Diagnosis Data)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
IFW_Error.ind(CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
IFW_Error.ind(CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	local action			
TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	local action			
LMPM_Schedule_add.cnf (+) (CREP)	local action			
LMPM_Schedule_add.cnf (-) (CREP)	local action			
IFW_Schedule_add.cnf(Port)	local action			
IFW_Schedule_remove.cnf (Port)	local action			

5.5.2.2.2 Parameters of FSPMDEV primitives

The parameters used with the primitives exchanged between the FSPMDEV and the AP-Context are described in FAL Service Definition.

5.5.2.3 State machine description

FSPMDEV State Machine has got only one possible state : Run

5.5.2.4 FSPMDEV state table

In general, the FSPMDEV defines no state machine because all services are transferred to the underlying protocol machines.

However, in case of additional multicast communication between IO device the related state machine is defined in Table 281.

Table 281 — FSPMDEV protocol machine for multicast communication

#	Current State	Event /Condition =>Action	Next State
1	run	Set Input.req(Slot Number, Subslot Number, IOPS, Subslot Input Data) /submodule with M-Provider functionality && NOT(output functionality) && SubmoduleProperties.SourceIOC == True => Data1 := Subslot Input Data, IOPS, CREP1 := CREP.PPM Data2 := Subslot Input Data, IOPS CREP2 := CREP.MPPM PPM_Set_Prov_Data.req(CREP1, Data1) PPM_Set_Prov_Data.req(CREP2, Data2)	run
2	run	Set Input.req(Slot Number, Subslot Number, IOPS, Subslot Input Data) /submodule with M-Provider functionality && NOT(output functionality) && SubmoduleProperties.SourceIOC == False => Data1 := IOPS CREP1 := CREP.PPM Data2 := Subslot Input Data, IOPS CREP2 := CREP.MPPM PPM_Set_Prov_Data.req(CREP1, Data1) PPM_Set_Prov_Data.req(CREP2, Data2)	run
3	run	Set Input.req(Slot Number, Subslot Number, IOPS, Subslot Input Data) /submodule with M-Provider functionality && output functionality && SubmoduleProperties.SourceIOC == True => Data1 := Subslot Input Data, IOPS, IOCS CREP1 := CREP.PPM Data2 := Subslot Input Data, IOPS CREP2 := CREP.MPPM PPM_Set_Prov_Data.req(CREP1, Data1) PPM_Set_Prov_Data.req(CREP2, Data2)	run
4	run	Set Input.req(Slot Number, Subslot Number, IOPS, Subslot Input Data) /submodule with M-Provider functionality && output functionality && SubmoduleProperties.SourceIOC == False => Data1 := IOCS CREP1 := CREP.PPM Data2 := Subslot Input Data, IOPS CREP2 := CREP.MPPM PPM_Set_Prov_Data.req(CREP1, Data1) PPM_Set_Prov_Data.req(CREP2, Data2)	run
5	run	Get Input IOCS.req(Slot Number, Subslot Number) /submodule with M-Provider functionality && SubmoduleProperties.SourceIOC == True => CREP := CREP.CPM CPM_Get_Cons_Data.req(CREP)	run

#	Current State	Event /Condition =>Action	Next State
6	run	Get Input IOCS.req(Slot Number, Subslot Number) /submodule with M-Provider functionality && SubmoduleProperties.SourceIOC == False => Get Input IOCS.cnf(-)	run
7	run	PPM_Set_Prov_Data.cnf(+) (CREP) /submodule with M-Provider functionality && CREP == CREP.PPM => ignore	run
8	run	PPM_Set_Prov_Data.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Provider functionality && CREP == CREP.PPM => ignore	run
9	run	PPM_Set_Prov_Data.cnf(+) (CREP) /submodule with M-Provider functionality && CREP == CREP.MPPM => Set Input.cnf(+)	run
10	run	PPM_Set_Prov_Data.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Provider functionality && CREP == CREP.MPPM => Set Input.cnf(-)	run
11	run	Set Input APDU Data Status.req(AREP, CREP, APDU data status flags) /submodule with M-Provider functionality => CREP1 := CREP.PPM CREP2 := CREP.MPPM PPM_Set_Prov_Status.req(CREP1, D_Status) PPM_Set_Prov_Status.req(CREP2, D_Status)	run
12	run	PPM_Set_Prov_Status.cnf(+) (CREP) /submodule with M-Provider functionality && CREP == CREP.PPM => ignore	run
13	run	PPM_Set_Prov_Status.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Provider functionality && CREP == CREP.PPM => ignore	run
14	run	PPM_Set_Prov_Status.cnf(+) (CREP) /submodule with M-Provider functionality && CREP == CREP.MPPM => Set Input APDU Data Status.cnf(+)	run
15	run	PPM_Set_Prov_Status.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Provider functionality && CREP == CREP.MPPM => Set Input APDU Data Status.cnf(-)	run
16	run	Get Output.req(Slot Number, Subslot Number) /submodule with M-Provider functionality => CREP := CREP.CPM CPM_Get_Cons_Data.req(CREP)	run

#	Current State	Event /Condition =>Action	Next State
17	run	CPM_Get_Cons_Data.cnf(+) (CREP, Data, New_Flag) /submodule with M-Provider functionality => Get Input IOCS.cnf(+)(IOCS, New Flag)	run
18	run	CPM_Get_Cons_Data.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Provider functionality => Get Input IOCS.cnf(-)	run
19	run	Set Output IOCS.req(Slot Number, Subslot Number, IOCS) /submodule with M-Provider functionality && output functionality => CREP := CREP.PPM PPM_Set_Prov_Data.req(CREP, IOCS) Set Output IOCS.cnf(+)	run
20	run	Set Output IOCS.req(Slot Number, Subslot Number, IOCS) /submodule with M-Provider functionality && NOT (output functionality) => Set Output IOCS.cnf(-)	run
21	run	CPM_New_Cons_Data.ind(CREP,APDU_Status) /submodule with M-Provider functionality && no change in APDU status => New Output.ind(Slot Number, Subslot Number)	run
22	run	CPM_New_Cons_Data.ind(CREP,APDU_Status) /submodule with M-Provider functionality && change in APDU status => New Output.ind(Slot Number, Subslot Number) New Output APDU Data Status.ind(AREP, CREP, APDU status flags)	run
23	run	CPM_No_Data.ind(CREP) /submodule with M-Provider functionality => Watchdog Flag := WATCHDOG_EXPIRED New Output.ind(Slot Number, Subslot Number, Watchdog Flag)	run
24	run	CPM_Start.ind(CREP) /submodule with M-Provider functionality => InData Flag := INDATA New Output.ind(Slot Number, Subslot Number, InData Flag) New Output APDU Data Status.ind(AREP, CREP, APDU status flags)	run
25	run	CPM_New_Cons_Data.ind(CREP,APDU_Status) /submodule with M-Consumer functionality && CREP == CREP.CPM => ignore	run
26	run	CPM_New_Cons_Data.ind(CREP,APDU_Status) /submodule with M-Consumer functionality && CREP == CREP.MCPM && no change in APDU status => New Output.ind(Slot Number, Subslot Number)	run
27	run	CPM_New_Cons_Data.ind(CREP,APDU_Status) /submodule with M-Consumer functionality && CREP == CREP.MCPM && change in APDU status => New Output.ind(Slot Number, Subslot Number) New Output APDU Data Status.ind(AREP, CREP, APDU status flags)	run

#	Current State	Event /Condition =>Action	Next State
28	run	Get Output.req(Slot Number, Subslot Number) /submodule with M-Consumer functionality => CREP := CREP.MCPM CPM_Get_Cons_Data.req(CREP)	run
29	run	CPM_Get_Cons_Data.cnf(+) (CREP, Data, New_Flag) /submodule with M-Consumer functionality && CREP := MCPM => Get Output.cnf(+)(IOPS, Subslot Output Data, New Flag)	run
30	run	CPM_Get_Cons_Data.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Consumer functionality => Get Output.cnf(-)	run
31	run	CPM_Get_Cons_Data.cnf(+) (CREP, Data, New_Flag) /submodule with M-Consumer functionality && input functionality && CREP := CPM => Get Output.cnf(+)(New Flag, IOCS)	run
32	run	Set Output IOCS.req(Slot Number, Subslot Number, IOCS) /submodule with M-Consumer functionality => Set Output IOCS.cnf(-)	run
33	run	Get Input IOCS.req(Slot Number, Subslot Number) /submodule with M-Consumer functionality && input functionality => CREP := CREP.CPM CPM_Get_Cons_Data.req(CREP)	run
34	run	Get Input IOCS.req(Slot Number, Subslot Number) /submodule with M-Consumer functionality && NOT (input functionality) => Get Input IOCS.cnf(-)	run
35	run	Set Input.req(Slot Number, Subslot Number, IOPS, Subslot Input Data) /submodule with M-Consumer functionality && input functionality => Data := Subslot Input Data, IOPS CREP := CREP.PPM PPM_Set_Prov_Data.req(CREP, Data)	run
36	run	Set Input.req(Slot Number, Subslot Number, IOPS, Subslot Input Data) /submodule with M-Consumer functionality && NOT (input functionality) => Data := IOPS CREP := CREP.PPM PPM_Set_Prov_Data.req(CREP, Data)	run
37	run	Set Input APDU Data Status.req(AREP, CREP, APDU data status flags) /submodule with M-Consumer functionality => CREP := CREP.PPM D_Status := APDU data status flag PPM_Set_Prov_Status.req(CREP, D_Status)	run
38	run	PPM_Set_Prov_Status.cnf(+) (CREP) /submodule with M-Consumer functionality => Set Input APDU Data Status.req(+)	run
39	run	PPM_Set_Prov_Status.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Consumer functionality => Set Input APDU Data Status.req(-)	run

#	Current State	Event /Condition =>Action	Next State
40	run	PPM_Set_Prov_Data.cnf(+) (CREP) /submodule with M-Consumer functionality => Set Input.cnf(+)	run
41	run	PPM_Set_Prov_Data.cnf(-) (CREP, ERRCLS, ERRCODE) /submodule with M-Consumer functionality => Set Input.cnf(-)	run
42	run	CPM_No_Data.ind(CREP) /submodule with M-Consumer functionality && CREP == CREP.CPM => ignore	run
43	run	CPM_No_Data.ind(CREP) /submodule with M-Consumer functionality && CREP == CREP.MCPM => Watchdog Flag := WATCHDOG_EXPIRED New Output.ind(Slot Number, Subslot Number, Watchdog Flag)	run
44	run	CPM_Start.ind(CREP) /submodule with M-Consumer functionality && CREP == CREP.CPM => ignore	run
45	run	CPM_Start.ind(CREP) /submodule with M-Consumer functionality && CREP == CREP.MCPM => InData Flag := INDATA New Output.ind(Slot Number, Subslot Number, In Data Flag)	run
46	run	CM_Stop.ind(CREP) /submodule with M-Consumer functionality && CREP == CREP.MCPM => For all submodules of M-Consumer-CR do: M-Provider Communication Stopped.ind(Slot Number, Subslot Number)	run

5.5.2.5 Functions

Table 282 shows the functions defined for FSPMDEV, which are used by service primitives issued by the FAL user.

Table 282 — Functions used by AP-Context (FAL user) to FSPMDEV

Name	Function
MAP_READ_RSP+	map service parameter 1:1 (=)
MAP_READ_RSP-	map service parameter 1:1 (=)
MAP_READ_INPD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Length Data, Length IOCS, Length IOPS, IOCS, IOPS, Subslot Input Data}
MAP_READ_INPD_RSP-	map service parameter 1:1 (=)

Name	Function
MAP_READ_OUTPD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Length IOCS, Length IOPS, Length Output Data, IOCS, IOPS, Output Data, Substitute Mode, Substitute Active Flag, Output Substitute Data}
MAP_READ_OUTPD_RSP-	map service parameter 1:1 (=)
MAP_READ_LOG_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Current Local Time Stamp, Number of Log Entries, [List of Entries (Local Time Stamp, AR UUID, PNIO Status, Entry Detail)]*}
MAP_READ_LOG_RSP-	map service parameter 1:1 (=)
MAP_READ_DIAG_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Diagnosis Item=List of Diagnosis Data
MAP_READ_DIAG_RSP-	map service parameter 1:1 (=)
MAP_READ_EID_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={List of Slots}
MAP_READ_EID_RSP-	map service parameter 1:1 (=)
MAP_READ_RID_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Number of Slots, List of Slots}
MAP_READ_RID_RSP-	map service parameter 1:1 (=)
MAP_READ_IDDIFF_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Number of APIs, List of APIs}
MAP_READ_IDDIFF_RSP-	map service parameter 1:1 (=)
MAP_READ_RPD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Real List of Ports}
MAP_READ_RPD_RSP-	map service parameter 1:1 (=)

Name	Function
MAP_READ_EPD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Expected List of Ports}
MAP_READ_EPD_RSP-	map service parameter 1:1 (=)
MAP_READ_APD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Adjusted List of Ports}
MAP_READ_APD_RSP-	map service parameter 1:1 (=)
MAP_READ_IRD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={{IR Data}}
MAP_READ_IRD_RSP-	map service parameter 1:1 (=)
MAP_READ_RSYNC_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Sync Data}
MAP_READ_RSYNC_RSP-	map service parameter 1:1 (=)
MAP_READ_ESYNC_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Sync Data}
MAP_READ_ESYNC_RSP-	map service parameter 1:1 (=)
MAP_READ_PDEV_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data= {Real Port Data, Expected Port Data, IR Data, Real Sync Data, Expected Sync Data}
MAP_READ_PDEV_RSP-	map service parameter 1:1 (=)
MAP_READ_ARD_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={Number of ARs, List of ARs}
MAP_READ_ARD_RSP-	map service parameter 1:1 (=)
MAP_READ_ISOM_RSP+	AREP=AREP Seq Number=Seq Number Add Data 1=0 Add Data 2=0 Length=Length Data={IsoM Data}

Name	Function
MAP_READ_ISOM_RSP-	map service parameter 1:1 (=)
MAP_WRITE_RSP+	map service parameter 1:1 (=)
MAP_WRITE_RSP-	map service parameter 1:1 (=)
MAP_WRITE_EPD_RSP+	map FAL User service parameter 1:1 (=) set AddData1=0 set AddData2=0
MAP_WRITE_EPD_RSP-	map service parameter 1:1 (=)
MAP_WRITE_APD_RSP+	map FAL User service parameter 1:1 (=) set AddData1=0 set AddData2=0
MAP_WRITE_APD_RSP-	map service parameter 1:1 (=)
MAP_WRITE_ISOM_RSP+	map FAL User service parameter 1:1 (=) set AddData1=0 set AddData2=0
MAP_WRITE_ISOM_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
MAP_CON_RSP-	map service parameter 1:1 (=)
MAP_REL_RSP+	AREP=AREP ReleaseBlock.SessionKey=SessionKey
MAP_REL_RSP-	map service parameter 1:1 (=)
MAP_AREADY_REQ	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number) ModuleDiffBlock=Module Diff Block
MAP_EOP_RSP+	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number)
MAP_EOP_RSP-	map service parameter 1:1 (=)
MAP_SIN_REQ	CREP=CREP Data={IOPS, Subslot Input Data}
MAP_GINIOCS_REQ	CREP=CREP
MAP_SAS_REQ	CREP=AREP.CREP D_Status={DataValid Flag, AR State Flag, ProviderState Flag, ProblemIndicator Flag}
MAP_OIOCS_REQ	CREP=CREP Data=IOCS
MAP_AN_REQ	map service parameter 1:1 (=)

Table 283 shows the functions defined for FSPMDEV, which are used by services primitives issued by the FSPMDEV.

Table 283 — Function used by FSPMDEV to AP-Context (FAL user)

Name	Function
MAP_READ_IND	if ((Index < =0x7FFF) OR (0xAFF0<=Index<=0xAFFF)) then {AREP=AREP API=API Target ARUUID = TargetUUID Slot Number = SlotNumber Subslot Number =SubslotNumber Index =Index Seq Number =SeqNumber Length=Length} else map to special FAL user Read service primitive
MAP_READ_INPD_IND	if (Index =0x8028) then {AREP=AREP API=API Target ARUUID = TargetUUID Slot Number = SlotNumber Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_OUTPD_IND	if (Index =0x8029) then {AREP=AREP API=API Target ARUUID = TargetUUID Slot Number = SlotNumber Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_LOG_IND	if (Index =0xF830) then {AREP=AREP Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_DIAG_IND	if ((Index =0x800A)OR(Index =0x800B)OR(Index =0x800C)OR (Index =0xC00A)OR(Index =0xC00B)OR(Index =0xC00C)OR (Index =0xE00A)OR(Index =0xE00B)OR(Index =0xE00C)OR (Index =0xF00A)OR(Index =0xF00B)OR(Index =0xF00C))then {AREP=AREP if (TargetUUID<>NULL) then Target ARUUID=TargetUUID else Target ARUUID ="IGNORE" if (Index =0xE00A)OR(Index =0xE00B)OR(Index =0xE00C) then API = "IGNORE else API=API if ((Index =0xF00A)OR(Index =0xF00B)OR(Index =0xF00C) OR (Index =0xE00A)OR(Index =0xE00B)OR(Index =0xE00C)) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xF00A)OR(Index =0xF00B)OR(Index =0xF00C) OR (Index =0xC00A)OR(Index =0xC00B)OR(Index =0xC00C)OR (Index =0xE00A)OR(Index =0xE00B)OR(Index =0xE00C)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber if ((Index =0x800A)OR(Index =0xC00A) OR(Index =0xE00A)OR(Index =0xF00A) then Diagnosis Item = CHANNEL if ((Index =0x800B)OR(Index =0xC00B) OR(Index =0xE00B)OR(Index =0xE00B)OR(Index =0xF00B) then Diagnosis Item = MANUFACTURER_ERROR_APPEARS if ((Index =0x800C)OR(Index =0xC00C) OR(Index =0xE00C)OR(Index =0xF00C) then Diagnosis Item = MANUFACTURER_ALL Length=Length} else map to other special FAL user Read service primitive

Name	Function
MAP_READ_EID_IND	if ((Index =0x8000)OR(Index =0xC000)OR(Index =0xE000)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE000) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE000)OR(Index =0xC000)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_RID_IND	if ((Index =0x8001)OR(Index =0xC001)OR(Index =0xE001)OR(Index =0xF000)) then {AREP=AREP API=API if (Index =0xF001) then Target ARUUID = "IGNORE" else Target ARUUID = TargetUUID if ((Index =0xE001)OR(Index =0xF001)) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE001)OR(Index =0xC001)OR(Index =0xF000)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_IDDIFF_IND	if (Index =0xE002) then {AREP=AREP Target ARUUID = TargetUUID Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_RPD_IND	if ((Index =0x802A)OR(Index =0xC02A)OR(Index =0xE02A)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE02A) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE02A)OR(Index =0xC02A)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_EPD_IND	if ((Index =0x802B)OR(Index =0xC02B)OR(Index =0xE02B)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE02B) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE02B)OR(Index =0xC02B)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_APD_IND	if ((Index =0x802F)OR(Index =0xC02F)OR(Index =0xE02F)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE02F) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE02F)OR(Index =0xC02F)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive

Name	Function
MAP_READ_IRD_IND	if ((Index =0x802C)OR(Index =0xC02C)OR(Index =0xE02C)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE02C) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE02C)OR(Index =0xC02C)) then Subslot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_RSYNC_IND	if ((Index =0x802D)OR(Index =0xC02D)OR(Index =0xE02D)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE02D) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE02D)OR(Index =0xC02D)) then Subslot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_ESYNC_IND	if ((Index =0x802E)OR(Index =0xC02E)OR(Index =0xE02E)) then {AREP=AREP Target ARUUID = TargetUUID if (Index =0xE02E) then Slot Number = "IGNORE" else Slot Number =SlotNumber if ((Index =0xE02E)OR(Index =0xC02E)) then Subslot Number = "IGNORE" else Subslot Number =SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_PDEV_IND	if (Index =0xF831) then {AREP=AREP Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_ARD_IND	if ((Index =0x8001)OR(Index =0xC001)OR(Index =0xF020)OR(Index =0xF820)) then {AREP=AREP if (Index =0xF820) then API = "IGNORE" else API=API if ((Index =0xF820)OR(Index =0xF020)) then Target ARUUID = "IGNORE" else Target ARUUID = TargetUUID Slot Number = "IGNORE" Sublot Number = "IGNORE" Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive
MAP_READ_ISOM_IND	if (Index =0x8030) then {AREP=AREP API=API Target ARUUID = TargetUUID Slot Number = SlotNumber Sublot Number = SubslotNumber Seq Number =SeqNumber Length=Length} else map to other special FAL user Read service primitive

Name	Function
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
MAP_WRITE_OSUBD_IND	if ((Index =0x801E)) then {AREP=AREP API=API Slot Number = SlotNumber Subslot Number =SubslotNumber Index =Index Multiple = Multiple Seq Number =SeqNumber Length=Length {Substitute Mode, Length Output Data, Output Substitute Data} = Data}
MAP_WRITE_EPD_IND	if ((Index =0x802B)OR(Index =0xC02B)OR(Index =0xE02B)) then {AREP=AREP API=API if (Index =0xE02B) then Slot Number = "IGNORE" else Slot Number = SlotNumber if ((Index =0xE02B)OR(Index =0xC02B)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Index =Index Multiple = Multiple Seq Number =SeqNumber Length=Length {Expected List of Ports} = Data} else map to special FAL user Write service primitive
MAP_WRITE_APD_IND	if ((Index =0x802F)OR(Index =0xC02F)OR(Index =0xE02F)) then {AREP=AREP API=API if (Index =0xE02F) then Slot Number = "IGNORE" else Slot Number = SlotNumber if ((Index =0xE02F)OR(Index =0xC02F)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Index =Index Multiple = Multiple Seq Number =SeqNumber Length=Length {Adjusted List of Ports} = Data} else map to special FAL user Write service primitive
MAP_WRITE_IRD_IND	if ((Index =0x802C)OR(Index =0xC02C)OR(Index =0xE02C)) then {AREP=AREP API=API if (Index =0xE02C) then Slot Number = "IGNORE" else Slot Number = SlotNumber if ((Index =0xE02C)OR(Index =0xC02C)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Multiple = Multiple Seq Number =SeqNumber Length=Length {IR Data} = Data} else map to special FAL user Write service primitive

Name	Function
MAP_WRITE_ESYNC_IND	if ((Index =0x802E)OR(Index =0xC02E)OR(Index =0xE02E)) then {AREP=AREP API=API if (Index =0xE02E) then Slot Number = "IGNORE" else Slot Number = SlotNumber if ((Index =0xE02E)OR(Index =0xC02E)) then Sublot Number = "IGNORE" else Subslot Number =SubslotNumber Multiple = Multiple Seq Number =SeqNumber Length=Length {Sync Data} = Data} else map to special FAL user Write service primitive
MAP_WRITE_ISOM_IND	if (Index =0x8030) then {AREP=AREP API=API Slot Number = SlotNumber Subslot Number =SubslotNumber Multiple = Multiple Seq Number =SeqNumber Length=Length IsoM Data = Data} else map to special FAL user Write service primitive
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 Multicast CR Block
MAP_REL_IND	AREP=AREP SessionKey=ReleaseBlock.SessionKey
MAP_AREADY_CNF+	AREP=AREP Session Key=ControlBlockPlug.SessionKey^ControlConnectPlug.SessionKey
MAP_AREADY_CNF-	map service parameter 1:1 (=)
MAP_EOP_IND	AREP=AREP Session Key= ControlBlockConnect.SessionKey^ControlBlockPlug.SessionKey Alarm Sequence Number=ControlBlockPlug.AlarmSequenceNumber
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 AREP=CREP.AREP CREP=CREP Slot Number Subslot Number
MAP_NEWSTATUS_IND	if APDUStatus has been changed AREP=CREP.AREP CREP=CREP Data Flag =APDU_Status AR State Flag =APDU_Status Provider State Flag =APDU_Status Problem Indicator Flag =APDU_Status
MAP_NODATA_IND	AREP=CREP.AREP CREP=CREP Slot Number =0 Subslot Number =0 Watchdog Flag=TRUE
MAP_SIN_CNF	AREP=CREP.AREP other ignore
MAP_SAS_CNF	AREP=CREP.AREP other ignore

Name	Function
MAP_OIOCS_CNF	AREP=CREP.AREP other ignore
MAP_GOUTD_CNF+	AREP=CREP IOPS=Data Subslot Output Data=Data New Flag=New_Flag IOCS=Data
MAP_GOUTD_CNF-	AREP=CREP.AREP other ignore
MAP_AN_CNF+	AREP=CREP.AREP
MAP_AN_CNF-	AREP=CREP.AREP Status
MAP_AA_IND	AREP =CREP.AREP API Alarm Type =Alarm_Type Slot Number=Slot_Number Subslot Number=Subslot_Number Alarm Specifier=Alarm_Specifier
MAP_SYNCH_IND	Global Cycle Counter =Cycle Status
MAP_DIAGEVT_IND	AREP =AREP CREP=CREP Alarm Item = Diagnosis Data

5.5.3 FSPMCTL

5.5.3.1 Primitive definitions

5.5.3.1.1 Primitives exchanged between FSPMCTL and AP-Context (FAL user)

Table 284 shows the service primitives including their associated parameters issued by the AP-Context (FAL user) and received by the FSPMCTL.

Table 284 — Primitives issued by AP-Context (FAL user) to FSPMCTL

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Index Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_REQ
Read Input Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_INPD_REQ
Read Output Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_OUTPD_REQ
Read Logbook.req	FAL USER	AREP Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_LOG_REQ

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read Device Diagnosis.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Diagnosis Item Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_DIAG_REQ
Read Expected Identification.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_EID_REQ
Read Real Identification.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_RID_REQ
Read Identification Difference.req	FAL USER	AREP Target ARUUID Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IDDIFF_REQ
Read Real Port Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_RPD_REQ
Read Expected Port Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_EPD_REQ
Read Adjusted Port Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_APD_REQ
Read IR Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_IRD_REQ
Read Real Sync Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_RSYNC_REQ

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Read Expected Sync Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_ESYNC_REQ
Read PDev Data.req	FAL USER	AREP Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_PDEV_REQ
Read AR Data.req	FAL USER	AREP Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_ARD_REQ
Read IsoM Data.req	FAL USER	AREP API Target ARUUID Slot Number Subslot Number Seq Number Length	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	MAP_READ_ISOM_REQ
Write.req	FAL USER	AREP API Slot Number Subslot Number Index Multiple Seq Number Length Data	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_REQ
Write Expected Port Data.req	FAL USER	AREP API Slot Number Subslot Number Multiple Seq Number Length Expected List of Ports	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_EPD_REQ
Write Adjusted Port Data.req	FAL USER	AREP API Slot Number Subslot Number Multiple Seq Number Length Adjusted List of Ports	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_EPD_REQ
Write IR Data.req	FAL USER	AREP API Slot Number Subslot Number Multiple Seq Number Length IR Data	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_IRD_REQ
Write Sync Data.req	FAL USER	AREP API Slot Number Subslot Number Multiple Seq Number Length Sync Data	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_ESYNC_REQ

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Write IsoM Data.req	FAL USER	AREP API Slot Number Subslot Number Multiple Seq Number Length IsoM Data	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	MAP_WRITE_ESYNC_REQ
Set Input IOCS.req	FAL USER	AREP CREP Slot Number Subslot Number IOCS	PPM_Set_Prov_Data.req (CREP,Data)	MAP_SINIOCS_REQ
Get Input.req	FAL USER	AREP CREP Slot Number Subslot Number	CPM_Get_Cons_Data.req (CREP)	MAP_GIN_REQ
Set Output.req	FAL USER	AREP CREP Slot Number Subslot Number IOPS Subslot Output Data	PPM_Set_Prov_Data.req (CREP,Data)	MAP_SOUT_REQ
Set Output APDU Data Status.req	FAL USER	AREP Data Flag AR State Flag Provider State Flag Problem Indicator Flag	PPM_Set_Prov_Status.req (CREP,D_Status)	MAP_SAS_REQ
Get Output IOCS.req	FAL USER	AREP CREP Slot Number Subslot Number	CPM_Get_Cons_Status.req (CREP)	MAP_GOUTIOCS_REQ
Connect.req	FAL USER	AREP AR Parameter Block List of IO CR Parameter Blocks List of Expected Submodule Blocks Alarm CR Parameter Block Parameter Server Block Multicast CR Block	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq)	MAP_CON_REQ
Release.req	FAL USER	AREP Session Key	CM_Release.req(AREP, ControlBlock)	MAP_REL_REQ
End Of Parameter.req	FAL USER	AREP Session Key Alarm Sequence Number	CM_DControl.req(AREP, ControlBlock)	MAP_EOP_REQ
Application Ready.rsp(+)	FAL USER	AREP Session Key Alarm Sequence Number	CM_CControl.rsp(+) (AREP, ControlBlock)	MAP_AREADY_RES+
Application Ready.rsp(-)	FAL USER	AREP Error Decode Error Code 1 Error Code 2	CM_CControl.rsp(-) (AREP, ControlBlock)	MAP_AREADY_RES-
Alarm Notification.req	FAL USER	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

Primitive name	Source	Associated parameters	Primitive mapped to	Parameter mapped to
Alarm Ack.req	FAL USER	AREP API Alarm Type Slot Number Subslot Number Alarm Specifier	ALPMR_Alarm_Ack.req (CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	MAP_AACK_REQ
local action			CM_CTL_Init.req(AREP)	
local action			IFW_Schedule_add.req (Port, Sched_list)	
local action			IFW_Schedule_remove. req(Port)	
local action			IFW_SetFWState.req (Port, FWState)	
local action			IFW_SetLEState.req (Port, LEState)	
local action			TMM_Set_Schedule_ Item.req(Port,Sched_list)	
local action			TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status)	
local action			TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status)	
local action			RCTL_Stop.req ()	
local action			SYN_Adelay_Req.req (Port, DA, SA, ADQ_list, ADQ_status)	
local action			SYN_Adelay_Req.req (Port, DA, SA, ADQ_list, ADQ_status)	

Table 285 shows the service primitives including their associated parameters issued by the FSPMCTL and received by the AP-Context (FAL user).

Table 285 — Primitives issued by FSPMCTL to AP-Context (FAL user)

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_CNF+	Read.cnf(+)	FSPM	AREP Seq Number Add Data 1 Add Data 2 Length Data
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_CNF-	Read.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_INPD_CNF+	Read Input Data.cnf(+)	FSPM	AREP Seq Number Length Length Data Length IOCS Length IOPS IOCS IOPS Subslot Input Data

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_INPD_CNF-	Read Input Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_OUTPD_CNF+	Read Output Data.cnf(+)	FSPM	AREP Seq Number Length Length Output Data Length IOCS Length IOPS IOCS IOPS Output Data Substitute Mode Substitute Active Flag Output Substitute Data
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_OUTPD_CNF-	Read Output Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_LOG_CNF+	Read Logbook.cnf(+)	FSPM	AREP Seq Number Length Current Local Time Stamp Number of Log Entries List of Entries Local Time Stamp AR UUID PNIO Status Entry Detail
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_LOG_CNF-	Read Logbook.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_DIAG_CNF+	Read Device Diagnosis.cnf(+)	FSPM	AREP Seq Number Length List of Diagnosis Data
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_DIAG_CNF-	Read Device Diagnosis.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_EID_CNF+	Read Expected Identification.cnf(+)	FSPM	AREP Seq Number Length List of Slots
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_EID_CNF-	Read Expected Identification.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RID_CNF+	Read Real Identification.cnf(+)	FSPM	AREP Seq Number Length List of Slots
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_RID_CNF-	Read Real Identification.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_IDDIFF_CNF+	Read Identification Difference.cnf(+)	FSPM	AREP Seq Number Length List of APIs
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_IDDIFF_CNF-	Read Identification Difference.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RPD_CNF+	Read Real Port Data.cnf(+)	FSPM	AREP Seq Number Length Real List of Ports
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_RPD_CNF-	Read Real Port Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_EPD_CNF+	Read Expected Port Data.cnf(+)	FSPM	AREP Seq Number Length Expected List of Elements
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_EPD_CNF-	Read Expected Port Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_APD_CNF+	Read Adjusted Port Data.cnf(+)	FSPM	AREP Seq Number Length Adjusted List of Elements
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_APD_CNF-	Read Adjusted Port Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_IRD_CNF+	Read IR Data.cnf(+)	FSPM	AREP Seq Number Length IR Data
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_IRD_CNF-	Read IR Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_RSINC_CNFP	Read Real Sync Data.cnf(+)	FSPM	AREP Seq Number Length Number of Sync Interfaces List of Sync Interfaces Slot Number Subslot Number PTP Subdomain ID Project ID Send Clock Factor PLL Window Sync Properties Role Stratum
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_RSINC_CNFM	Read Real Sync Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_ESINC_CNFP	Read Expected Sync Data.cnf(+)	FSPM	AREP Seq Number Length Number of Sync Interfaces List of Sync Interfaces Slot Number Subslot Number PTP Subdomain ID Project ID Send Clock Factor PLL Window Sync Properties Role Stratum
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_ESINC_CNFM	Read Expected Sync Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_PDEV_CNFP	Read PDev Data.cnf(+)	FSPM	AREP Seq Number Length Real Port Data IR Data Real Sync Data
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_PDEV_CNFM	Read PDev Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_ARD_CNFP	Read AR Data.cnf(+)	FSPM	AREP Seq Number Length Number Of ARs List of ARs
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_ARD_CNFM	Read AR Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	MAP_READ_ISOM_CNF+	Read IsoM Data.cnf(+)	FSPM	AREP Seq Number Length IsoM Data
CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_READ_ISOM_CNF-	Read IsoM Data.cnf(-)	FSPM	AREP Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_CNF+	Write.cnf(+)	FSPM	AREP Multiple Seq Number Add Data 1 Add Data 2
CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_CNF-	Write.cnf(-)	FSPM	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_EPD_CNF+	Write Expected Port Data.cnf(+)	FSPM	AREP Multiple Seq Number
CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_EPD_CNF-	Write Expected Port Data.cnf(-)	FSPM	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_APD_CNF+	Write Adjusted Port Data.cnf(+)	FSPM	AREP Multiple Seq Number
CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_APD_CNF-	Write Adjusted Port Data.cnf(-)	FSPM	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_IRD_CNF+	Write IR Data.cnf(+)	FSPM	AREP Multiple Seq Number
CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_IRD_CNF-	Write IR Data.cnf(-)	FSPM	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	MAP_WRITE_ESYNC_CNF+	Write Sync Data.cnf(+)	FSPM	AREP Multiple Seq Number

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_WRITE_ESYNC_CNF-	Write Sync Data.cnf(-)	FSPM	AREP Multiple Seq Number Error Decode Error Code 1 Error Code 2 Add Data 1 Add Data 2
CM_InData.ind(AREP)	MAP_INDATA_IND	New Input.ind	FSPM	AREP CREP Slot Number Subslot Number Watchdog Flag InData Flag
CM_New_Input_Data.ind(AREP)	MAP_NEWDATA_IND	New Input.ind	FSPM	AREP CREP Slot Number Subslot Number Watchdog Flag InData Flag
CM_New_Input_Data.ind(AREP)	MAP_NEW_STATUS_IND	New Input APDU Data Status.ind	FSPM	AREP CREP DataValid Flag ARState Flag ProviderState Flag ProblemIndicator Flag
CM_Stop.ind	local action			
CPM_NoData.ind(CREP)	MAP_NODATA_IND	New Input.ind	FSPM	AREP CREP Slot Number Subslot Number Watchdog Flag InData Flag
CPM_Get_Cons_Data.cnf(+)(CREP, Data, New_Flag)	MAP_GIND_CNF+	Get Input.cnf(+)	FSPM	AREP IOPS Subslot Input Data New Flag IOCS
CPM_Get_Cons_Data.cnf(-)(CREP,ERRCLS,ERRCODE)	MAP_GIND_CNF-	Get Input.cnf(-)	FSPM	AREP
CPM_Get_Cons_Status.cnf(+)(CREP, Status, RecvCounter)	MAP_GOUTIO_CS_CNF+	Get Output IOCS.cnf(+)	FSPM	AREP IOCS
CPM_Get_Cons_Status.cnf(-)(CREP,ERRCLS,ERRCODE)	MAP_GOUTIO_CS_CNF-	Get Output IOCS.cnf(-)	FSPM	AREP
PPM_Set_Prov_Data.cnf(+)(CREP)	MAP_SIN_CNF	Set Input.cnf(+)	FSPM	AREP
PPM_Set_Prov_Data.cnf(-)(CREP,ERRCLS,ERRCODE)	MAP_SIN_CNF	Set Input.cnf(-)	FSPM	AREP
PPM_Set_Prov_Data.cnf(+)(CREP)	MAP_IIOCS_CNF	Set Input IOCS.cnf(+)	FSPM	AREP
PPM_Set_Prov_Data.cnf(-)(CREP,ERRCLS,ERRCODE)	MAP_IIOCS_CNF	Set Input IOCS.cnf(-)	FSPM	AREP
PPM_Set_Prov_Status.cnf(+)(CREP)	MAP_SAS_CNF	Set Input APDU Data Status.cnf(+)	FSPM	AREP
PPM_Set_Prov_Status.cnf(-)(CREP,ERRCLS,ERRCODE)	MAP_SAS_CNF	Set Input APDU Data Status.cnf(-)	FSPM	AREP

Primitive mapped to	Parameter mapped from	Primitive name	Source	Associated parameters
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 Notification.ind	FSPM	AREP API Alarm Priority Alarm Type Slot Number Subslot Number Alarm Specifier Module Ident Number Submodule Ident Number Alarm Item
ALPMR_Alarm_Ack.cnf(+)		Alarm Ack.cnf(+)	FSPM	AREP
ALPMR_Alarm_Ack.cnf(-) (CREP,ERRCLS, ERRCODE)		Alarm Ack.cnf(-)	FSPM	AREP
CM_Connect.cnf(+) (AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	MAP_CON_CNFP	Connect.cnf(+)	FSPM	AREP AR Response Block List of IO CR Response Blocks Alarm CR Response Module Diff Block
CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_CON_CNFM	Connect.cnf(-)	FSPM	AREP Error Decode Error Code 1 Error Code 2
CM_Release.cnf(+) (AREP, ControlBlock)	MAP_REL_CNFP	Release.cnf(+)	FSPM	AREP Session Key
CM_Release.cnf (-) (AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	MAP_REL_CNFM	Release.cnf(-)	FSPM	AREP Error Decode Error Code 1 Error Code 2
		Abort.ind	FSPM	AREP
CM_DControl.cnf(+) (AREP, ControlBlock)	AREP	End Of Parameter.cnf(+)	FSPM	AREP Session Key Alarm Sequence Number
CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	AREP	End Of Parameter.cnf(-)	FSPM	AREP Error Decode Error Code 1 Error Code 2
CM_CControl.ind (AREP, ControlBlock, ModulDiffBlock)	AREP	Application ready.ind	FSPM	AREP Session Key Alarm Sequence Number Module Diff Block Comm Diff Block
ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
IFW_Error.ind(CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
IFW_Error.ind(CREP, ERRCLS, ERRCODE)	MAP_DIAGEVT_IND	Diagnosis Event.ind	FSPM	AREP CREP Alarm Item
TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	local action			
LMPM_Schedule_add.cnf (+) (CREP)	local action			
LMPM_Schedule_add.cnf (-) (CREP)	local action			
IFW_Schedule_add.cnf(Port)	local action			
IFW_Schedule_remove.cnf (Port)	local action			

5.5.3.1.2 Parameters of FSPMCTL primitives

The parameters used with the primitives exchanged between the FSPMCTL and the AP-Context are described in FAL Service Definition.

5.5.3.2 State machine description

5.5.3.3 FSPMCTL state table

The FSPMCTL defines no state machine because all services are transferred to the underlying protocol machines.

5.5.3.4 Functions

Table 286 shows the functions defined for FSPMCTL, which are used by services primitives issued by the FAL user.

Table 286 — Function used by AP-Context (FAL user) to FSPMCTL

Name	Function
MAP_READ_REQ	map service parameter 1:1 (=)
MAP_READ_INPD_REQ	AREP=AREP API=API TargetUUID=Target ARUUID SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x8028 SeqNumber=Seq Number Length=Length
MAP_READ_OUTPD_REQ	AREP=AREP API=API TargetUUID=Target ARUUID SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x8029 SeqNumber=Seq Number Length=Length
MAP_READ_LOG_REQ	AREP=AREP API=0 TargetUUID=0 SlotNumber=0 SubslotNumber=0 Index =0xF830 SeqNumber=Seq Number Length=Length
MAP_READ_DIAG_REQ	AAREP=AREP SeqNumber=Seq Number Length=Length if (Diagnosis Item = CHANNEL) then if (!API&&!Slot&&!Subslot) then API=0 if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE00A else if ((API&&!Slot&&!Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xF00A else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0

Name	Function
	<pre> Index = 0xC00A else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index = 0x800A if (Diagnosis Item = MANUFACT+URER_ERROR_APPEARS) then if (!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index = 0xE00B else if ((API&&!Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index = 0xF00B else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index = 0xC00B else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index = 0x800B if (Diagnosis Item = MANUFACTURER_ALL) then if (!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index = 0xE00C else if ((API&&!Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index = 0xF00C else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index = 0xC00C else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index = 0x800C </pre>
MAP_READ_EID_REQ	<pre> AREP=AREP SeqNumber=Seq Number Length=Length if (!API&&!Slot&&!Subslot) then </pre>

Name	Function
	API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE001 else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC001 else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x8001
MAP_READ_RID_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if (API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xF000 else if ((!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE000 else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC000 else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x8000
MAP_READ_IDDIFF_REQ	AREP=AREP SeqNumber=Seq Number Length=Length API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE002
MAP_READ_RPD_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if ((!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0

Name	Function
	SubslotNumber=0 Index =0xE02A else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC02A else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802A
MAP_READ_EPD_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if ((!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE02B else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC02B else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802B
MAP_READ_APD_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if ((!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE02F else if ((API&&Slot&&!Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC02F else if ((API&&Slot&&Subslot) then API=API if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802F
MAP_READ_IRD_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if ((!API&&!Slot&&!Subslot) then API=0 if (Target ARUUUID<>0) then TargetUUID=Target ARUUUID else

Name	Function
	TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE02C else if ((API&&Slot&&Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC02C else if ((API&&Slot&&Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802C
MAP_READ_RSYNC_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if ((API&&Slot&&Subslot) then API=0 if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE02D else if ((API&&Slot&&Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC02D else if ((API&&Slot&&Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802D
MAP_READ_ESYNC_REQ	AREP=AREP SeqNumber=Seq Number Length=Length if ((API&&Slot&&Subslot) then API=0 if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xE02E else if ((API&&Slot&&Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=0 Index =0xC02E else if ((API&&Slot&&Subslot) then API=API if (Target ARUUID<>0) then TargetUUID=Target ARUUID else TargetUUID=NULL SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802E
MAP_READ_PDEV_REQ	AREP=AREP SeqNumber=Seq Number Length=Length

Name	Function
	API=0 TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xF831
MAP_READ_ARC_REQ	AREP=AREP SeqNumber=Seq Number Length=Length API=0 TargetUUID=NULL SlotNumber=0 SubslotNumber=0 Index =0xF820
MAP_READ_ISOM_REQ	AREP=AREP SeqNumber=Seq Number Length=Length API=API TargetUUID=TargetUUID SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x8030
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_EPD_REQ	AREP=AREP Multiple=Multiple TargetUUID=NULL SeqNumber=Seq Number Length=Length Data={Expected List of Ports} if ((API&&Slot&&Subslot) then API=0 SlotNumber=0 SubslotNumber=0 Index =0xE02A else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=0 Index =0xC02A else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802A
MAP_WRITE_APD_REQ	AREP=AREP Multiple=Multiple TargetUUID=NULL SeqNumber=Seq Number Length=Length Data={Expected List of Ports} if ((API&&Slot&&Subslot) then API=0 SlotNumber=0 SubslotNumber=0 Index =0xE02F else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=0 Index =0xC02F else if ((API&&Slot&&Subslot) then

Name	Function
	API=API SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802F
MAP_WRITE_IRD_REQ	AREP=AREP Multiple TargetUUID=NULL SeqNumber=Seq Number Length=Length Data={IR Data} if ((API&&Slot&&Subslot) then API=0 SlotNumber=0 SubslotNumber=0 Index =0xE02C else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=0 Index =0xC02C else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802C
MAP_WRITE_ESYNC_REQ	AREP=AREP Multiple=Multiple TargetUUID=NULL SeqNumber=Seq Number Length=Length Data={Sync Data} if ((API&&Slot&&Subslot) then API=0 SlotNumber=0 SubslotNumber=0 Index =0xE02E else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=0 Index =0xC02E else if ((API&&Slot&&Subslot) then API=API SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x802E
MAP_WRITE_ISOM_REQ	AREP=AREP Multiple=Multiple TargetUUID=Target ARUUID SeqNumber=Seq Number Length=Length Data=IsoM Data API=API SlotNumber=Slot Number SubslotNumber=Subslot Number Index =0x8030
MAP_CON_REQ	AREP=AREP ARBlockReq=AR Parameter Block ListOfIOCRBlockReq=List of IO CR Parameter Blocks AlarmCRBlockReq=Alarm CR Parameter Block ListOfExpectedSubmoduleBlockReq=List of Expected Submodule Blocks PrmCount ListOfWriteBlockReq Parameter Server Block Multicast CR Block
MAP_REL_REQ	AREP=AREP ReleaseBlock.SessionKey=SessionKey
MAP_ALREADY_RES+	AREP=AREP

Name	Function
	ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number)
MAP_AREADY_RES-	AREP=AREP Error Decode Error Code 1 Error Code 2
MAP_EOP_REQ	AREP=AREP ControlBlockConnect.SessionKey=Session Key^(ControlBlockPlug.SessionKey=Session Key,ControlBlockPlug.AlarmSequenceNumber=Alarm Sequence Number)
MAP_GIN_REQ	CREP=CREP
MAP_GOUTIOCS_REQ	CREP=CREP
MAP_SAS_REQ	CREP=AREP.CREP D_Status={DataValid Flag, AR State Flag, ProviderState Flag, ProblemIndicator Flag}
MAP_SINIOCS_REQ	CREP=CREP Data=IOCS
MAP_SOUT_REQ	CREP=CREP Data={IOPS,Subslot Output Data}
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_AACK_REQ	CREP=AREP.CREP API Alarm_Type=Alarm Type Slot_Number=Slot Number Subslot_Number=Subslot Number Alarm_Specifier=Alarm Specifier Sequence_Number PNIO_Status

Table 287 shows the functions defined for FSPMCTL, which are used by services primitives issued by the FSPMCTL.

Table 287 — Functions used by FSPMCTL to AP-Context (FAL user)

Name	Function
MAP_READ_CNF+	if(ServiceReqWasRead(AREP,Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Add Data 1 = AddData1 Add Data 2 =AddData2 Length = Length Data = Data
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_INPD_CNF+	if(ServiceReqWasReadInput(AREP,Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Length = Length

Name	Function
	Length Data = Data.LengthData Length IOCS = Data.LengthIOCS Length IOPS = Data.LengthIOPS IOCS = Data.IOCS IOPS = Data.IOPS Subslot Input Data=Data.DataObjectElement
MAP_READ_INPD_CNF-	if(ServiceReqWasReadInput(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_OUTPD_CNF+	if(ServiceReqWasReadOutput(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Length = Length Length Data=Data.RecordOutputDataObjectElement.LengthData Length IOCS=Data.RecordOutputDataObjectElement.LengthIOCS Length IOPS=Data.RecordOutputDataObjectElement.LengthIOPS IOCS=Data.RecordOutputDataObjectElement.IOCS IOPS=Data.RecordOutputDataObjectElement.IOPS Output Data=Data.RecordOutputDataObjectElement.Data Substitute Mode=Data.RecordOutputDataObjectElement.SubstituteMode Substitute Active Flag=Data.RecordOutputDataObjectElement.SubstituteActiveFlag Output Substitute Data=Data.RecordOutputDataSubstituteObjectElement.SubstituteData
MAP_READ_OUTPD_CNF-	if(ServiceReqWasReadOutput(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_LOG_CNF+	if(ServiceReqWasReadLogbook(AREP, Seq Number)=TRUE) then AREP=AREP Seq Number=Seq Number Length=Length Current Local Time Stamp=Data.ActualLocalTimeStamp Number of Log Entries=Data.NumberOfLogEntries List of Entries Local Time Stamp=Data.LocalTimeStamp AR UUID=Data.ARUUID PNIO Status=Data.PNIOStatus Entry Detail=Data.EntryDetail
MAP_READ_LOG_CNF-	if(ServiceReqWasReadLogbook(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_DIAG_CNF+	if(ServiceReqWasReadDeviceDiagnosis(AREP, Seq Number)=TRUE) then AREP=AREP Seq Number=Seq Number Length=Length List of Diagnosis Data = Data
MAP_READ_DIAG_CNF-	if(ServiceReqWasReadDeviceDiagnosis(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Error Decode = ErrorDecode

Name	Function
	Error Code 1 = ErrorCode1 Error Code 2 = ErrorCode2 Add Data 1 = AddData1 Add Data 2 =AddData2
MAP_READ_EID_CNF+	if(ServiceReqWasReadExpectedIdentification(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Length = Length List of Slots = Data
MAP_READ_EID_CNF-	if(ServiceReqWasReadExpectedIdentification(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_RID_CNF+	if(ServiceReqWasReadExpectedIdentification(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Length = Length List of Slots = Data
MAP_READ_RID_CNF-	if(ServiceReqWasReadRealIdentification(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_IDDIFF_CNF+	iif(ServiceReqWasReadIdentificationDifference(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number = Seq Number Length = Length List of Slots = Data
MAP_READ_IDDIFF_CNF-	if(ServiceReqWasReadIdentificationDifference(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_RPD_CNF+	if(ServiceReqWasReadRealPortData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq number Length = Length Real List of Ports = Data
MAP_READ_RPD_CNF-	if(ServiceReqWasReadRealPortData(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_EPD_CNF+	if(ServiceReqWasReadExpectedPortData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq number Length = Length Expected List of Ports

Name	Function
MAP_READ_EPD_CNF-	if(ServiceReqWasReadExpectedPortData(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_APD_CNF+	if(ServiceReqWasReadExpectedPortData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq number Length = Length Adjusted List of Ports
MAP_READ_APD_CNF-	if(ServiceReqWasReadExpectedPortData(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_IRD_CNF+	if(ServiceReqWasReadIRDData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq number Length = Length IR Data = Data
MAP_READ_IRD_CNF-	if(ServiceReqWasReadIRDData(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_RSYNC_CNF+	if(ServiceReqWasReadRealSyncData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq number Length = Length Sync Data = Data
MAP_READ_RSYNC_CNF-	if(ServiceReqWasReadRealSyncData(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_ESYNC_CNF+	if(ServiceReqWasReadExpectedSyncData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq number Length = Length Sync Data = Data
MAP_READ_ESYNC_CNF-	if(ServiceReqWasReadExpectedSyncData(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_PDEV_CNF+	if(ServiceReqWasReadPDevData(AREP, Seq Number)=TRUE) then AREP = AREP Seq Number =Seq Number

Name	Function
	Length = Length Real Port Data = Data.PDPortData IR Data = Data.PDIRData Real Sync Data = Data.PDSyncData
MAP_READ_PDEV_CNF-	if(ServiceReqWasReadPDevData(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_ARD_CNF+	if(ServiceReqWasReadARData(AREP, Seq Number)=TRUE) then AREP=AREP Seq Number=Seq Number Length=Length AR Data=Data
MAP_READ_ARD_CNF-	if(ServiceReqWasReadARData(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_ISOM_CNF+	if(ServiceReqWasReadIsoMData(AREP, Seq Number)=TRUE) then AREP=AREP Seq Number=Seq Number Length=Length IsoM Data=Data
MAP_READ_ISOM_CNF-	if(ServiceReqWasReadIsoMData(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_WRITE_CNF+	if(ServiceReqWasWrite(AREP, Seq Number)=TRUE) then AREP=AREP Multiple Seq Number =SeqNumber Add Data 1=AddData1 Add Data 2=AddData2
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_EPD_CNF+	if(ServiceReqWasWriteExpectedPortData(AREP, Seq Number)=TRUE) then AREP=AREP Multiple = Multiple Seq Number =SeqNumber
MAP_WRITE_EPD_CNF-	if(ServiceReqWasWriteExpectedPortData(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

Name	Function
MAP_WRITE_APD_CNF+	if(ServiceReqWasWriteExpectedPortData(AREP, Seq Number)=TRUE) then AREP=AREP Multiple = Multiple Seq Number =SeqNumber
MAP_WRITE_APD_CNF-	if(ServiceReqWasWriteExpectedPortData(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_IRD_CNF+	if(ServiceReqWasWriteIRDData(AREP, Seq Number)=TRUE) then AREP=AREP Multiple Seq Number =SeqNumber
MAP_WRITE_IRD_CNF-	if(ServiceReqWasWriteIRDData(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_ESYNC_CNF+	if(ServiceReqWasWriteSyncData(AREP, Seq Number)=TRUE) then AREP=AREP Multiple Seq Number =SeqNumber
MAP_WRITE_ESYNC_CNF-	if(ServiceReqWasWriteSyncData(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_CON_CNF+	AREP =AREP AR Response Block=ARBlockRes List of IO CR Response Blocks=ListOfIOCRBlockRes Alarm CR Response=AlarmCRBlockRes Block Module Diff Block=ModuleDiffBlock
MAP_CON_CNF-	AREP =AREP Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2
MAP_REL_CNF+	AREP=AREP Session Key=ReleaseBlock.SessionKey
MAP_REL_CNF-	AREP =AREP Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2
MAP_EOP_CNF+	AREP=AREP Session Key= ControlBlockConnect.SessionKey^ControlBlockPlug.SessionKey Alarm Sequence Number=ControlBlockPlug.AlarmSequenceNumber
MAP_EOP_CNF-	AREP =AREP Error Decode =ErrorDecode Error Code 1 =ErrorCode1 Error Code 2 =ErrorCode2
MAP_AREADY_IND	AREP=AREP Session Key=ControlBlockPlug.SessionKey^ControlConnectPlug.SessionKey

Name	Function
	Module Diff Block=ModulDiffBlock Comm Diff Block
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 AREP=AREP CREP=CREP Slot Number Subslot Number
MAP_NEWSTATUS_IND	if APDUStatus has been changed AREP=CREP.AREP CREP=CREP Data Flag =APDU_Status AR State Flag =APDU_Status Provider State Flag =APDU_Status Problem Indicator Flag =APDU_Status
MAP_NODATA_IND	AREP=CREP.AREP CREP=CREP Slot Number =0 Subslot Number =0 Watchdog Flag=TRUE
MAP_GIND_CNF+	AREP=CREP IOPS=Data Subslot Input Data=Data New Flag=New_Flag IOCS=Data
MAP_GIND_CNF-	AREP=CREP.AREP other ignore
MAP_SIN_CNF	AREP=CREP.AREP other ignore
MAP_SAS_CNF	AREP=CREP.AREP other ignore
MAP_IIOCS_CNF	AREP=CREP.AREP other ignore
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_AA_CNF	AREP=CREP.AREP other ignore

5.6 Application Relationship Protocol Machines (ARPMs)

5.6.1 PPM

5.6.1.1 Primitive definitions

5.6.1.1.1 Primitives exchanged between PPM and FSPMDEV or FSPMCTL

Table 288 shows the service primitives including their associated parameters issued by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV) and received by PPM.

Table 288 — Primitives issued by FSPMDEV or FSPMCTL to PPM

Primitive name	Source	Associated parameters	Functions
Set_Prov_Data.req	FSPM	CREP Data	The service Set_Prov_Data contains new user data for the PPM internal buffer.

Primitive name	Source	Associated parameters	Functions
Set_Prov_Status.req	FSPM	CREP D_Status	The service Set_Prov_Status contains a new provider status value for the PPM internal buffer.

Table 289 shows the service primitives including their associated parameters issued by PPM and received by the FSPMCTL or FSPMDEV.

Table 289 — Primitives issued by PPM to FSPMDEV or FSPMCTL

Primitive name	Source	Associated parameters	Functions
Set_Prov_Data.cnf(+)	PPM	CREP	This service primitive indicates that the Set_Prov_Data service succeeded.
Set_Prov_Data.cnf(-)	PPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Set_Prov_Data service failed.
Set_Prov_Status.cnf(+)	PPM	CREP	This service primitive indicates that the Set_Prov_Status service succeeded.
Set_Prov_Status.cnf(-)	PPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Set_Prov_Status service failed.

5.6.1.1.2 Primitives exchanged between PPM and CMDEV or CMCTL

Table 290 shows the service primitives including their associated parameters issued by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV) and received by PPM.

Table 290 — Primitives issued by CMDEV or CMCTL to PPM

Primitive name	Source	Associated parameters	Functions
Activate.req	FSPM	CREP DA SA FrameId Prio VLAN SoftSyn ReductionRatio Phase Sequence Default_Values Default_Status	The service Activate initializes the PPM and loads the schedule to the DMPM.
Close.req	FSPM	CREP	The service Close deinitializes the PPM and stops the transmission of data. The schedule is cleared.

Table 291 shows the service primitives including their associated parameters issued by PPM and received by the CMCTL or CMDEV.

Table 291 — Primitives issued by PPM to CMDEV or CMCTL

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	PPM	CREP	This service primitive indicates that the Activate service succeeded.
Activate.cnf(-)	PPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Activate service failed.
Close.cnf(+)	PPM	CREP	This service primitive indicates that the Close service succeeded.
Close.cnf(-)	PPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Close service failed.
Start.ind	PPM	CREP	This service primitive indicates the first transmission of data.
Error.ind	PPM	CREP No_Data_Send	This service primitive indicates a failure during transmission of data.

5.6.1.1.3 Primitives exchanged between PPM and LMPM

Table 292 shows the service primitives including their associated parameters issued by LMPM and received by PPM.

Table 292 — Primitives issued by LMPM to PPM

Primitive name	Source	Associated parameters	Functions
LMPM_Time_Event.ind	LMPM	CREP Cycle	

Table 293 shows the service primitives including their associated parameters issued by PPM and received by LMPM.

Table 293 — Primitives issued by PPM to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_Schedule_add.req	PPM	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	PPM	CREP	

5.6.1.1.4 Parameters of PPM primitives

The parameters used with the primitives exchanged between the FSPM and the PPM are described in FAL Service Definition.

5.6.1.2 State machine description

The W-START state indicates that an initialisation is needed. The Activate service sets the machine in the RUN state waiting for an Provider Data service requests and Time events. After issuing the transmission of data the CRUN state is used to wait for a confirmation. 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. A Close in CRUN has to wait for the outstanding confirmation in the WCON state before entering W-START.

Local variables of the PPM

Buffer_Data (OctetString)

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 CL-RPC-PDU with NDRDataCyclic.

NOTE DataItem* includes all DataItems for the dedicated PDU.

Buffer_Status (Unsigned 16)

This local variable contains the status (encoded as APDU_Status) which shall be used as storage of the status dedicated for conveyance of the next Data-RTC-PDU or CL-RPC-PDU with NDRDataCyclic.

New_Stat (Boolean)

This local variable will be used to indicate the first transmission successful transmission of Data after Open.

5.6.1.3 PPM state table

Table 294 contains the complete description of the PPM state table.

Table 294 — PPM state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	PPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status) /SoftSync => Initial_ReductionRatio := Reduction_Ratio New_Stat := FALSE; Buffer_Data := Default_Values Buffer_Status := Default_Status LMPM_Schedule_add.req (CREP,ReductionRatio, Phase:=NIL, Sequence) PPM_Activate.cnf (+)(CREP)	RUN
2	W-START	PPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status) /!SoftSync => New_Stat := FALSE; Buffer_Data := Default_Values Buffer_Status := Default_Status LMPM_Schedule_add.req (CREP,ReductionRatio, Phase, Sequence) PPM_Activate.cnf (+)(CREP)	RUN
3	W-START	PPM_Close.req(CREP) => PPM_Close.cnf (+)(CREP)	W-START
4	W-START	PPM_Set_Prov_Data.req (CREP,Data) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Prov_Data.cnf(-) (CREP,ERRCLS,ERRCODE)	W-START
5	W-START	PPM_Set_Prov_Status.req (CREP,D_Status) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Prov_Status.cnf(-) (CREP,ERRCLS,ERRCODE)	W-START
6	W-START	C_Data_cnf => ignore	W-START
7	RUN	PPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Activate.cnf (-)(CREP,ERRCLS,ERRCODE)	RUN
8	RUN	PPM_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) PPM_Close.cnf (+)(CREP)	W-START
9	RUN	PPM_Set_Prov_Data.req (CREP,Data) => Buffer_Data := Data PPM_Set_Prov_Data.cnf(+) (CREP)	RUN
10	RUN	PPM_Set_Prov_Status.req (CREP,D_Status) => Buffer_Status := D_Status PPM_Set_Prov_Status.cnf(+) (CREP)	RUN
11	RUN	C_Data_cnf => ignore	RUN

#	Current State	Event /Condition =>Action	Next State
12	RUN	LMPM_Time_Event.ind(CREP, Cycle) /!SoftSync => C_SDU := Buffer_Data, APDU_Status := (Cycle_Count, Buffer_Status) C_Data_req	CRUN
13	RUN	LMPM_Time_Event.ind(CREP, Cycle) /SoftSync && CYCLE_INBAND => C_SDU := Buffer_Data, Reduction_Ratio := Initial_Reduction_Ratio APDU_Status := (Buffer_Status) LMPM_Schedule_add.req (CREP, ReductionRatio, Phase:=NIL, Sequence) C_Data_req	CRUN
14	RUN	LMPM_Time_Event.ind(CREP, Cycle) /SoftSync && CYCLE_TOO_LONG => C_SDU := Buffer_Data, APDU_Status := (Buffer_Status) Reduction_Ratio := Reduction_Ratio – 1 LMPM_Schedule_add.req (CREP, ReductionRatio, Phase:=NIL, Sequence) C_Data_req	CRUN
15	RUN	LMPM_Time_Event.ind(CREP, Cycle) /SoftSync && CYCLE_TOO_SHORT => C_SDU := Buffer_Data, APDU_Status := (Buffer_Status) Reduction_Ratio := Reduction_Ratio + 1 LMPM_Schedule_add.req (CREP, ReductionRatio, Phase:=NIL, Sequence) C_Data_req	CRUN
16	CRUN	PPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status) => ERRCLS := CXTX ERRCODE := INVALID_STATE PPM_Activate.cnf (-)(CREP,ERRCLS,ERRCODE)	CRUN
17	CRUN	PPM_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) PPM_Close.cnf (+)(CREP)	WCON
18	CRUN	C_Data_cnf /LMPM_status != LS/IV && !New_Stat => New_Stat := TRUE PPM_Start.ind(CREP)	RUN
19	CRUN	C_Data_cnf /LMPM_status != LS/IV && New_Stat =>	RUN
20	CRUN	C_Data_cnf /LMPM_status == LS/IV => PPM_Error.ind(CREP,No_Data_Send)	RUN
21	CRUN	LMPM_Time_Event.ind(CREP, Cycle) => PPM_Error.ind(CREP,No_Data_Send)	CRUN
22	CRUN	PPM_Set_Prov_Data.req (CREP,Data) => Buffer_Data := Data PPM_Set_Prov_Data.cnf(+) (CREP)	CRUN

#	Current State	Event /Condition =>Action	Next State
23	CRUN	PPM_Set_Prov_Status.req (CREP,D_Status) => Buffer_Status := D_Status PPM_Set_Prov_Status.cnf(+) (CREP)	CRUN
24	WCON	PPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status) => New_Stat := FALSE; Buffer_Data := Default_Values Buffer_Status := Default_Status LMPM_Schedule_add.req (CREP,ReductionRatio, Phase, Sequence) PPM_Activate.cnf (+)(CREP)	CRUN
25	WCON	PPM_Close.req(CREP) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Close.cnf (-)(CREP,ERRCLS,ERRCODE)	WCON
26	WCON	C_Data_cnf /LMPM_status != LS/IV =>	W-START
27	WCON	C_Data_cnf /LMPM_status == LS/IV => PPM_Error.ind(CREP,No_Data_Send)	W-START
28	WCON	LMPM_Time_Event.ind(CREP, Cycle) => PPM_Error.ind(CREP,No_Data_Send)	WCON
29	WCON	PPM_Set_Prov_Data.req (CREP,Data) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Prov_Data.cnf(-) (CREP,ERRCLS,ERRCODE)	WCON
30	WCON	PPM_Set_Prov_Status.req (CREP,D_Status) => ERRCLS := CTXT ERRCODE := INVALID_STATE PPM_Set_Prov_Status.cnf(-) (CREP,ERRCLS,ERRCODE)	WCON

5.6.1.4 Functions

Table 295 contains the functions or macros used by the PPM, their arguments and their descriptions.

Table 295 — Functions used by the PPM

Function name	Description
CYCLE_INBAND	(SendClockFactor/2 >= (CPM.Cycle – CycleCounter) MOD SendClockFactor*Initial_ReductionRatio) ((CPM.Cycle – CycleCounter) MOD SendClockFactor*Initial_ReductionRatio >= SendClockFactor*Initial_ReductionRatio-SendClockFactor/2)
CYCLE_TOO_LONG	!CYCLE_INBAND && ((CPM.Cycle – CycleCounter) MOD SendClockFactor*Initial_ReductionRatio < SendClockFactorInitial_ReductionRatio/2) && !((ReductionRatio – Initial_ReductionRatio) = -1)
CYCLE_TOO_SHORT	!CYCLE_INBAND && ((CPM.Cycle – CycleCounter) MOD SendClockFactor*Initial_ReductionRatio >= SendClockFactor*Initial_ReductionRatio/2) && !((ReductionRatio – Initial_ReductionRatio) = 1)
CYCLE_OK	!CYCLE_TOO_LONG && !CYCLE_TOO_SHORT
C_Data_cnf	if (txOption == RTC) LMPM_C_Data.cnf (CREP, LMPM_status, Cycle) if (txOption == UDP) UDP_C_Data.cnf (IPPort, CREP, LMPM_status, Cycle)

Function name	Description
C_Data_req	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)

NOTE 1 The arithmetic operations used in Table 295 are signed integer operations without overflow. ABS means integer without sign and MOD is the remainder of an integer division.

NOTE 2 SoftSync, as a local device attribute, is only applicable starting with the reduction ratio greater than 3.

5.6.2 CPM

5.6.2.1 Primitive definitions

5.6.2.1.1 Primitives exchanged between CPM and FSPMDEV or FSPMCTL

Table 296 shows the service primitives including their associated parameters issued by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV) and received by CPM.

Table 296 — Primitives issued by FSPMDEV or FSPMCTL to CPM

Primitive name	Source	Associated parameters	Functions
Get_Cons_Data.req	FSPM	CREP	The service Get_Cons_Data reads new user data for the CPM internal buffer.
Get_Cons_Status.req	FSPM	CREP	The service Get_Prov_Status reads a new provider status value for the PPM internal buffer.
Set_RedRole.req	FSPM	CREP RedRole	The service CPM_Set_RedRole contains the role for redundancy (e.g. primary).

Table 297 shows the service primitives including their associated parameters issued by CPM and received by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV).

Table 297 — Primitives issued by CPM to FSPM

Primitive name	Source	Associated parameters	Functions
Get_Cons_Data.cnf(+)	CPM	CREP Data New_Flag	This service primitive indicates that the Get_Cons_Data service succeeded.
Get_Cons_Data.cnf(-)	CPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Get_Cons_Data service failed.
Get_Cons_Status.cnf(+)	CPM	CREP Status RecvCounter	This service primitive indicates that the Get_Cons_Status service succeeded.
Get_Cons_Status.cnf(-)	CPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Get_Cons_Status service failed.
Set_RedRole.cnf	CPM	CREP	This service primitive indicates that the CPM_Set_RedRole service succeeded.
New_Cons_Data.ind	CPM	CREP APDU_Status	This service primitive indicates that new Consumer Data have been received.
Start.ind	CPM	CREP	This service primitive indicates that new Consumer Data have been received first time and the protocol machine has just started with its operation.
NoData.ind	CPM	CREP	This service primitive indicates that the Watch Dog time has expired.

5.6.2.1.2 Primitives exchanged between CPM and CMDEV or CMCTL

Table 298 shows the service primitives including their associated parameters issued by CM of the IO controller (CMCTL) or of the IO device (CMDEV) and received by CPM.

Table 298 — Primitives issued by CMDEV or CMCTL to CPM

Primitive name	Source	Associated parameters	Functions
Activate.req	FSPM	CREP DA SA FrameId Prio VLAN StartTon Timeout_Base WatchdogFactor DataHoldFactor Default_Value Default_Status	The service Activate initializes the CPM and loads the schedule to the DMPM.
Close.req	FSPM	CREP	The service Close deinitializes the CPM and stops the transmission of data. The schedule is cleared.

Table 299 shows the service primitives including their associated parameters issued by CPM and received by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV).

Table 299 — Primitives issued by CPM to CMCTL or CMDEV

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	CPM	CREP	This service primitive indicates that the Activate service succeeded
Activate.cnf(-)	CPM	CREP ERRCLS ERRCODE	This service primitive indicates that the Activate service failed
Close.cnf(+)	CPM	CREP	This service primitive indicates that the Close service succeeded
Stop.ind	CPM	CREP	This service primitive indicates that Data Hold time has expired.

5.6.2.1.3 Primitives exchanged between CPM and LMPM

Table 300 shows the service primitives including their associated parameters issued by LMPM and received by CPM.

Table 300 — Primitives issued by LMPM to CPM

Primitive name	Source	Associated parameters	Functions
LMPM_Time_Event.ind	LMPM	CREP Cycle	

Table 300 shows the service primitives including their associated parameters issued by LMPM and received by CPM.

Table 301 — Primitives issued by CPM to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_Schedule_add.req	CPM	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	CPM	CREP	

5.6.2.1.4 Parameter of CPM primitives

The parameters used with the primitives exchanged between the FSPM and the CPM are described in the Service Definition.

5.6.2.2 State machine description

The W-START state indicates that the initialisation is needed. The Activate service sets the machine in the RUN state or in the FRUN 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 an 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.

Local variables of the CPM

Cycle (Unsigned 16)

This local variable contains the last valid Cycle Counter conveyed with a LMPM_C_Data indication with valid data.

RecvCnt (Unsigned 16)

This local variable contains the counter value of the received cyclic PDU since last Cons Status request.

Buffer_Data (Octet String)

This local variable contains the last valid service data unit conveyed with a LMPM_C_Data indication.

Buffer_Status (Unsigned 16)

This local variable contains the last valid APDU_Status conveyed with a LMPM_C_Data indication.

New_Data (Boolean)

This local variable signals the receipt of a valid LMPM_C_Data indication (set to FALSE by a Get_Cons_Data request).

WDt (Unsigned 16)

This local variable contains the counter value of the time events since last LMPM_C_Data indication.

DHt (Unsigned 16)

This local variable contains the counter value of the time events since last LMPM_C_Data indication with valid data.

5.6.2.3 CPM state table

Table 302 contains the complete description of the CPM state table.

Table 302 — CPM state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	CPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, Exp_Lenth, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status) /StartTon = TRUE => InitCy := TRUE, Store DA,SA,FrameID,Prio, VLAN, Timeout_Base, WatchDogFactor, DataHoldFactor, Default_Value, Default_Status New_Data := FALSE, WDt := 0, DHt := 0, Cycle := 0, RecvCnt := 0, Buffer_Data := Default_Value, Buffer_Status := Default_Status Primary := TRUE LMPM_Schedule_add.req (CREP, ReductionRatio:=Timeout_Base, Phase:=0, Sequence:=0) CPM_Activate.cnf (+)(CREP)	RUN
2	W-START	CPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, Exp_Lenth, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status) /StartTon = FALSE	FRUN

#	Current State	Event /Condition =>Action	Next State
		=> Store DA,SA,FrameID,Prio, VLAN, Timeout_Base, WatchDogFactor, DataHoldFactor, Default_Value, Default_Status New_Data := FALSE, WDt := 0, DHt := 0, Cycle := 0, RecvCnt := 0, Buffer_Data := Default_Value, Buffer_Status := Default_Status Primary := TRUE LMPM_Schedule_add.req (CREP, ReductionRatio:=Timeout_Base, Phase:=0, Sequence:=0) CPM_Activate.cnf (+)(CREP)	
3	W-START	CPM_Close.req(CREP) => CPM_Close.cnf (+)(CREP)	W-START
4	W-START	CPM_Get_Cons_Status.req (CREP) => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Get_Cons_Status.cnf(-) (CREP,ERRCLS,ERRCODE)	W-START
5	W-START	CPM_Set_RedRole.req (CREP, RedRole) => Primary := RedRole CPM_Set_RedRole.cnf (CREP)	W-START
6	W-START	CPM_Get_Cons_Data.req (CREP) => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Get_Cons_Data.cnf(-) (CREP,ERRCLS,ERRCODE)	W-START
7	W-START	C_Data_ind => ignore	W-START
8	FRUN	CPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, Exp_Lenth, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status) => ERRCLS := CTXT ERRCODE := INVALID_STATE CPM_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	FRUN
9	FRUN	CPM_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) CPM_Close.cnf (+)(CREP)	W-START
10	FRUN	CPM_Get_Cons_Status.req (CREP) => Status := Cycle, Buffer_Status, RecvCounter := RecvCnt, RecvCnt := 0 CPM_Get_Cons_Status.cnf(+) (CREP, Status, RecvCounter)	FRUN
11	FRUN	CPM_Set_RedRole.req (CREP, RedRole) => Primary := RedRole CPM_Set_RedRole.cnf (CREP)	FRUN
12	FRUN	CPM_Get_Cons_Data.req (CREP) => Data := Buffer_Data, New_Flag := New_Data, New_Data := FALSE CPM_Get_Cons_Data.cnf(+) (CREP, Data, New_Flag)	FRUN
13	FRUN	C_Data_ind /APDU_Status.DataStatus.DataValid && CHECK_PAR_C => InitCy := FALSE, New_Data := TRUE,	RUN

#	Current State	Event /Condition =>Action	Next State
		DHt := 0, WDt := 0, Cycle:= APDU_Status.Cycle_Counter, Buffer_Data := Filter (Data) Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_Start.ind(CREP) CPM_New_Cons_Data.ind(CREP, APDU_Status)	
14	FRUN	C_Data_ind / !APDU_Status.DataStatus.DataValid && !Primary && CHECK_PAR_C => DHt := 0, WDt := 0, Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1	FRUN
15	FRUN	C_Data_ind /!APDU_Status.DataStatus.DataValid && Primary && CHECK_PAR_C => WDt := 0, Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1	FRUN
16	FRUN	C_Data_ind /!CHECK_PAR_C => ignore	FRUN
17	FRUN	LMPM_Time_Event.ind(CREP) => ignore	FRUN
18	RUN	CPM_Activate.req(CREP, DA, SA, FrameID, Prio, VLAN, Exp_Lenth, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status) => ERRCLS := CTEXT ERRCODE := INVALID_STATE CPM_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	RUN
19	RUN	CPM_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) CPM_Close.cnf (+)(CREP)	W-START
20	RUN	CPM_Get_Cons_Status.req (CREP) => Status := Cycle, Buffer_Status, RecvCounter := RecvCnt, RecvCnt := 0 CPM_Get_Cons_Status.cnf(+) (CREP, Status, RecvCounter)	RUN
21	RUN	CPM_Set_RedRole.req (CREP, RedRole) => Primary := RedRole CPM_Set_RedRole.cnf (CREP)	RUN
22	RUN	CPM_Get_Cons_Data.req (CREP) => Data := Buffer_Data, New_Flag := New_Data, New_Data := FALSE CPM_Get_Cons_Data.cnf(+) (CREP, Data, New_Flag)	RUN
23	RUN	C_Data_ind /((APDU_Status.CycleCounter-Cycle)>> InitCy) && (APDU_Status.DataStatus.DataValid) && CHECK_PAR_C => InitCy := FALSE, New_Data := TRUE, DHt := 0, WDt := 0, Cycle:= APDU_Status.Cycle_Counter, Buffer_Data := Filter (Data)	RUN

#	Current State	Event /Condition =>Action	Next State
		Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1 CPM_New_Cons_Data.ind(CREP, APDU_Status)	
24	RUN	C_Data_ind /((APDU_Status.CycleCounter-Cycle)>> InitCy) && !APDU_Status.DataStatus.DataValid && !Primary && CHECK_PAR_C => InitCy := FALSE, DHt := 0, WDt := 0, Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1	RUN
25	RUN	C_Data_ind /((APDU_Status.CycleCounter-Cycle)<< && InitCy) && (!APDU_Status.DataStatus.DataValid && Primary) && CHECK_PAR_C => InitCy := FALSE, WDt := 0, Buffer_Status := APDU_Status RecvCnt := RecvCnt + 1	RUN
26	RUN	C_Data_ind /((APDU_Status.CycleCounter-Cycle)<< && !InitCy) !CHECK_PAR_C => ignore	RUN
27	RUN	LMPM_Time_Event.ind(CREP) /DHt > (DataHoldTime-1) && WDt > (WatchDogTime-1) => DHt := 0, WDt := 0 CPM_NoData.ind(CREP) CPM_Stop.ind(CREP)	FRUN
28	RUN	LMPM_Time_Event.ind(CREP) /DHt > (DataHoldTime-1) && WDt < (WatchDogTime) => DHt := 0, WDt := WDt+1 CPM_Stop.ind(CREP)	FRUN
29	RUN	LMPM_Time_Event.ind(CREP) /DHt < (DataHoldTime) && WDt > (WatchDogTime-1) => DHt := DHt + 1, WDt := 0 CPM_NoData.ind(CREP)	RUN
30	RUN	LMPM_Time_Event.ind(CREP) /DHt < (DataHoldTime) && WDt < (WatchDogTime) => DHt := DHt + 1, WDt := WDt + 1	RUN

5.6.2.4 Functions

Table 303 contains the functions or macros used by the CPM, their arguments and their descriptions.

Table 303 — Functions used by the CPM

Function name	Description
C_Data_ind	if (txOption == RTC) LMPM_C_Data.ind (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) if (txOption == UDP) UDP_C_Data.ind (IPPort, CREP, DA, SA, Prio, C_SDU, APDU_Status)
<<	result of the subtraction of two 16 bit unsigned integers is between -0xffff and 0 Sign overflows will not be calculated
>>	means not << (result between 0x8000 and 0x1000 or between 1 and 0x7FFF)
CHECK_PAR_C	C_SDU.Length == Exp_Length && SA == stored SA && APDU_Status.TransferStatus == 0 && if (txOption == UDP) IPPort == Exp_IPPort

5.6.3 ALPMI

5.6.3.1 Primitive definitions

5.6.3.1.1 Primitives exchanged between ALPMI and FSPMDEV or FSPMCTL

Table 304 shows the service primitives including their associated parameters issued by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV) and received by ALPMI.

Table 304 — Primitives issued by FSPMDEV or FSPMCTL to ALPMI

Primitive name	Source	Associated parameters	Functions
Alarm_Notification.req	FSPM	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.

Table 305 shows the service primitives including their associated parameters issued by ALPMI and received by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV).

Table 305 — Primitives issued by ALPMI to FSPMDEV or FSPMCTL

Primitive name	Source	Associated parameters	Functions
Alarm_Notification.cnf(+)	ALPMI	CREP	This service primitive indicates that the Alarm_Notification service succeeded.
Alarm_Notification.cnf(-)	ALPMI	CREP ERRCLS ERRCODE	This service primitive indicates that the Alarm_Notification service failed.
Alarm_Ack.ind	ALPMI	CREP Alarm_Type Slot_Number Subslot_Number Alarm_Specifier Sequence_Number PNIO_Status	This service primitive indicates the alarm acknowledge

5.6.3.1.2 Primitives exchanged between ALPMI and CMDEV or CMCTL

Table 306 shows the service primitives including their associated parameters issued by CM of the IO controller (CMCTL) or of the IO device (CMDEV) and received by ALPMI.

Table 306 — Primitives issued by CMDEV or CMCTL to ALPMI

Primitive name	Source	Associated parameters	Functions
Activate.req	CM	CREP DA SA VLAN RTATimeoutFactor MRetry	The service Activate initializes the ALPMI and requests the initialization of the APMS and APMR protocol machines.
Close.req	CM	CREP	The service Close deinitializes the ALPMI and closes the APMR and APMS protocol machines.

Table 307 shows the service primitives including their associated parameters issued by ALPMI and received by CM of the IO controller (CMCTL) or of the IO device (CMDEV).

Table 307 — Primitives issued by ALPMI to CMCTL or CMDEV

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	ALPMI	CREP	This service primitive indicates that the Activate service succeeded.
Activate.cnf(-)	ALPMI	CREP ERRCLS ERRCODE	This service primitive indicates that the Activate service failed.
Close.cnf(+)	ALPMI	CREP	This service primitive indicates that the Close service succeeded.
Error.ind	ALPMI	CREP ERRCLS ERRCODE	This service primitive indicates a failure during transmission of alarm data.

5.6.3.1.3 Primitives exchanged between ALPMI and APMR

Table 308 shows the service primitives including their associated parameters issued by APMR and received by ALPMI.

Table 308 — Primitives issued by APMR to ALPMI

Primitive name	Source	Associated parameters	Functions
APMR_A_Data.ind	APMR	CREP Data	
APMR_Ack.cnf(-)	APMR	CREP ERRCLS ERRCODE	
APMR_Ack.cnf(+)	APMR	CREP	
APMR_Activate.cnf(-)	APMR	CREP ERRCLS ERRCODE	
APMR_Activate.cnf(+)	APMR	CREP	
APMR_Close.cnf(-)	APMR	CREP ERRCLS ERRCODE	
APMR_Error.ind	APMR	CREP ERRCLS ERRCODE	

Table 309 shows the service primitives including their associated parameters issued by ALPMI and received by APMR.

Table 309 — Primitives issued by ALPMI to APMR

Primitive name	Source	Associated parameters	Functions
APMR_ACK.req	ALPMI	CREP	
APMR_Activate.req	ALPMI	CREP DA SA Frameld Prio VLAN	
APMR_Close.req	ALPMI	CREP	

5.6.3.1.4 Primitives exchanged between ALPMI and APMS

Table 310 shows the service primitives including their associated parameters issued by APMS and received by ALPMI.

Table 310 — Primitives issued by APMS to ALPMI

Primitive name	Source	Associated parameters	Functions
APMS_A_Data.cnf(-)	APMS	CREP ERRCLS ERRCODE	

Primitive name	Source	Associated parameters	Functions
APMS_A_Data.cnf(+)	APMS	CREP	
APMS_Activate.cnf(-)	APMS	CREP ERRCLS ERRCODE	
APMS_Activate.cnf(+)	APMS	CREP	
APMS_Close.cnf(-)	APMS	CREP ERRCLS ERRCODE	
APMS_Close.cnf(+)	APMS	CREP	
APMS_Error.ind	APMS	CREP ERRCLS ERRCODE	

Table 311 shows the service primitives including their associated parameters issued by ALPMI and received by APMS.

Table 311 — Primitives issued by ALPMI to APMS

Primitive name	Source	Associated parameters	Functions
APMS_A_Data.req	ALPMI	CREP Data	
APMS_Activate.req	ALPMI	CREP DA SA Frameld Prio VLAN RTATimeoutFactor MRetry	
APMS_Close.req	ALPMI	CREP	

5.6.3.1.5 Parameters of ALPMI primitives

The parameters used with the primitives exchanged between ALPMI and CMDEV, CMCTL and FSPM are described in the FAL service Definition.

5.6.3.2 State machine description

The W-START state indicates that the initialisation is needed. The Activate service sets the machine in the W-START-APMS for APMS initialization and then in the W-START-APMR state for APMR initialization. After successful initialization the state machine is waiting for Alarm Notification requests in the state A-Alarm and enters afterwards the W-ACK state waiting for the Alarm Ack. A Close service request is needed to reenter the W-START state after closing the APMS and APMR protocol machines.

5.6.3.3 ALPMI state table

Table 312 contains the complete description of the ALPMI state machine.

Table 312 — ALPMI state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> CREP.APMS := create(APMS) CREP.APMR := create(APMR)	W-START

#	Current State	Event /Condition =>Action	Next State
2	W-START	ALPMI_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) /CREP.Priority = Low => FrameId := 0xfe01 Prio := 5 APMS_Activate.req(CREP.APMS, DA, SA, FrameId, Prio, VLAN, RTATimeoutFactor, MRetry)	W-START-APMS
3	W-START	ALPMI_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) /CREP.Priority = High => FrameId := 0xfc01 Prio := 6 APMS_Activate.req(CREP.APMS, DA, SA, FrameId, Prio, VLAN, RTATimeoutFactor, MRetry)	W-START-APMS
4	W-START	ALPMI_Close.req(CREP) => ALPMI_Close.cnf(+)(CREP)	W-START
5	W-START	APMS_A_Data.cnf(+)(CREP) => ignore	W-START
6	W-START	APMS_A_Data.cnf(-) (CREP,ERRCLS,ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-START
7	W-START	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) => ALPMI_Alarm_Notification.cnf(-)(CREP)	W-START
8	W-START	APMR_A_Data.ind(CREP.APMR, Data) => ignore	W-START
9	W-START	APMR_Error.ind(CREP.APMR, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-START
10	W-START	APMS_Error.ind(CREP.APMS, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-START
11	W-START-APMS	APMS_Activate.cnf(+)(CREP.APMS) => APMR_Activate.req(CREP.APMR, DA, SA, FrameId, Prio, VLAN)	W-START-APMR
12	W-START-APMS	APMS_Activate.cnf(-)(CREP.APMS, ERRCLS, ERRCODE) => ALPMI_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-START
13	W-START-APMR	APMR_Activate.cnf(+)(CREP.APMR) => ALPMI_Activate.cnf(+)(CREP)	W-Alarm
14	W-START-APMR	APMR_Activate.cnf(-)(CREP.APMR, ERRCLS, ERRCODE) => ALPMI_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-START
15	W-Alarm	ALPMI_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) => ERRCLS:= PROTOCOL ERRCODE:= WRONG-STATE ALPMI_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-Alarm

#	Current State	Event /Condition =>Action	Next State
16	W-Alarm	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) => Data := Alarm Notification PDU APMS_A_Data.req(CREP.APMS, Data)	W-ACK
17	W-Alarm	ALPMI_Close.req(CREP) => APMR_Close.req(CREP.APMR)	W-CLOSE-APMR
18	W-Alarm	APMS_A_Data.cnf(+)(CREP) =>	W-Alarm
19	W-Alarm	APMS_A_Data.cnf(-) (CREP,ERRCLS,ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-Alarm
20	W-Alarm	APMR_A_Data.ind(CREP.APMR, Data) => ignore	W-Alarm
21	W-Alarm	APMR_Error.ind(CREP.APMR, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-Alarm
22	W-Alarm	APMS_Error.ind(CREP.APMS, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-Alarm
23	W-ACK	ALPMI_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) => ERRCLS:= PROTOCOL ERRCODE:= WRONG-STATE ALPMI_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-ACK
24	W-ACK	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) => ALPMI_Alarm_Notification.cnf(-)(CREP)	W-ACK
25	W-ACK	ALPMI_Close.req(CREP) => APMR_Close.req(CREP.APMR)	W-CLOSE-APMR
26	W-ACK	APMS_A_Data.cnf(+)(CREP) => ignore	W-ACK
27	W-ACK	APMS_A_Data.cnf(-) (CREP,ERRCLS,ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-ACK
28	W-ACK	APMR_A_Data.ind(CREP.APMR, Data) /Data != Alarm-Ack-PDU => ERRCLS:= PROTOCOL ERRCODE:= WRONG-ACK-PDU APMR_ACK.req(CREP.APMR) ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-AACK
29	W-ACK	APMR_A_Data.ind(CREP.APMR, Data) /Data = Alarm-Ack-PDU => APMR_ACK.req(CREP.APMR) ALPMI_Alarm_Notification.cnf(+)(CREP) ALPMI_Alarm_Ack.ind(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status)	W-AACK

#	Current State	Event /Condition =>Action	Next State
30	W-ACK	APMR_Error.ind(CREP.APMR, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-Alarm
31	W-ACK	APMS_Error.ind(CREP.APMS, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-Alarm
32	W-AACK	APMR_Ack.cnf(+)(CREP.APMR) =>	W-Alarm
33	W-AACK	APMR_Ack.cnf(-)(CREP.APMR, ERRCLS, ERRCODE) => ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-Alarm
34	W-CLOSE-APMR	APMR_Close.cnf(+)(CREP.APMR) => APMS_Close.req(CREP.APMS)	W-CLOSE-APMS
35	W-CLOSE-APMR	APMR_Close.cnf(-)(CREP.APMR, ERRCLS, ERRCODE) => APMS_Close.req(CREP.APMS) ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-CLOSE-APMS
36	W-CLOSE-APMS	APMS_Close.cnf(+)(CREP.APMS) => ALPMI_Close.cnf(+)(CREP)	W-START
37	W-CLOSE-APMS	APMS_Close.cnf(-)(CREP.APMS, ERRCLS, ERRCODE) => ALPMI_Close.cnf(+)(CREP) ALPMI_Error.ind(CREP, ERRCLS, ERRCODE)	W-START

5.6.3.4 Functions

There are no functions defined for ALPMI.

5.6.4 ALPMR

5.6.4.1 Primitive definitions

5.6.4.1.1 Primitives exchanged between ALPMR and FSPMDEV or FSPMCTL

Table 313 shows the service primitives including their associated parameters issued by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV) and received by ALPMR.

Table 313 — Primitives issued by FSPMDEV or FSPMCTL to ALPMR

Primitive name	Source	Associated parameters	Functions
Alarm_Ack.req	FSPM	CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status	The service Alarm_Ack conveys the alarm acknowledgement.

Table 314 shows the service primitives including their associated parameters issued by ALPMR and received by FSPM of the IO controller (FSPMCTL) or of the IO device (FSPMDEV).

Table 314 — Primitives issued by ALPMR to FSPMDEV or FSPMCTL

Primitive name	Source	Associated parameters	Functions
Alarm_Ack.cnf(+)	ALPMR	CREP	This service primitive indicates that the Alarm_Ack service succeeded.
Alarm_Ack.cnf(-)	ALPMR	CREP ERRCLS ERRCODE	This service primitive indicates that the Alarm_Ack service failed.

Primitive name	Source	Associated parameters	Functions
Alarm_Notification.ind	ALPMR	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.

5.6.4.1.2 Primitives exchanged between ALPMR and CMDEV or CMCTL

Table 315 shows the service primitives including their associated parameters issued by CM of the IO controller (CMCTL) or of the IO device (CMDEV) and received by ALPMR.

Table 315 — Primitives issued by CMDEV or CMCTL to ALPMR

Primitive name	Source	Associated parameters	Functions
Activate.req	CM	CREP DA SA VLAN RTATimeoutFactor MRetry	The service Activate initializes the ALPMR and requests the initialization of the APMS and APMR protocol machines.
Close.req	CM	CREP	The service Close deinitializes the ALPMI and closes the APMR and APMS protocol machines.

Table 316 shows the service primitives including their associated parameters issued by ALPMR and received by CM of the IO controller (CMCTL) or of the IO device (CMDEV).

Table 316 — Primitives issued by ALPMR to CMCTL or CMDEV

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	ALPMR	CREP	This service primitive indicates that the Activate service succeeded.
Activate.cnf(-)	ALPMR	CREP ERRCLS ERRCODE	This service primitive indicates that the Activate service failed.
Close.cnf(+)	ALPMR	CREP	This service primitive indicates that the Close service succeeded.
Error.ind	ALPMR	CREP, ERRCLS, ERRCODE	This service primitive indicates a failure during transmission of alarm data.

5.6.4.1.3 Primitives exchanged between ALPMR and APMR

Table 317 shows the service primitives including their associated parameters issued by APMR and received by ALPMR.

Table 317 — Primitives issued by APMR to ALPMR

Primitive name	Source	Associated parameters	Functions
APMR_A_Data.ind	APMR	CREP Data	
APMR_Ack.cnf(-)	APMR	CREP ERRCLS ERRCODE	
APMR_Ack.cnf(+)	APMR	CREP	
APMR_Activate.cnf(-)	APMR	CREP ERRCLS ERRCODE	
APMR_Activate.cnf(+)	APMR	CREP	

Primitive name	Source	Associated parameters	Functions
APMR_Close.cnf(-)	APMR	CREP ERRCLS ERRCODE	
APMR_Error.ind	APMR	CREP ERRCLS ERRCODE	

Table 318 shows the service primitives including their associated parameters issued by ALPMR and received by APMR.

Table 318 — Primitives issued by ALPMR to APMR

Primitive name	Source	Associated parameters	Functions
APMR_ACK.req	ALPMI	CREP	
APMR_Activate.req	ALPMI	CREP DA SA Frameld Prio VLAN	
APMR_Close.req	ALPMI	CREP	

5.6.4.1.4 Primitives exchanged between ALPMR and APMS

Table 319 shows the service primitives including their associated parameters issued by APMS and received by ALPMR.

Table 319 — Primitives issued by APMS to ALPMR

Primitive name	Source	Associated parameters	Functions
APMS_A_Data.cnf(-)	APMS	CREP ERRCLS ERRCODE	
APMS_A_Data.cnf(+)	APMS	CREP	
APMS_Activate.cnf(-)	APMS	CREP ERRCLS ERRCODE	
APMS_Activate.cnf(+)	APMS	CREP	
APMS_Close.cnf(-)	APMS	CREP ERRCLS ERRCODE	
APMS_Close.cnf(+)	APMS	CREP	
APMS_Error.ind	APMS	CREP ERRCLS ERRCODE	

Table 320 shows the service primitives including their associated parameters issued by ALPMR and received by APMS.

Table 320 — Primitives issued by ALPMR to APMS

Primitive name	Source	Associated parameters	Functions
APMS_A_Data.req	ALPMI	CREP Data	
APMS_Activate.req	ALPMI	CREP DA SA Frameld Prio VLAN RTATimeoutFactor MRetry	

Primitive name	Source	Associated parameters	Functions
APMS_Close.req	ALPMI	CREP	

5.6.4.1.5 Parameter of ALPMR primitives

The parameters used with the primitives exchanged between the ALPMR and CMDEV, CMCTL and FSPM are described in the service Definition.

5.6.4.2 State machine description

The W-START state indicates that the initialisation is needed. The Activate service sets the machine in the W-START-APMS for APMS initialization and then in the W-START-APMR state for APMR initialization. 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 the Alarm Ack. A Close service request is needed to reenter the W-START state after closing the APMS and APMR protocol machines.

5.6.4.3 ALPMR state table

Table 321 contains the complete description of the ALPMR state machine.

Table 321 — ALPMR state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> CREP.APMS := create(APMS) CREP.APMR := create(APMR)	W-START
2	W-START	ALPMR_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) /CREP.Priority = Low => Frameld := 0xfe01 Prio := 5 APMS_Activate.req(CREP.APMS, DA, SA, Frameld, Prio, VLAN, RTATimeoutFactor, MRetry)	W-START-APMS
3	W-START	ALPMR_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) /CREP.Priority = High => Frameld := 0xfc01 Prio := 6 APMS_Activate.req(CREP.APMS, DA, SA, Frameld, Prio, VLAN, RTATimeoutFactor, MRetry)	W-START-APMS
4	W-START	ALPMR_Close.req(CREP) => ALPMR_Close.cnf(+)(CREP)	W-START
5	W-START	ALPMR_Alarm_Ack.req(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status) => ERRCLS:= PROTOCOL ERRCODE:= WRONG-STATE ALPMR_Alarm_Ack.cnf(-)(CREP,ERRCLS, ERRCODE)	W-START
6	W-START	APMR_A_Data.ind(CREP.APMR, Data) => ignore	W-START
7	W-START	APMR_Error.ind(CREP.APMR, ERRCLS, ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-START
8	W-START	APMS_A_Data.cnf(+)(CREP) => ignore	W-START

#	Current State	Event /Condition =>Action	Next State
9	W-START	APMS_A_Data.cnf(-) (CREP,ERRCLS,ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-START
10	W-START	APMS_Error.ind(CREP.APMS, ERRCLS, ERRCODE) => ignore	W-START
11	W-START-APMS	APMS_Activate.cnf(+)(CREP.APMS) => APMR_Activate.req(CREP.APMR, DA, SA, FrameId, Prio, VLAN)	W-START-APMR
12	W-START-APMS	APMS_Activate.cnf(-)(CREP.APMS, ERRCLS, ERRCODE) => ALPMR_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-START
13	W-START-APMR	APMR_Activate.cnf(+)(CREP.APMR) => ALPMR_Activate.cnf(+)(CREP)	W-Notify
14	W-START-APMR	APMR_Activate.cnf(-)(CREP.APMR, ERRCLS, ERRCODE) => ALPMR_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-START
15	W-Notify	ALPMR_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) => ERRCLS:= PROTOCOL ERRCODE:= WRONG-STATE ALPMR_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-Notify
16	W-Notify	ALPMR_Alarm_Ack.req(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status) => ERRCLS:= PROTOCOL ERRCODE:= WRONG-STATE ALPMR_Alarm_Ack.cnf(-)(CREP,ERRCLS, ERRCODE)	W-Notify
17	W-Notify	ALPMR_Close.req(CREP) => ALPMR_Close.cnf(+)(CREP)	W-START
18	W-Notify	APMR_A_Data.ind(CREP.APMR, Data) /Data != Alarm-Notification-PDU => ERRCLS:= PROTOCOL ERRCODE:= WRONG-NOTIFICATION-PDU APMR_ACK.req(CREP.APMR) ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-AACK
19	W-Notify	APMR_A_Data.ind(CREP.APMR, Data) /Data = Alarm-Notification-PDU => APMR_ACK.req(CREP.APMR) 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)	W-AACK
20	W-Notify	APMR_Error.ind(CREP.APMR, ERRCLS, ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-Notify
21	W-Notify	APMS_A_Data.cnf(+)(CREP) => ignore	W-Notify
22	W-Notify	APMS_A_Data.cnf(-) (CREP,ERRCLS,ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-Notify
23	W-Notify	APMS_Error.ind(CREP.APMS, ERRCLS, ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-Notify
24	W-AACK	APMR_Ack.cnf(+)(CREP.APMR) =>	W-User-Ack

#	Current State	Event /Condition =>Action	Next State
25	W-AACK	APMR_Ack.cnf(-)(CREP.APMR, ERRCLS, ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-User-Ack
26	W-CLOSE-APMR	APMR_Close.cnf(+)(CREP.APMR) => APMS_Close.req(CREP.APMS)	W-CLOSE-APMS
27	W-CLOSE-APMR	APMR_Close.cnf(-)(CREP.APMR, ERRCLS, ERRCODE) => APMS_Close.req(CREP.APMS) ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-CLOSE-APMS
28	W-CLOSE-APMS	APMS_Close.cnf(+)(CREP.APMS) => ALPMR_Close.cnf(+)(CREP)	W-START
29	W-CLOSE-APMS	APMS_Close.cnf(-)(CREP.APMS, ERRCLS, ERRCODE) => ALPMR_Close.cnf(+)(CREP) ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-START
30	W-User-Ack	ALPMR_Activate.req(CREP, DA, SA, VLAN, RTATimeoutFactor, MRetry) => ERRCLS:= PROTOCOL ERRCODE:= WRONG-STATE ALPMR_Activate.cnf(-)(CREP, ERRCLS, ERRCODE)	W-User-Ack
31	W-User-Ack	ALPMR_Alarm_Ack.req(CREP, Alarm_Type, Slot_Number, Subslot_Number, Alarm_Specifier, Sequence_Number, PNIO_Status) => Data := Alarm Ack PDU APMS_A_Data.req(CREP.APMS, Data)	W-Notify
32	W-User-Ack	ALPMR_Close.req(CREP) => APMR_Close.req(CREP.APMR)	W-CLOSE-APMR
33	W-User-Ack	APMR_A_Data.ind(CREP.APMR, Data) => ignore	W-User-Ack
34	W-User-Ack	APMR_Error.ind(CREP.APMR, ERRCLS, ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-Notify
35	W-User-Ack	APMS_A_Data.cnf(+)(CREP) => ignore	W-User-Ack
36	W-User-Ack	APMS_A_Data.cnf(-) (CREP,ERRCLS,ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-User-Ack
37	W-User-Ack	APMS_Error.ind(CREP.APMS, ERRCLS, ERRCODE) => ALPMR_Error.ind(CREP, ERRCLS, ERRCODE)	W-User-Ack

5.6.4.4 Functions

There are no functions defined for ALPMR.

5.6.5 APMS

5.6.5.1 Primitive definitions

5.6.5.1.1 Primitives exchanged between APMS and ALPMI or ALPMR

Table 322 shows the service primitives including their associated parameters issued by ALPMI or ALPMR and received by APMS.

Table 322 — Primitives issued by ALPMI/ALPMR to APMS

Primitive name	Source	Associated parameters	Functions
Activate.req	ALPM	CREP DA SA FrameId Prio VLAN RTATimeoutFactor MRetry	The service Activate initializes the ALPMS and loads the schedule to the DMPM.
Close.req	ALPM	CREP	The service Close deinitializes the APMS and stops the transmission of data. The schedule is cleared.
A_Data.req	ALPM	CREP A_Data	The service A_Data conveys acyclic data.

Table 323 shows the service primitives including their associated parameters issued by APMS and received by the ALPMI or ALPMR.

Table 323 — Primitives issued by APMS to ALPMI/ALPMR

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	APMS	CREP	This service primitive indicates that the Activate service succeeded
Activate.cnf(-)	APMS	CREP ERRCLS ERRCODE	This service primitive indicates that the Activate service failed
Close.cnf(+)	APMS	CREP	This service primitive indicates that the Close service succeeded
A_Data.cnf(+)	APMS	CREP	This service primitive indicates that the A_Data service and therefore the conveyance of acyclic data succeeded.
A_Data.cnf(-)	APMS	CREP ERRCLS ERRCODE	This service primitive indicates that the A_Data service failed.
Error.ind	APMS	CREP ERRCLS ERRCODE	This service primitive indicates a failure during transmission of data.

5.6.5.1.2 Primitives exchanged between APMS and LMPM

Table 324 shows the service primitives including their associated parameters issued by LMPM and received by APMS.

Table 324 — Primitives issued by LMPM to APMS

Primitive name	Source	Associated parameters	Functions
LMPM_Time_Event.ind	LMPM	CREP	

Table 325 shows the service primitives including their associated parameters issued by APMS and received by LMPM.

Table 325 — Primitives issued by APMS to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_Schedule_add.req	APMS	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	APMS	CREP	

5.6.5.1.3 Parameter of APMS primitives

The parameters used with the primitives exchanged between the FSPM and the CPM are described in the Service Definition.

5.6.5.2 State machine description

The CLOSED state indicates that an initialisation is needed. Activate service sets the machine in the OPEN state waiting for an A-Data service request. After issuing the transmission of data the WACK state is used to wait for an acknowledge. Receiving the acknowledge will set back the machine to the OPEN state. A Close service request resets the protocol machine to the CLOSED state.

Local variables of the APMS

Seq_Count (Unsigned 16)

This local variable contains the counter value which shall be used at the conveyance of the next Data-RTA-PDU.

Seq_CountO (Unsigned 16)

This local variable contains the previous counter value of the Seq_Count variable.

Retry (Unsigned8)

This local variable is loaded with MRetry and decremented at every retransmission of a frame. A value of zero indicates the failure in that transaction. A new Activate service is needed to re-establish the communication between the peer providers.

TimeAct (Boolean)

This local variable will be used to find the first full timer cycle after transmitting or receiving a PDU.

5.6.5.3 APMS state table

Table 326 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.

Table 326 — APMS state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	APMS_Activate.req(CREP,DA,SA, FrameID,Prio, VLAN, RTATimeoutFactor, MRetry) => Seq_Count:=0xffff Seq_CountO:=0xfffe Store DA,SA,FrameID,Prio, VLAN, MRetry LMPM_Schedule_add.req (CREP,ReductionRatio:=RTATimeoutFactor, Phase:=NIL,Sequence:=0) APMS_Activate.cnf (+)(CREP)	OPEN
2	CLOSED	APMS_Close.req(CREP) => APMS_Close.cnf (+)(CREP)	CLOSED
3	CLOSED	APMS_A_Data.req(CREP, Data) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMS_A_Data.cnf(-)(CREP,ERRCLS,ERRCODE)	CLOSED
4	CLOSED	A_Data_ind => ignore	CLOSED
5	CLOSED	A_Data_cnf /LMPM_status != IV/LS => ignore	CLOSED

#	Current State	Event /Condition =>Action	Next State
6	CLOSED	A_Data_cnf /LMPM_status == IV/LS => ERRCLS := PROTOCOL ERRCODE := LMPM APMS_Error.ind(CREP,ERRCLS,ERRCODE)	CLOSED
7	OPEN	APMS_Activate.req(CREP,DA,SA, FrameID,Prio, VLAN, RTATimeoutFactor, MRetry) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMS_Activate.cnf (-)(CREP,ERRCLS,ERRCODE)	OPEN
8	OPEN	APMS_Close.req(CREP) => A_SDU.Version:=1 A_SDU.PDUType:=ERR A_SDU.Flags:=0 A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_Count A_SDU.AckSeqNum:=Seq_CountO of Server A_SDU.ERRCLS := PROTOCOL A_SDU.ERRCODE := ABORT A_Data_req LMPM_Schedule_remove.req(CREP) APMS_Close.cnf (+)(CREP)	CLOSED
9	OPEN	APMS_A_Data.req(CREP, Data) => A_SDU.Version:=1 A_SDU.PDUType:=DATA A_SDU.Flags:=Tack A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_Count A_SDU.AckSeqNum:=Seq_CountO of Server A_SDU.SDU:=Data Store A_SDU.Flags,SDU Retry:= M_Retry Tim_Act:= FALSE A_Data_req	WACK
10	OPEN	A_Data_ind /A_SDU.PDUType == ACK && A_SDU.AckSeqNum == Seq_CountO => ignore	OPEN
11	OPEN	A_Data_ind /A_SDU.PDUType == ACK && A_SDU.AckSeqNum != Seq_CountO => ERRCLS := PROTOCOL ERRCODE := SEQNUM APMS_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
12	OPEN	A_Data_ind /A_SDU.PDUType == NAK && A_SDU.AckSeqNum == Seq_CountO => ignore	OPEN

#	Current State	Event /Condition =>Action	Next State
13	OPEN	A_Data_ind /A_SDU.PDUType == NAK && A_SDU.AckSeqNum != Seq_CountO => A_SDU.Version:=1 A_SDU.PDUType:=ERR A_SDU.Flags:=0 A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_Count A_SDU.AckSeqNum:=Seq_CountO of Server A_SDU.ERRCLS := PROTOCOL A_SDU.ERRCODE := SEQNUM A_Data_req APMS_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
14	OPEN	A_Data_cnf /LMPM_status != IV/LS => ignore	OPEN
15	OPEN	A_Data_cnf /LMPM_status == IV/LS => ERRCLS := PROTOCOL ERRCODE := LMPM APMS_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
16	OPEN	LMPM_Time_Event.ind(CREP) => ignore	OPEN
17	WACK	APMS_Activate.req(CREP,DA,SA, FrameID,Prio, VLAN, RTATimeoutFactor, MRetry) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMS_Activate.cnf (-)(CREP,ERRCLS,ERRCODE)	WACK
18	WACK	APMS_Close.req(CREP) => A_SDU.Version:=1 A_SDU.PDUType:=ERR A_SDU.Flags:=0 A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_Count A_SDU.AckSeqNum:=Seq_CountO of Server A_SDU.ERRCLS := PROTOCOL A_SDU.ERRCODE := ABORT A_Data_req LMPM_Schedule_remove.req(CREP) APMS_Close.cnf (+)(CREP)	CLOSED
19	WACK	APMS_A_Data.req(CREP, Data) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMS_A_Data.cnf(-)(CREP,ERRCLS,ERRCODE)	WACK
20	WACK	A_Data_ind /A_SDU.PDUType == ACK && A_SDU.AckSeqNum == Seq_Count => Seq_CountO=Seq_Count Seq_Count:= (Seq_Count+1) & 0x7fff APMS_A_Data.cnf(+)(CREP)	OPEN

#	Current State	Event /Condition =>Action	Next State
21	WACK	A_Data_ind /A_SDU.PDUType == ACK && A_SDU.AckSeqNum == Seq_CountO => ignore	WACK
22	WACK	A_Data_ind /A_SDU.PDUType == ACK && A_SDU.AckSeqNum != Seq_Count && A_SDU.AckSeqNum != Seq_CountO => ERRCLS := PROTOCOL ERRCODE := SEQNUM APMS_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
23	WACK	A_Data_ind /A_SDU.PDUType == NAK && A_SDU.AckSeqNum == Seq_Count A_SDU.AckSeqNum == Seq_CountO => ignore	WACK
24	WACK	A_Data_ind /A_SDU.PDUType == NAK && A_SDU.AckSeqNum != Seq_Count && A_SDU.AckSeqNum != Seq_CountO => A_SDU.Version:=1 A_SDU.PDUType:=ERR A_SDU.Flags:=0 A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_Count A_SDU.AckSeqNum:=Seq_CountO of Server A_SDU.ERRCLS := PROTOCOL A_SDU.ERRCODE := SEQNUM A_Data_req APMS_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
25	WACK	A_Data_cnf /LMPM_status != IV/LS => ignore	WACK
26	WACK	A_Data_cnf /LMPM_status == IV/LS => ERRCLS := PROTOCOL ERRCODE := LMPM APMS_Error.ind(CREP,ERRCLS,ERRCODE)	WACK
27	WACK	LMPM_Time_Event.ind(CREP) /!Tim_Act => Tim_Act:=TRUE	WACK
28	WACK	LMPM_Time_Event.ind(CREP) /Tim_Act && Retry != 0 => A_SDU.Version:=1 A_SDU.PDUType:=DATA A_SDU.Flags:=from Stored A_SDU.Flags A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_Count A_SDU.AckSeqNum:=Seq_CountO of Server A_SDU.SDU:=from Stored A_SDU.SDU Retry:= Retry-1 Tim_Act:= FALSE A_Data_req	WACK

#	Current State	Event /Condition =>Action	Next State
29	WACK	LMPM_Time_Event.ind(CREP) /Tim_Act && Retry = 0 => ERRCLS := PROTOCOL ERRCODE := TIMEOUT APMS_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN

5.6.5.4 Functions

Table 327 contains the functions or macros used by the APMS and APMR, their arguments and their descriptions.

Table 327 — Functions used by the APMS and APMR

Name	Function
A_Data_ind	if (txOption == RTC) LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) if (txOption == UDP) UDP_A_Data.ind (IPPort, CREP, DA, SA, Prio, A_SDU)
A_Data_cnf	if (txOption == RTC) LMPM_A_Data.cnf (CREP, LMPM_status) if (txOption == UDP) UDP_A_Data.cnf (IPPort, CREP, LMPM_status)
A_Data_req	if (txOption == RTC) LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU) if (txOption == UDP) UDP_A_Data.req (IPPort, CREP, DA, SA, Prio, A_SDU)

5.6.6 APMR

5.6.6.1 Primitive definitions

5.6.6.1.1 Primitives exchanged between APMR and ALPMI or ALPMR

Table 328 shows the service primitives including their associated parameters issued by ALPMI or ALPMR and received by APMR.

Table 328 — Primitives issued by ALPMI/ALPMR to APMR

Primitive name	Source	Associated parameters	Functions
Activate.req	ALPMI/ ALPMR	CREP DA SA Frameld Prio VLAN MRetry	The service Activate initializes the ALPMR.
Close.req	ALPMI/ ALPMR	CREP	The service Close closes the ALPMR.
ACK.req	ALPMI/ ALPMR	CREP	The service ACK sends the ACK-RTA-PDU to acknowledge.

Table 329 shows the service primitives including their associated parameters issued by APMR and received by the ALPMI or ALPMR.

Table 329 — Primitives issued by APMR to ALPMI/ALPMR

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	APMR	CREP	This service primitive indicates that the Activate service succeeded.
Activate.cnf(-)	APMR	CREP ERRCLS ERRCODE	This service primitive indicates that the Activate service failed.
Close.cnf(+)	APMR	CREP	This service primitive indicates that the Close service succeeded.
ACK.cnf(+)	APMR	CREP	This service primitive indicates that the ACK service succeeded.
ACK.cnf(-)	APMR	CREP ERRCLS ERRCODE	This service primitive indicates that the ACK service failed.

Primitive name	Source	Associated parameters	Functions
A_Data.ind	APMR	CREP Data	This service primitive indicates a reception of a Data-RTA- PDU.
Error.ind	APMR	CREP ERRCLS ERRCODE	This service primitive indicates a transmission or protocol error.

5.6.6.1.2 Parameter of APMR primitives

The parameters used with the primitives exchanged between the FSPM and the CPM are described in the Service Definition.

5.6.6.2 State machine description

The CLOSED state indicates that an initialisation is needed. The Activate service sets the machine in the OPEN state waiting for LMPM_A_Data service indication. After passing a Data indication to the user the WACK state is used to wait for an acknowledge from the APMR user. Receiving the acknowledge will set back the machine to the OPEN state. An Error or a Close service request sets the protocol machine to the CLOSED state.

Local variables of the APMR

Seq_Count (Unsigned 16)

This local variable contains the counter value which shall be used at the receipt of the next Data-RTA-PDU.

Seq_CountO (Unsigned 16)

This local variable contains the previous counter value of the Seq_Count variable.

5.6.6.3 APMR state table

Table 330 contains the complete description of the APMR state machine. A LMPM_A_Data.ind primitive will be accepted if PDUType field is Data-RTA-PDU and the PDUType.Version field is 0.

Table 330 — APMR state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	APMR_Activate.req(CREP,DA,SA,FrameId,Prio, VLAN, MRetry) => Seq_Count:=0xffff Seq_CountO:=0xfffe Store DA,SA,FrameId,Prio, VLAN, MRetry APMR_Activate.cnf (+)(CREP)	OPEN
2	CLOSED	APMR_Close.req(CREP) => APMR_Close.cnf (+)(CREP)	CLOSED
3	CLOSED	APMR_ACK.req(CREP) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMR_ACK.cnf(-) (CREP,ERRCLS,ERRCODE)	CLOSED
4	CLOSED	A_Data_ind => ignore	CLOSED
5	CLOSED	A_Data_cnf /LMPM_status != IV/LS => ignore	CLOSED

#	Current State	Event /Condition =>Action	Next State
6	CLOSED	A_Data_cnf /LMPM_status == IV/LS => ERRCLS := PROTOCOL ERRCODE := LMPM APMR_Error.ind(CREP,ERRCLS,ERRCODE)	CLOSED
7	OPEN	APMR_Activate.req(CREP,DA,SA,FrameId,Prio, VLAN, MRetry) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMR_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	OPEN
8	OPEN	APMR_Close.req(CREP) => Seq_CountO :=Seq_Count A_SDU.Version :=0 A_SDU.PDUType := ERR A_SDU.Flags :=0 A_SDU.Window :=1 A_SDU.SendSeqNum :=Seq_CountO of Sender A_SDU.AckSeqNum :=Seq_CountO A_SDU.ERRCLS := PROTOCOL A_SDU.ERRCODE := ABORT A_Data_req APMR_Close.cnf (+)(CREP)	CLOSED
9	OPEN	APMR_ACK.req(CREP) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMR_ACK.cnf(-) (CREP,ERRCLS,ERRCODE)	OPEN
10	OPEN	A_Data_ind /A_SDU.Type != DATA !A_SDU.AddFlags.Tack => ignore	OPEN
11	OPEN	A_Data_ind /A_SDU.Type = DATA && A_SDU.AddFlags.Tack && A_SDU.SendSeqNum = Seq_Count => Data := A_SDU. APMR_A_Data.ind(CREP, Data)	WACK
12	OPEN	A_Data_ind /A_SDU.Type = DATA && A_SDU.AddFlags.Tack && A_SDU.SendSeqNum = Seq_CountO => A_SDU.Version:=1 A_SDU.Type:=ACK A_SDU.Flags:=0 A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_CountO of Sender A_SDU.AckSeqNum:=Seq_CountO A_Data_req	OPEN

#	Current State	Event /Condition =>Action	Next State
13	OPEN	A_Data_ind /A_SDU.Type = DATA && A_SDU.AddFlags.Tack && A_SDU.SendSeqNum! = Seq_Count && A_SDU.SendSeqNum != Seq_CountO => ERRCLS := PROTOCOL ERRCODE := SEQNUM A_SDU.Version:=1 A_SDU.Type:=NAK A_SDU.Flags:=0 A_SDU.Window:=1 A_SDU.SendSeqNum:=Seq_CountO of Sender A_SDU.AckSeqNum:=Seq_CountO APMR_Error.ind(CREP,ERRCLS,ERRCODE) A_Data_req	OPEN
14	OPEN	A_Data_cnf /LMPM_status != IV/LS => ignore	OPEN
15	OPEN	A_Data_cnf /LMPM_status == IV/LS => ERRCLS := PROTOCOL ERRCODE := LMPM APMR_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
16	WACK	APMR_Activate.req(CREP,DA,SA,Frameld,Prio, VLAN, MRetry) => ERRCLS := CTXT ERRCODE := INVALID_STATE APMR_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	WACK
17	WACK	APMR_Close.req(CREP) => Seq_CountO :=Seq_Count A_SDU.Version :=0 A_SDU.Type := ERR A_SDU.Flags :=0 A_SDU.Window :=1 A_SDU.SendSeqNum :=Seq_CountO of Sender A_SDU.AckSeqNum :=Seq_CountO A_SDU.ERRCLS := PROTOCOL A_SDU.ERRCODE := ABORT A_Data_req APMR_Close.cnf (+)(CREP)	CLOSED
18	WACK	APMR_ACK.req(CREP) => Seq_CountO :=Seq_Count Seq_Count := (Seq_Count+1) & 0x7fff A_SDU.Version :=1 A_SDU.Type :=ACK A_SDU.Flags :=0 A_SDU.Window :=1 A_SDU.SendSeqNum :=Seq_CountO of Sender A_SDU.AckSeqNum :=Seq_CountO A_Data_req APMR_ACK.cnf(+)(CREP)	OPEN
19	OPEN	A_Data_ind /A_SDU.Type != ERR => APMR_Error.ind(CREP,ERRCLS,ERRCODE)	OPEN
20	WACK	A_Data_ind /A_SDU.Type != DATA !A_SDU.AddFlags.Tack =>	OPEN

#	Current State	Event /Condition =>Action	Next State
21	WACK	A_Data_ind /A_SDU.Type = DATA && A_SDU.AddFlags.Tack && A_SDU.SendSeqNum = Seq_Count => ignore	WACK
22	WACK	A_Data_ind /A_SDU.Type = DATA && A_SDU.AddFlags.Tack && A_SDU.SendSeqNum != Seq_Count => ERRCLS := PROTOCOL ERRCODE := SEQNUM APMR_Error.ind(CREP,ERRCLS,ERRCODE)	WACK
23	WACK	A_Data_ind /A_SDU.Type != ERR => APMR_Error.ind(CREP,ERRCLS,ERRCODE)	WACK
24	WACK	A_Data_cnf /LMPM_status != IV/LS => ignore	WACK
25	WACK	A_Data_cnf /LMPM_status == IV/LS => ERRCLS := PROTOCOL ERRCODE := LMPM APMR_Error.ind(CREP,ERRCLS,ERRCODE)	WACK

5.6.6.4 Functions

Table 327 contains the functions or macros used by the APMS and APMR, their arguments and their descriptions.

5.6.7 NRPM

5.6.7.1 Primitive definitions

5.6.7.1.1 Primitives exchanged between NRPM and CMCTL

Table 331 shows the service primitives including their associated parameters issued by CMCTL and received by NRPM.

Table 331 — Primitives issued by CMCTL to NRPM

Primitive name	Source	Associated parameters	Functions
Init.req	CMCTL	Add_resolution Dev_Name Dev_IP_Para DCP_Para	The service Init initializes the NRPM and starts the name resolution process.
RM_Read.req	CMCTL	AREP API TargetUUID SlotNumber SubslotNumber Index SeqNumber Length	The service RM_Read reads data.
RM_Write.req	CMCTL	AREP SlotNumber SubslotNumber Index SeqNumber Length Data	The service RM_Write reads data.

Primitive name	Source	Associated parameters	Functions
RM_Connect.req	CMCTL	AREP ARBlockReq ListOfIOCRBlockReq AlarmCRBlockReq ListOfExpectedSubmodul BlockReq	The service RM_Connect initiates a connection.
RM_Release.req	CMCTL	AREP ControlBlock	The service RM_Release terminates a connection.
RM_Cancel.req	CMCTL	AREP ControlBlock	The service RM_Cancel terminates a connection.
RM_Abort.req	CMCTL	AREP	The service RM_Abort terminates a connection.
RM_CControl.rsp(+)	CMCTL	AREP ControlBlock	The service RM_CControl responds to application ready.
RM_CControl.rsp(-)	CMCTL	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	The service RM_CControl responds to application ready.
RM_DControl.req	CMCTL	AREP ControlBlock	The service RM_DControl signals end of parameter.
RM_Abort.req	CMCTL	AREP	The service RM_Abort terminates a connection.

Table 332 shows the service primitives including their associated parameters issued by NRPM and received by the CMCTL.

Table 332 — Primitives issued by NRPM to CMCTL

Primitive name	Source	Associated parameters	Functions
Init.cnf(+)	NRPM	Dev_MAC_Addr	This service primitive indicates that the Activate service succeeded and delivers the MAC Address of the IO device.
Init.cnf(-)	NRPM	Err	This service primitive indicates that the address and name resolution service failed. The Err parameter contains the reason for the failure (e.g. no device, multiple devices)
CMCTL_Init.req	NRPM	AREP	In combination with the successful initialization this service is used to initialize an AREP for the IO device within the CMCTL.
RM_Abort.cnf	NRPM	AREP	This service primitive indicates that the Abort service succeeded.
RM_CControl.ind	NRPM	AREP ControlBlockConnect	This service primitive indicates application ready.
RM_Connect.cnf(-)	NRPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive indicates that the RM_Connect service failed.
RM_Connect.cnf(+)	NRPM	AREP ARBlockRes ListOfIOCRBlockRes AlarmCRBlockRes ModuleDiffBlock	This service primitive indicates that the RM_Connect service succeeded.
RM_DControl.cnf(-)	NRPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive indicates that the RM_DControl service failed.

Primitive name	Source	Associated parameters	Functions
RM_DControl.cnf(+)	NRPM	AREP ControlBlock	This service primitive indicates that the RM_DControl service succeeded.
RM_Read.cnf(-)	NRPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive indicates that the RM_Read service failed.
RM_Read.cnf(+)	NRPM	AREP Seq Number AddData1 AddData2	This service primitive indicates that the RM_Read service succeeded.
RM_Realease.cnf(-)	NRPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive indicates that the RM_Realease service failed.
RM_Realease.cnf(+)	NRPM	AREP ControlBlock	This service primitive indicates that the RM_Release service succeeded.
RM_Write.cnf(-)	NRPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive indicates that the RM_Write service failed.
RM_Write.cnf(+)	NRPM	AREP Seq Number AddData1 AddData2	This service primitive indicates that the RM_Write service succeeded.

5.6.7.1.2 Primitives exchanged between NRPM and other machines

Table 333 shows the service primitives including their associated parameters issued by other machines and received by NRPM.

Table 333 — Primitives issued by other machines to NRPM

Primitive name	Source	Associated parameters	Functions
ARP_getMACQ.ind	ARP	MAC_Address IP_Address	
DCP_Identify.cnf(-)	DCP	CREP ERRCLS ERRCODE	
DCP_Identify.cnf(+)	DCP	CREP	
DCP_IdentifyQ.ind	DCP	CREP SA ListOfData	
DCP_SetValue.cnf(-)	DCP	CREP ERRCLS ERRCODE	
DCP_SetValue.cnf(+)	DCP	CREP DA ListOfData	
DCPMCS_Activate.cnf (+)	DCPMCS	CREP	
DCPMCS_Close.cnf (-)	DCPMCS	CREP ERRCLS ERRCODE	
DCPMCS_Close.cnf (+)	DCPMCS	CREP	
DNS_GetHostByName.ind	DNS	IP_Add	

Table 334 shows the service primitives including their associated parameters issued by NRPM and received by other machines.

Table 334 — Primitives issued by NRPM to other machines

Primitive name	Source	Associated parameters	Functions
ARP_getMAC.req	NRPM	IP_Address	
DCPMCS_Activate.req	NRPM	CREP VLAN-Prio VLAN-ID DCPMC_Timeout	
DCP_IdentifyF.req	NRPM	CREP, ListofFilter, ResponseDelay	
DCP_Set.req	NRPM	CREP DA DataSet Dev_IP_Para	
DCPMCS_Close.req	NRPM	CREP	
DCPUCS_Activate.req	NRPM	CREP VLAN-Prio VLAN-ID DCPUC_Timeout DCPUC_Retry	
DCPUCS_Close.req	NRPM	CREP	
DNS_GetHostByName.req	NRPM	Dev_Name	

5.6.7.1.3 Parameters of NRPM primitives

The parameters used with the primitives exchanged between the FSPM and the CPM are described in the Service Definition.

5.6.7.2 State machine description

The name resolution protocol machine is present for each device (AR) of an IO controller. It is used for name and address resolution of an IO device.

5.6.7.3 NRPM state table

Table 335 contains the complete description of the NRPM state machine.

Table 335 — NRPM state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	NRPM_Init.req(Add_resolution, Dev_Name, Dev_IP_Para,DCP_Para) => Store Parameters DCPMCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPMC_Timeout)	ACT1
2	ACT1	DCPUCS_Activate.cnf (-) (CREP, ERRCLS, ERRCODE) => NRPM_Init.cnf(-) (Err=No_DCPact) DCPMCS_Close.req(CREP)	POWER-ON
3	ACT1	DCPMCS_Activate.cnf (+)(CREP) => DCPUCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPUC_Timeout, DCPUC_Retry)	ACT2
4	ACT2	DCPUCS_Activate.cnf (-) (CREP, ERRCLS, ERRCODE) => NRPM_Init.cnf(-) (Err=No_DCPact) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
5	ACT2	DCPUCS_Activate.cnf (+)(CREP) =>	W_DCP_I

#	Current State	Event /Condition =>Action	Next State
6	DCP_I	/Add_resolution == DNS => Start Timer(DCP_Short_Timer) DNS_GetHostByName.req(Dev_Name)	W_DNS_I
7	DCP_I	/Add_resolution == DCP => ListofFilter := Filter with Dev_Name ResponseDelay := 1 DCP_IdentifyF.req(CREP, ListofFilter, ResponseDelay)	W_DCP_I
8	DCP_I	/Add_resolution == none => IP_Addr := IP_Add ARP_getMAC.req (IP_Address=Dev_IP_Para.IP)	W_ARP
9	W_DNS_I	DNS_GetHostByName.ind(IP_Add) => IP_Addr := IP_Add ARP_getMAC.req (IP_Address=Dev_IP_Para.IP)	W_ARP
10	W_DNS_I	Time_out => NRPM_Init.cnf(-) (Err=No_Dev_Name) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
11	W_DCP_I	DCP_IdentifyQ.ind(CREP,SA,ListOfData) => IP_Addr := ListOfData.IP MAC_Addr := SA	W_DCP_T
12	W_DCP_I	DCP_Identify.cnf(+)(CREP) => NRPM_Init.cnf(-) (Err=No_Dev_Name) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
13	W_DCP_I	DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE) => NRPM_Init.cnf(-) (Err=No_Dev_Name) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
14	W_DCP_T	DCP_IdentifyQ.ind(CREP,SA,ListOfData) => NRPM_Init.cnf(-) (Err=Multiple_Dev_Name) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
15	W_DCP_T	DCP_Identify.cnf(+)(CREP) => Start Timer(ARP_Timer) ARP_getMAC.req (IP_Address=Dev_IP_Para.IP)	W_ARP_I
16	W_DCP_T	DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE) => NRPM_Init.cnf(-) (Err=No_Dev_Name) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
17	W_ARP_I	ARP_getMACQ.ind(MAC_Address, IP_Address) /IP_Address == Dev_IP_Para.IP =>	W_ARP_T

#	Current State	Event /Condition =>Action	Next State
18	W_ARP_I	ARP_getMACQ.ind(MAC_Address, IP_Address) /IP_Address != Dev_IP_Para.IP => ignore	W_ARP_I
19	W_ARP_I	Time_out => MAC_Address := MAC_Addr DCP_Set.req(CREP, DA=MAC_Address, DataSet=IP, Dev_IP_Para)	W_SET_IP
20	W_ARP_T	ARP_getMACQ.ind(MAC_Address, IP_Address) /IP_Address == Dev_IP_Para.IP => NRPM_Init.cnf(-) (Err=No_IP_Addr) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
21	W_ARP_T	ARP_getMACQ.ind(MAC_Address, IP_Address) /IP_Address != Dev_IP_Para.IP => ignore	W_ARP_T
22	W_ARP_T	Time_out => Dev_MAC_Addr := MAC_Addr NRPM_Init.cnf(+) (Dev_MAC_Addr) CMCTL_Init.req(AREP) RPC_Alloc.req	OPEN
23	W_SET_IP	DCP_SetValue.cnf(+)(CREP,DA,ListOfData) => Dev_MAC_Addr := MAC_Addr NRPM_Init.cnf(+) (Dev_MAC_Addr) CMCTL_Init.req(AREP) RPC_Alloc.req	OPEN
24	W_SET_IP	DCP_SetValue.cnf(-) (CREP,DA,ERRCLS,ERRCODE) => NRPM_Init.cnf(-) (Err=DCP_Error) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
25	W_ARP	ARP_getMACQ.ind(MAC_Address, IP_Address) /IP_Address == Dev_IP_Para.IP => NRPM_Init.cnf(+) (Dev_MAC_Addr) CMCTL_Init.req(AREP) RPC_Alloc.req	OPEN
26	W_ARP	ARP_getMACQ.ind(MAC_Address, IP_Address) /IP_Address != Dev_IP_Para.IP => NRPM_Init.cnf(-) (Err=No_IP_Addr)	W_ARP
27	W_ARP	Time_out => NRPM_Init.cnf(-) (Err=No_IP_Addr) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
28	OPEN	RM_Abort.req (AREP) => RM_Abort.cnf (AREP) DCPMCS_Close.req(CREP) DCPUCS_Close.req(CREP)	POWER-ON
29	OPEN	RPC_CControl.ind(Arg) /VALID_Block_Structure => AREP := Arg.AREP RM_CControl.ind (AREP, ControlBlock, ModuleDiffBlock)	OPEN

#	Current State	Event /Condition =>Action	Next State
30	OPEN	RPC_CControl.ind(Arg) /!VALID_Block_Structure => Arg.ErrorCode2 := WRONG_BLOCK RPC_CControl.rsp(-)(Arg)	OPEN
31	OPEN	RM_CControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => encode Error(Only NDR-Header) RPC_CControl.rsp(-)(Arg)	OPEN
32	OPEN	RM_CControl.rsp(+)(AREP, ControlBlock) => RPC_CControl.rsp(+)(Arg)	OPEN
33	OPEN	RM_DControl.req (AREP, ControlBlock) => RPC_DControl.req(Arg)	OPEN
34	OPEN	RPC_DControl.cnf(-)(Arg) => RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
35	OPEN	RPC_DControl.cnf(+)(Arg) /!VALID_Block_Structure => AREP := Arg.AREP RM_DControl.cnf(+)(AREP, ControlBlock)	OPEN
36	OPEN	RPC_DControl.cnf(+)(Arg) /!VALID_Block_Structure => ErrorCode2 := WRONG_BLOCK RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
37	OPEN	RM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => RPC_Write.req(Arg)	OPEN
38	OPEN	RPC_Write.cnf(-)(Arg) => RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
39	OPEN	RPC_Write.cnf(+)(Arg) /!VALID_Block_Structure => AREP := Arg.AREP RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	OPEN
40	OPEN	RPC_Write.cnf(+)(Arg) /!VALID_Block_Structure => ErrorCode2 := WRONG_BLOCK RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
41	OPEN	RM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => RPC_Read.req(Arg)	OPEN
42	OPEN	RPC_Read.cnf(-)(Arg) => RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
43	OPEN	RPC_Read.cnf(+)(Arg) /!VALID_Block_Structure => AREP := Arg.AREP RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2)	OPEN

#	Current State	Event /Condition =>Action	Next State
44	OPEN	RPC_Read.cnf(+)(Arg) /!VALID_Block_Structure => ErrorCode2 := WRONG_BLOCK RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
45	OPEN	RM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => RPC_Connect.req(Arg)	OPEN
46	OPEN	RPC_Connect.cnf(-)(Arg) => RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
47	OPEN	RPC_Connect.cnf(+)(Arg) /!VALID_Block_Structure => AREP := Arg.AREP RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	OPEN
48	OPEN	RPC_Connect.cnf(+)(Arg) /!VALID_Block_Structure => ErrorCode2 := WRONG_BLOCK RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
49	OPEN	RM_Release.req(AREP, ControlBlock) => RPC_Release.req(Arg)	OPEN
50	OPEN	RPC_Release.cnf(-)(Arg) => RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
51	OPEN	RPC_Release.cnf(+)(Arg) /!VALID_Block_Structure => AREP := Arg.AREP RM_Release.cnf(+)(AREP, ControlBlock)	OPEN
52	OPEN	RPC_Release.cnf(+)(Arg) /!VALID_Block_Structure => ErrorCode2 := WRONG_BLOCK RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	OPEN
53	any state	DCPUCS_Close.cnf (-) (CREP, ERRCLS, ERRCODE) =>	same state
54	any state	DCPUCS_Close.cnf (+)(CREP) =>	same state
55	any state	DCPMCS_Close.cnf (-) (CREP, ERRCLS, ERRCODE) =>	same state
56	any state	DCPMCS_Close.cnf (+)(CREP) =>	same state

5.6.7.4 Functions

The table below contains the functions or macros used by the NRPM and RMPM, their arguments and their descriptions.

Table 336 — Functions used by the NRPM and RMPM

Name	Function
CHECK_AR_TYPE	check all Data of CM_Connect-REQ-PDU check reduction ratio and phase check against attributes of IO Data ASE, Alarm ASE check data volume
CHECK_IDENT_NUMBER	check Ident
VALID_SELECTORS	ListOfSelector encoding and contents comply with DCP specification
VALID_FILTER	ListOfFilter encoding and contents comply with DCP specification
VALID_DATA	ListOfData encoding and contents comply with DCP specification
VALID_Block_Structure	Block Structure of RPC service does not comply with coding rules
NAME_IP_Change	NAME in ListOfData != Dev_Name IP in ListOfData != Dev_IP_Para
Search_in_List_of_RM_Entries	Scan List of RM Entries for a specific AREP
Store Data in Data Set	Store ListOfData Parameter in DataSet of the station
X in ListOfData	Search a Element in ListOfData with a

5.6.8 RMPM

5.6.8.1 Primitive definitions

5.6.8.1.1 Primitives exchanged between RMPM and CMDEV

Table 337 shows the service primitives including their associated parameters issued by CM of the IO device (CMDEV) and received by RMPM.

Table 337 — Primitives issued by CMDEV to RMPM

Primitive name	Source	Associated parameters	Functions
RM_Abort.req	CMDEV	AREP	The service Abort removes the stored AREP from the list of AREPs.
RM_CControl.req	CMDEV	AREP ControlBlock ModuleDiffBlock	The service CControl conveys the application ready flag.
RM_Connect.rsp(+)	CMDEV	AREP ARBlockRes ListOfIOCRBlockRes AlarmCRBlockRes ModuleDiffBlock	This service primitive is a positive response to establish an application relationship.
RM_Connect.rsp(-)	CMDEV	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative response and the requested application relationship is not established.
RM_DControl.rsp(+)	CMDEV	AREP ControlBlock	This service primitive is a response to end of parameter signal.
RM_DControl.rsp(-)	CMDEV	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	
RM_Read.rsp(+)	CMDEV	AREP SeqNumber AddData1 AddData2	This service primitive is the response to the Read service.
RM_Read.rsp(-)	CMDEV	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	

Primitive name	Source	Associated parameters	Functions
RM_Write.rsp(+)	CMDEV	AREP SeqNumber AddData1 AddData2	This service primitive is the response to the Write service.
RM_Write.rsp(-)	CMDEV	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	
RM_Release.rsp(+)	CMDEV	AREP ControlBlock	
RM_Release.rsp(-)	CMDEV	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	

Table 338 shows the service primitives including their associated parameters issued by RMPM and received by CM of the IO device (CMDEV).

Table 338 — Primitives issued by RMPM to CMDEV

Primitive name	Source	Associated parameters	Functions
RM_Init.cnf(+)	RMPM	-	This service primitive indicates that the Init service succeeded.
RM_Abort.cnf	RMPM	AREP	This service primitive confirms the Abort service.
RM_CControl.cnf(+)	RMPM	AREP ControlBlock	This service primitives confirm the CControl service.
RM_CControl.cnf(-)	RMPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	
RM_Connect.ind	RMPM	AREP ARBlockReq ListOfIOCRBlockReq AlarmCRBlockReq ListOfExpectedSubmoduleBlockReq	This service primitive indicates a request to establish an application relationship.
RM_DControl.ind	RMPM	AREP ControlBlock	This service primitive indicates the end of parameter.
RM_Read.ind	RMPM	AREP API TargetUUID SlotNumber SubslotNumber Index SeqNumber Length	This service primitive indicates the Read service.
RM_Write.ind	RMPM	AREP SlotNumber SubslotNumber Index SeqNumber Length Data	This service primitive indicates the Write service.
RM_Release.ind	RMPM	AREP ControlBlock	This service primitive indicates that the application relationship should be released.

5.6.8.1.2 Primitives exchanged between RMPM and RPC

Table 343 shows the service primitives including their associated parameters issued by RPC and received by RMPM.

Table 339 — Primitives issued by RPC to RMPM

Primitive name	Source	Associated parameters	Functions
RPC_CControl.cnf(+)	RPC	Arg	
RPC_CControl.cnf(-)	RPC	Arg	
RPC_Connect.ind	RPC	Arg	
RPC_DControl.ind	RPC	Arg	
RPC_Read.ind	RPC	Arg	
RPC_Release.ind	RPC	Arg	
RPC_Write.ind	RPC	Arg	

Table 344 shows the service primitives including their associated parameters issued by RMPM and received by RPC.

Table 340 — Primitives issued by RMPM to RPC

Primitive name	Source	Associated parameters	Functions
RPC_CControl.req	RMPM	Arg	
RPC_Connect.rsp(-)	RMPM	Arg	
RPC_Connect.rsp(+)	RMPM	Arg	
RPC_DControl.rsp(-)	RMPM	Arg	
RPC_DControl.rsp(-)	RMPM	Arg	
RPC_DControl.rsp(+)	RMPM	Arg	
RPC_Read.rsp(-)	RMPM	Arg	
RPC_Release.rsp(-)	RMPM	Arg	
RPC_Release.rsp(+)	RMPM	Arg	
RPC_Write.rsp(-)	RMPM	Arg	
RPC_Write.rsp(+)	RMPM	Arg	

5.6.8.1.3 Primitives exchanged between RMPM and other machines

Table 341 shows the service primitives including their associated parameters issued by other machines and received by RMPM.

Table 341 — Primitives issued by other machines to RMPM

Primitive name	Source	Associated parameters	Functions
DHCPOFFER.ind	DHCP	IP_Para ListOfData	
DCP_Get.ind	DCP	CREP SA Selectors	
DCP_Identify.ind	DCP	CREP SA ListofFilters ResponseDelay	
DCP_Set.ind	DCP	CREP SA ListOfData	
DCPMCR_Activate.cnf (+)	DCPMCR	CREP	
DCPMCR_Close.cnf (-)	DCPMCR	CREP ERRCLS ERRCODE	
DCPMCR_Close.cnf (+)	DCPMCR	CREP	
DCPUCR_Activate.cnf (-)	DCPUCR	CREP ERRCLS ERRCODE	
DCPUCR_Activate.cnf (+)	DCPUCR	CREP	
DCPUCR_Close.cnf (-)	DCPUCR	CREP ERRCLS ERRCODE	
DCPUCR_Close.cnf (+)	DCPUCR	CREP	

Primitive name	Source	Associated parameters	Functions
RM_Init.req	FSPM	-	The service Init initializes the RMPM.

Table 342 shows the service primitives including their associated parameters issued by RMPM and received by other machines.

Table 342 — Primitives issued by RMPM to other machines

Primitive name	Source	Associated parameters	Functions
DCPMCR_Activate.req	RMPM	CREP ListOfFilters Send_MC_Response	
DCPMCR_Close.req	RMPM	CREP	
DCP_IdentifyQ.req	RMPM	CREP DA istOfData	
DCP_Get.rsp (+)	RMPM	CREP DA ListOfData	
DCP_Set.rsp (-)	RMPM	CREP DA ERRCLS ERRCODE	
DCP_Set.rsp (+)	RMPM	CREP DA ListOfData	
DHCPDISCOVER.req	RMPM	Options	
RM_Init.cnf	RMPM	-	
RM_Stopped.ind	RMPM	-	This service primitive indicates that a new device name was set and IP stack was therefore stopped to receive new IP parameter.
NRPM_Init.cnf(-)	RMPM	Err	

5.6.8.1.4 Parameters of RMPM primitives

The parameters used with the primitives exchanged between the RMPM and the CPM are described in the Service Definition.

5.6.8.2 State machine description

The Device Resource Management protocol machine is present once per IO Device. It is used for name/address resolution of an IO-Device as well as the allocation of CM Device State Machines to AR.

5.6.8.3 RMPM state table

Table 343 contains the complete description of the RMPM state machine.

Table 343 — RMPM state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	RM_Init.req => Store Parameters DCPMCR_Activate.req(CREP, ListOfFilters, Send_MC_Response)	ACT1
2	ACT1	DCPUCR_Activate.cnf (-) (CREP, ERRCLS, ERRCODE) => NRPM_Init.cnf(-) (Err=No_DCPact) DCPMCR_Close.req(CREP)	POWER-ON
3	ACT1	DCPMCR_Activate.cnf (+)(CREP) => DCPUCR_Activate.req(CREP, Hold_requestor)	ACT2

#	Current State	Event /Condition =>Action	Next State
4	ACT2	DCPUCR_Activate.cnf (-) (CREP, ERRCLS, ERRCODE) => NRPM_Init.cnf(-) (Err=No_DCPact) DCPMCR_Close.req(CREP) DCPUCR_Close.req(CREP)	POWER-ON
5	ACT2	DCPUCR_Activate.cnf (+)(CREP) =>	W_DCP_I
6	SetUp	/Dev_Name == NIL && DHCP_En => Start Timer(DHCP_RetryTime) DHCPDISCOVER.req(Options)	Set_Name
7	SetUp	/Dev_Name == NIL && !DHCP_En =>	Set_Name
8	SetUp	/Dev_Name != NIL && Dev_IP_Para == NIL && DHCP_En => Start Timer(DHCP_RetryTime) DHCPDISCOVER.req(Options)	Set_IP
9	SetUp	/Dev_Name != NIL && Dev_IP_Para == NIL && !DHCP_En =>	Set_IP
10	SetUp	/Dev_Name != NIL && Dev_IP_Para != NIL => Init_List_RM_Entries Init_List_AR_Resc AR_Active := 0 Start IP, UDP, RPC RM_Init.cnf	W_Conect
11	Set_Name	DHCPOFFER.ind(IP_Para, ListOfData) /VALID_DATA && (NAME in ListOfData) && NAME-length != 0 => Dev_IP_Para :=Data Init_List_RM_Entries Init_List_AR_Resc AR_Active := 0 Stop Timer(DHCP_RetryTime) Start IP, UDP, RPC Store Data in DataSet RM_Init.cnf	W_Conect
12	Set_Name	DHCPOFFER.ind(IP_Para, ListOfData) /!(NAME in ListOfData) && NAME-length != 0 => Store Data in DataSet	Set_Name
13	Set_Name	DHCPOFFER.ind(IP_Para, ListOfData) /!VALID_DATA => ignore	Set_Name

#	Current State	Event /Condition =>Action	Next State
14	Set_Name	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && ((NAME in ListOfData) && NAME-length != 0) && ((IP in ListOfData) Dev_IP_Para != NIL) => Init_List_RM_Entries Init_List_AR_Resc AR_Active := 0 Stop Timer(DHCP_RetryTime) Start IP, UDP, RPC Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData) RM_Init.cnf	W_Conect
15	Set_Name	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && ((NAME in ListOfData) && NAME-length != 0) && !(IP in ListOfData) && Dev_IP_Para == NIL && DHCP_En => Start Timer(DHCP_RetryTime) Store Data in DataSet Dev_IP_Para := NIL DCP_Set.rsp (+)(CREP,DA,ListOfData) DHCPDISCOVER.req(Options)	Set_IP
16	Set_Name	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && ((NAME in ListOfData) && NAME-length != 0) && !(IP in ListOfData) && Dev_IP_Para == NIL && !DHCP_En => Store Data in DataSet Dev_IP_Para := NIL DCP_Set.rsp (+)(CREP,DA,ListOfData)	Set_IP
17	Set_Name	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && (!(NAME in ListOfData) NAME-length == 0) => Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData)	Set_Name
18	Set_Name	DCP_Set.ind (CREP, SA, ListOfData) /!VALID_DATA => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Set.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	Set_Name
19	Set_Name	DCP_Identify.ind (CREP, SA, ListofFilters, ResponseDelay) /VALID_FILTERS => Retrieve Data from DataSet DA := SA DCP_IdentifyQ.req (CREP,DA,ListOfData)	Set_Name
20	Set_Name	DCP_Identify.ind (CREP, SA, ListofFilters, ResponseDelay) /!VALID_FILTERS => ignore	Set_Name
21	Set_Name	DCP_Get.ind (CREP, SA, Selectors) /VALID_SELECTORS => Retrieve Data from DataSet DCP_Get.rsp (+)(CREP,DA,ListOfData)	Set_Name
22	Set_Name	DCP_Get.ind (CREP, SA, Selectors) /!VALID_SELECTORS => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Get.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	Set_Name

#	Current State	Event /Condition =>Action	Next State
23	Set_Name	Time out => ignore	Set_Name
24	Set_IP	DHCPOFFER.ind(IP_Para, ListOfData) /VALID_DATA && !((NAME in ListOfData) && NAME-length == 0) => Dev_IP_Para :=Data Init_List_RM_Entries Init_List_AR_Resc AR_Active := 0 Stop Timer(DHCP_RetryTime) Start IP, UDP, RPC Store Data in DataSet RM_Init.cnf	W_Conect
25	Set_IP	DHCPOFFER.ind(IP_Para, ListOfData) /INVALID_DATA ((NAME in ListOfData) && NAME-length == 0) => ignore	Set_IP
26	Set_IP	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && !((NAME in ListOfData) && NAME-length == 0) && (IP in ListOfData) => Init_List_RM_Entries Init_List_AR_Resc AR_Active := 0 Stop Timer(DHCP_RetryTime) Start IP, UDP, RPC Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData) RM_Init.cnf	W_Conect
27	Set_IP	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && ((NAME in ListOfData) && NAME-length == 0) => Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData)	Set_Name
28	Set_IP	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && !((NAME in ListOfData) && NAME-length == 0) && !(IP in ListOfData) => Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData)	Set_IP
29	Set_IP	DCP_Set.ind (CREP, SA, ListOfData) /INVALID_DATA => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Set.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	Set_IP
30	Set_IP	DCP_Identify.ind (CREP, SA, ListofFilters, ResponseDelay) /VALID_FILTERS => Retrieve Data from DataSet DA := SA DCP_IdentifyQ.req (CREP,DA,ListOfData)	Set_IP
31	Set_IP	DCP_Identify.ind (CREP, SA, ListofFilters, ResponseDelay) /INVALID_FILTERS => ignore	Set_IP

#	Current State	Event /Condition =>Action	Next State
32	Set_IP	DCP_Get.ind (CREP, SA, Selectors) /VALID_SELECTORS => Retrieve Data from DataSet DCP_Get.rsp (+)(CREP,DA,ListOfData)	Set_IP
33	Set_IP	DCP_Get.ind (CREP, SA, Selectors) /!VALID_SELECTORS => Error := Invalid_DataSet DCP_Get.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	Set_IP
34	Set_IP	Time out => ExponentialBackOff(DHCP_RetryTime) Start Timer(DHCP_RetryTime) DHCPDISCOVER.req(Options)	Set_IP
35	W_Conect	DHCPOFFER.ind(IP_Para, ListOfData) /VALID_DATA && !(NAME in ListOfData) && NAME-length == 0) && AR_Active == 0 => Stop RPC, UDP, IP Store Data in DataSet Start RPC, UDP, IP	W_Conect
36	W_Conect	DHCPOFFER.ind(IP_Para, ListOfData) /!VALID_DATA ((NAME in ListOfData) && NAME-length == 0) AR_Active != 0 => ignore	W_Conect
37	W_Conect	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && ((NAME in ListOfData) && NAME-length == 0) => Stop RPC, UDP, IP Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData) RM_Stopped.ind	Set_Name
38	W_Conect	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && !(NAME in ListOfData) && NAME-length == 0) && !NAME_IP_Change => Store Data in DataSet DCP_Set.rsp (+)(CREP,DA,ListOfData)	W_Conect
39	W_Conect	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && !(NAME in ListOfData) && NAME-length == 0) && NAME_IP_Change && AR_Active == 0 => Stop RPC, UDP, IP Store Data in DataSet Start RPC, UDP, IP DCP_Set.rsp (+)(CREP,DA,ListOfData)	W_Conect
40	W_Conect	DCP_Set.ind (CREP, SA, ListOfData) /VALID_DATA && !(NAME in ListOfData) && NAME-length == 0) && NAME_IP_Change && !AR_Active == 0 => ERRCLS := Invalid_Parameter ERRCODE := Invalid_State DCP_Set.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	W_Conect
41	W_Conect	DCP_Set.ind (CREP, SA, ListOfData) /!VALID_DATA => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Set.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	W_Conect

#	Current State	Event /Condition =>Action	Next State
42	W_Conect	DCP_Identify.ind (CREP, SA, ListofFilters, ResponseDelay) /VALID_FILTERS => Retrieve Data from DataSet DA := SA DCP_IdentifyQ.req (CREP,DA,ListOfData)	W_Conect
43	W_Conect	DCP_Identify.ind (CREP, SA, ListofFilters, ResponseDelay) /!VALID_FILTERS => ignore	W_Conect
44	W_Conect	DCP_Get.ind (CREP, SA, Selectors) /VALID_SELECTORS => Retrieve Data from DataSet DCP_Get.rsp (+)(CREP,DA,ListOfData)	W_Conect
45	W_Conect	DCP_Get.ind (CREP, SA, Selectors) /!VALID_SELECTORS => ERRCLS := Invalid_Parameter ERRCODE := Invalid_DataSet DCP_Get.rsp (-) (CREP,DA,ERRCLS,ERRCODE)	W_Conect
46	W_Conect	RM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => encode IOxControlReq RPC_CControl.req (Arg)	W_Conect
47	W_Conect	RPC_CControl.cnf(-)(Arg) => AREP := Arg.AREP RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_Conect
48	W_Conect	RPC_CControl.cnf(+)(Arg) /VALID_Block_Structure => AREP := Arg.AREP RM_CControl.cnf(+)(AREP, ControlBlock)	W_Conect
49	W_Conect	RPC_CControl.cnf(+)(Arg) /!VALID_Block_Structure => ErrorCode2 := WRONG_BLOCK RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_Conect
50	W_Conect	RPC_DControl.ind(Arg) /Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && VALID_Block_Structure => AREP := Arg.AREP RM_DControl.ind (AREP, ControlBlock)	W_Conect
51	W_Conect	RPC_DControl.ind(Arg) /Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && !VALID_Block_Structure => Arg.ErrorCode2 := WRONG_BLOCK RPC_DControl.rsp(-)(Arg)	W_Conect
52	W_Conect	RPC_DControl.ind(Arg) /Search_in_List_of_RM_Entries(Arg.ARep, In_Use=FALSE) == NIL => Arg.ErrorCode2 := NO_AREP_ACT RPC_DControl.rsp(-)(Arg)	W_Conect

#	Current State	Event /Condition =>Action	Next State
53	W_Conect	RM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCo de1, ErrorCo de2, AddData1, AddData2) => encode Error(Only NDR-Header) RPC_DControl.rsp(-)(Arg)	W_Conect
54	W_Conect	RM_DControl.rsp(+)(AREP, ControlBlock) => encode IODControlRes RPC_DControl.rsp(+)(Arg)	W_Conect
55	W_Conect	RPC_Write.ind(Arg) /Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && VALID_Block_Structure => AREP := Arg.AREP Store API, SlotNumber, SubslotNumber, Index, SeqNumber RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	W_Conect
56	W_Conect	RPC_Write.ind(Arg) /Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && INVALID_Block_Structure => Arg.ErrorCode2 := WRONG_BLOCK RPC_Write.rsp(-)(Arg)	W_Conect
57	W_Conect	RPC_Write.ind(Arg) /Search_in_List_of_RM_Entries(Arg.ARep, In_Use=FALSE) == NIL => Arg.ErrorCode2 := NO_AREP_ACT(2) RPC_Write.rsp(-)(Arg)	W_Conect
58	W_Conect	RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCo de1, ErrorCo de2, AddData1, AddData2) => encode Error(Only NDR-Header) RPC_Write.rsp(-)(Arg)	W_Conect
59	W_Conect	RM_Write.rsp(+)(AREP, SeqNumber, AddData1, AddData2) => Retrieve API, SlotNumber, SubslotNumber, Index, SeqNumber encode IODWriteRes RPC_Write.rsp(+)(Arg)	W_Conect
60	W_Conect	RPC_Read.ind(Arg) /Arg.AREP==NIL && VALID_Block_Structure => AREP := NIL RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	W_Conect
61	W_Conect	RPC_Read.ind(Arg) /Arg.AREP==NIL && INVALID_Block_Structure => Arg.ErrorCode2 := WRONG_BLOCK RPC_Read.rsp(-)(Arg)	W_Conect
62	W_Conect	RPC_Read.ind(Arg) /Arg.AREP!=NIL && Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && VALID_Block_Structure => AREP := Arg.AREP Store API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	W_Conect
63	W_Conect	RPC_Read.ind(Arg) /Arg.AREP!=NIL && Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && INVALID_Block_Structure => Arg.ErrorCode2 := WRONG_BLOCK RPC_Read.rsp(-)(Arg)	W_Conect

#	Current State	Event /Condition =>Action	Next State
64	W_Conect	RPC_Read.ind(Arg) /Arg.AREP!=NIL && Search_in_List_of_RM_Entries(Arg.ARep, In_Use=FALSE) == NIL => Arg.ErrorCode2 := NO_AREP_ACT RPC_Read.rsp(-)(Arg)	W_Conect
65	W_Conect	RM_Read.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => encode Error(Only NDR-Header) RPC_Read.rsp(-)(Arg)	W_Conect
66	W_Conect	RM_Read.rsp(+)(AREP, SeqNumber, AddData1, AddData2) => Retrieve API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber encode IODReadRes RPC_Read.rsp(+)(Arg)	W_Conect
67	W_Conect	RPC_Connect.ind(Arg) /List_of_AR_resc == NIL && VALID_Block_Structure && Arg.AREP!=NIL => Arg.ErrorCode2 := NO_AREP_RSC(1) RPC_Connect.rsp(-)(Arg)	W_Conect
68	W_Conect	RPC_Connect.ind(Arg) /List_of_AR_resc != NIL && VALID_Block_Structure && Arg.AREP!=NIL && Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) == NIL => Put_to_List_of_RM_Entries(AREP=Arg.ARBLOCK.ARUUID) AR_Active := AR_Active + 1 RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq)	W_Conect
69	W_Conect	RPC_Connect.ind(Arg) /List_of_AR_resc != NIL && VALID_Block_Structure && Arg.AREP!=NIL && Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) == NIL => RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq)	W_Conect
70	W_Conect	RPC_Connect.ind(Arg) /! VALID_Block_Structure Arg.AREP==NIL => Arg.ErrorCode2 := WRONG_BLOCK RPC_Connect.rsp(-)(Arg)	W_Conect
71	W_Conect	RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => encode Error(Only NDR-Header) Rem_from_List_of_RM_Entries(AREP) AR_Active := AR_Active – 1 RPC_Connect.rsp(-)(Arg)	W_Conect
72	W_Conect	RM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => encode IODConnectRes RPC_Connect.rsp(+)(Arg)	W_Conect
73	W_Conect	RPC_Release.ind(Arg) /Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && VALID_Block_Structure => Rem_from_List_of_RM_Entries(Arg.AREP) AR_Active := AR_Active – 1 AREP := Arg.AREP RM_Release.ind(AREP, ControlBlock)	W_Conect

#	Current State	Event /Condition =>Action	Next State
74	W_Conect	RPC_Release.ind(Arg) /Search_in_List_of_RM_Entries(Arg.AREP, In_Use=FALSE) != NIL && !INVALID_Block_Structure => Arg.ErrorCode2 := WRONG_BLOCK RPC_Release.rsp(-)(Arg)	W_Conect
75	W_Conect	RPC_Release.ind(Arg) /Search_in_List_of_RM_Entries(Arg.ARep, In_Use=FALSE) == NIL => Arg.ErrorCode2 := NO_AREP_ACT(2) RPC_Release.rsp(-)(Arg)	W_Conect
76	W_Conect	RM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => encode Error(Only NDR-Header) RPC_Release.rsp(-)(Arg)	W_Conect
77	W_Conect	RM_Release.rsp(+)(AREP, ControlBlock) => encode IODReleaseRes RPC_Release.rsp(+)(Arg)	W_Conect
78	W_Conect	RM_Abort.req(AREP) /Search_in_List_of_RM_Entries(AREP, In_Use=FALSE) != NIL => Rem_from_List_of_RM_Entries(AREP) AR_Active := AR_Active – 1 RM_Abort.cnf(AREP)	W_Conect
79	W_Conect	RM_Abort.req(AREP) /Search_in_List_of_RM_Entries(ARep, In_Use=FALSE) == NIL => RM_Abort.cnf(AREP)	W_Conect
80	any state	DCPUCR_Close.cnf (-) (CREP, ERRCLS, ERRCODE) =>	same state
81	any state	DCPUCR_Close.cnf (+)(CREP) =>	same state
82	any state	DCPMCR_Close.cnf (-) (CREP, ERRCLS, ERRCODE) =>	same state
83	any state	DCPMCR_Close.cnf (+)(CREP) =>	same state

5.6.8.4 Functions

There are the same functions as defined for NRPM (see 5.6.7.4) .

5.6.9 CMDEV

5.6.9.1 Primitive definitions

5.6.9.1.1 Primitives exchanged between FSPMDEV and CMDEV

Table 344 shows the service primitives including their associated parameters issued by FSPM of the IO device (FSPMDEV) and received by CMDEV.

Table 344 — Primitives issued by FSPMDEV to CMDEV

Primitive name	Source	Associated parameters	Functions
Init.req	FSPM	-	The service Init initializes the RMPM.
Abort.req	FSPM	AREP	The service Abort removes the stored AREP from the list of AREPs.
CControl.req	FSPM	AREP CCTl Parameter	The service CControl conveys the appliction ready flag.

Primitive name	Source	Associated parameters	Functions
Connect.rsp(+)	FSPM	AREP AR Response Block List of IO CR Response Blocks Alarm CR Response Block List of IO Data Response Blocks	This service primitive is a positive response to establish an application relationship.
Connect.rsp(-)	FSPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative response and the requested application relationship is not established.
DControl.rsp(+)	FSPM	AREP DCTI Response	This service primitive is a response to end of parameter signal.
DControl.rsp(-)	FSPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a response to end of parameter signal.
Read.rsp(+)	FSPM	AREP Length Data	This service primitive is the response to the Read service.
Read.rsp(-)	FSPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is the response to the Read service.
Write.rsp(+)	FSPM	AREP	This service primitive is the response to the Write service.
Write.rsp(-)	FSPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is the response to the Write service.
CM_Release.rsp(+)	FSPM	AREP	This service primitive is the response to the Release service.
CM_Release.rsp(-)	FSPM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is the response to the Release service.

Table 345 shows the service primitives including their associated parameters issued by CMDEV and received by FSPM of the IO device (FSPMDEV).

Table 345 — Primitives issued by CMDEV to FSPMDEV

Primitive name	Source	Associated parameters	Functions
Init.cnf(+)	CM	-	This service primitive indicates that the Init service succeeded.
CControl.cnf	CM	AREP, Ct_Arg	This service primitive confirms the CControl service.
Connect.ind	CM	AREP, Ident_Number, AR Type, List of related CRs	This service primitive indicates a request to establish an application relationship.
DControl.ind	CM	AREP, Ct_Arg	This service primitive indicates the end of parameter.
Read.ind	CM	AREP, R_Arg	This service primitive indicates the Read service.
Write.ind	CM	AREP, W_Arg	This service primitive indicates the Write service.

Primitive name	Source	Associated parameters	Functions
Release.ind	CM	AREP	This service primitive indicates that the application relationship should be released.
Stopped.ind	CM	-	This service primitive indicates that a new device name was set and IP stack was therefore stopped to receive new IP parameter.

5.6.9.1.2 Parameters of CMDEV primitives

The parameters used with the primitives exchanged between the FSPM and the CMDEV are described in FAL Service Definition.

5.6.9.2 State machine description

The CM Device protocol machine is present for each AR of an IO Device.

5.6.9.3 CMDEV state table

Table 346 contains the complete description of the CMDEV state machine.

Table 346 — 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(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq) /CHECK_AR_IO_TYPE && ListOfMCRBlock == NIL => CREP.PPM := create (PPM) CREP.CPM := create (CPM) CREP.APM := create (AIPM) First := TRUE Multicast_Comm := FALSE CM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq)	W_CRES
3	W_CIND	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq) /CHECK_AR_IO_TYPE && ListOfMCRBlock != NIL && Resources available => CREP.PPM := create (PPM) CREP.MPPM := create (MPPM) CREP.CPM := create (CPM) CREP.APM := create (AIPM) First := TRUE Multicast_Comm := TRUE NRMC_Timeout := FALSE CM_ApplReady := FALSE RM_ApplReady := FALSE or all M-Consumer-CRs part of the AR do: CREP.NRMC := create (NRMC) CM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq)	W_CRES

#	Current State	Event /Condition =>Action	Next State
3	W_CIND	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq) /CHECK_AR_IO_TYPE && ListOfMCRBlock != NIL && Resources not available => ErrorDecode:= PNIO ErrorCode1 := resource ErrorCode2:= detail LogEntry() RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2)	W_CIND
5	W_CIND	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, ListOfMCRBlockReq) /!CHECK_AR_IO_TYPE => ErrorDecode:= PNIO ErrorCode1 := block of error ErrorCode2:= detail LogEntry() RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2)	W_CIND
6	W_CIND	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_CIND
7	W_CIND	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CIND
8	W_CIND	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => ErrorDecode:= PNIORW ErrorCode1 := state conflict RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CIND
9	W_CIND	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_CIND
10	W_CIND	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CIND
11	W_CIND	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIORW ErrorCode1 := state conflict RM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CIND
12	W_CIND	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_CIND
13	W_CIND	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CIND

#	Current State	Event /Condition =>Action	Next State
14	W_CIND	RM_DControl.ind(AREP, ControlBlock) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2)	W_CIND
15	W_CIND	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_CIND
16	W_CIND	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CIND
17	W_CIND	RM_Release.ind(AREP, ControlBlock) => RM_Release.rsp(+)(AREP, ControlBlock)	W_CIND
18	W_CIND	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	W_CIND
19	W_CIND	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CIND
20	W_CIND	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CIND
21	W_CIND	RM_CControl.cnf(+)(AREP, ControlBlock) => ignore	W_CIND
22	W_CIND	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CIND
23	W_CRES	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
24	W_CRES	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) /IOCRProperties.MProviderDataStatus == 0x00 => for all MI-CRs part of the AR do: Inv_Msk := INVALID ; Rem_Sts(CREP), Default_Status(CREP) := Default_Status(CREP) & Inv_Msk PPM_Activate.req(CREP.PPM, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status)	W_PPM_O
25	W_CRES	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) /IOCRProperties.MProviderDataStatus == 0x01 => PPM_Activate.req(CREP.PPM, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status)	W_PPM_O

#	Current State	Event /Condition =>Action	Next State
26	W_CRES	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2)	W_CIND
27	W_CRES	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => ignore	W_CRES
28	W_CRES	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_CRES
29	W_CRES	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CRES
30	W_CRES	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ignore	W_CRES
31	W_CRES	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_CRES
32	W_CRES	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CRES
33	W_CRES	RM_DControl.ind(AREP, ControlBlock) => ignore	W_CRES
34	W_CRES	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_CRES
35	W_CRES	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CRES
36	W_CRES	RM_Release.ind(AREP, ControlBlock) => ignore	W_CRES
37	W_CRES	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	W_CRES
38	W_CRES	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_CRES
39	W_CRES	CM_Set_Prov_Status.req (AREP, CREP, D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP, D_Status)	W_CRES

#	Current State	Event /Condition =>Action	Next State
40	W_CRES	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CRES
41	W_CRES	RM_CControl.cnf(+)(AREP, ControlBlock) =>	W_CRES
42	W_CRES	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CRES
43	W_PPM_O	PPM_Activate.cnf(CREP.PPM) /NOT (last one) => ignore	W_PPM_O
44	W_PPM_O	PPM_Activate.cnf(CREP.PPM) /last one && ListOfMCRBlock == NIL => ALPMI_Activate.req(CREP.ALPMI, DA, SA, VLAN, RTATimeoutFactor, MRetry)	W_ALPM_O
45	W_PPM_O	PPM_Activate.cnf(CREP.PPM) /last one && ListOfMCRBlock != NIL => Start NRMCTimer(MCTimeoutFactor) ProviderStationName := MCRBlockReq.ProviderStationName NRMCT_Activate.req(CREP, ProviderStationName, CR_Parameter)	W_NRMCT_O
46	W_NRMCT_O	NRMCT_Activate.cnf(+)(CREP) /NOT (last one) => ignore NRMCT_Activate.req(CREP, ProviderStationName, CR_Parameter)	W_NRMCT_O
47	W_NRMCT_O	NRMCT_Activate.cnf(+)(CREP) /last one => ALPMI_Activate.req(CREP.ALPMI, DA, SA, VLAN, RTATimeoutFactor, MRetry)	W_ALPM_O
48	W_NRMCT_O	NRMCT_Activate.cnf(-)(CREP, Err) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict Stop Timer RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
49	W_ALPM_O	ALPMI_Activate.cnf(CREP.APM) => Start Timer(CMI_Timeout) RM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	W_PIND
50	W_PIND	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict Stop Timer RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
51	W_PIND	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_PIND
52	W_PIND	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PIND
53	W_PIND	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) /First => First := FALSE Stop Timer CPM_Activate.req(CREP.CPM, DA, SA, FrameID, Prio, VLAN, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status)	W_ACTC1
54	W_PIND	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) /!First => CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	W_PRES
55	W_PIND	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_PIND
56	W_PIND	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PIND
57	W_PIND	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PIND
58	W_PIND	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_PIND
59	W_PIND	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PIND
60	W_PIND	RM_DControl.ind(AREP, ControlBlock) /First => First := FALSE Stop Timer CPM_Activate.req(CREP.CPM, DA, SA, FrameID, Prio, VLAN, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status)	W_ACTC2
61	W_PIND	RM_DControl.ind(AREP, ControlBlock) /!First => CM_DControl.ind(AREP, ControlBlock)	W_ERES
62	W_PIND	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_PIND

#	Current State	Event /Condition =>Action	Next State
63	W_PIND	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PIND
64	W_PIND	RM_Release.ind(AREP, ControlBlock) => Stop Timer CM_Release.ind(AREP, ControlBlock)	W_RRES
65	W_PIND	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	W_PIND
66	W_PIND	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PIND
67	W_PIND	CM_Set_Prov_Status.req (AREP, CREP, D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP, D_Status)	W_PIND
68	W_PIND	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNIOEW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PIND
69	W_PIND	RM_CControl.cnf(+)(AREP, ControlBlock) => ignore	W_PIND
70	W_PIND	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PIND
71	W_PIND	Timeout => RM_Abort.req(AREP)	REMOVE
72	W_PIND	NRMC_InData.ind(CREP) => ignore	W_PIND
73	W_PIND	Timeout_NRMC_Timer => NRMC_Timeout := TRUE	W_PIND
74	W_PIND	CM_Abort.req (AREP) => Stop Timer RM_Abort.req(AREP)	REMOVE
75	W_PIND	CPM_Stop.ind(CREP) /CREP != CREP.MCPM => Stop Timer RM_Abort.req(AREP)	REMOVE
76	W_PIND	CPM_New_Data.ind(CREP.CPM, APDU_Status) => Stop Timer	W_PIND

#	Current State	Event /Condition =>Action	Next State
77	W_PIND	PPM_Error.ind => Stop Timer RM_Abort.req(AREP)	REMOVE
78	W_ACTC1	CPM_Activate.cnf(CREP.CPM) => CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	W_PRES
79	W_ACTC1	NRMC_InData.ind(CREP) => ignore	W_ACTC1
80	W_ACTC1	Timeout_NRMC_Timer => NRMC_Timeout := TRUE	W_ACTC1
81	W_ACTC2	CPM_Activate.cnf(CREP.CPM) => CM_DControl.ind(AREP, ControlBlock)	W_ERES
82	W_ACTC2	NRMC_InData.ind(CREP) => ignore	W_ACTC2
83	W_ACTC2	Timeout_NRMC_Timer => NRMC_Timeout := TRUE	W_ACTC2
84	W_PRES	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ignore	W_PRES
85	W_PRES	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_PRES
86	W_PRES	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PRES
87	W_PRES	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => ignore	W_PRES
88	W_PRES	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => RM_Write.rsp(+)(AREP, Multiple, SeqNumber, AddData1, AddData2)	W_PIND
89	W_PRES	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PIND
90	W_PRES	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ignore	W_PRES

#	Current State	Event /Condition =>Action	Next State
91	W_PRES	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_PRES
92	W_PRES	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PRES
93	W_PRES	RM_DControl.ind(AREP, ControlBlock) => ignore	W_PRES
94	W_PRES	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_PRES
95	W_PRES	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PRES
96	W_PRES	RM_Release.ind(AREP, ControlBlock) => ignore	W_PRES
97	W_PRES	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	W_PRES
98	W_PRES	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PRES
99	W_PRES	CM_Set_Prov_Status.req (AREP, CREP, D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP, D_Status)	W_PRES
100	W_PRES	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNORW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PRES
101	W_PRES	RM_CControl.cnf(+)(AREP, ControlBlock) => ignore	W_PRES
102	W_PRES	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_PRES
103	W_PRES	CM_Abort.req (AREP) => RM_Abort.req(AREP)	REMOVE
104	W_PRES	CPM_Stop.ind(CREP) /CREP != CREP.MCPM => RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
105	W_PRES	NRMC_InData.ind(CREP) => ignore	W_PRES
106	W_PRES	Timeout_NRMC_Timer => NRMC_Timeout := TRUE	W_PRES
107	W_PRES	CPM_New_Data.ind(CREP.CPM, APDU_Status) => ignore	W_PRES
108	W_PRES	PPM_Error.ind => RM_Abort.req(AREP)	REMOVE
109	W_ERES	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
110	W_ERES	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_ERES
111	W_ERES	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ERES
112	W_ERES	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => RM_Abort.req(AREP)	REMOVE
113	W_ERES	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_ERES
114	W_ERES	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ERES
115	W_ERES	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => RM_Abort.req(AREP)	REMOVE
116	W_ERES	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_ERES
117	W_ERES	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ERES
118	W_ERES	RM_DControl.ind(AREP, ControlBlock) => RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
119	W_ERES	CM_DControl.rsp(+) (AREP, ControlBlock) => StoreControlBlock RM_DControl.rsp(+)(AREP, ControlBlock)	W_ARDY
120	W_ERES	CM_DControl.rsp(-) (AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
121	W_ERES	RM_Release.ind (AREP, ControlBlock) => RM_Abort.req(AREP)	REMOVE
122	W_ERES	CM_Release.rsp(+) (AREP, ControlBlock) => ignore	W_ERES
123	W_ERES	CM_Release.rsp(-) (AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ERES
124	W_ERES	CM_Set_Prov_Status.req (AREP, CREP, D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP, D_Status)	W_ERES
125	W_ERES	CM_CControl.req (AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNORW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ERES
126	W_ERES	RM_CControl.cnf(+) (AREP, ControlBlock) => ignore	W_ERES
127	W_ERES	RM_CControl.cnf(-) (AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ERES
128	W_ERES	CM_Abort.req (AREP) => RM_Abort.req(AREP)	REMOVE
129	W_ERES	CPM_Stop.ind (CREP) /CREP != CREP.MCPM => RM_Abort.req(AREP)	REMOVE
130	W_ERES	NRMC_InData.ind (CREP) => ignore	W_ERES
131	W_ERES	Timeout_NRMC_Timer => NRMC_Timeout := TRUE	W_ERES
132	W_ERES	CPM_New_Data.ind (CREP.CPM, APDU_Status) => ignore	W_ERES
133	W_ERES	PPM_Error.ind => RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
134	W_ARDY	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
135	W_ARDY	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_ARDY
136	W_ARDY	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ARDY
137	W_ARDY	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ARDY
138	W_ARDY	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_ARDY
139	W_ARDY	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ARDY
140	W_ARDY	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ARDY
141	W_ARDY	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_ARDY
142	W_ARDY	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ARDY
143	W_ARDY	RM_DControl.ind(AREP, ControlBlock) /PRMEND => Retrieve ControlBlock RM_DControl.rsp(+)(AREP, ControlBlock)	W_ARDY
144	W_ARDY	RM_DControl.ind(AREP, ControlBlock) /!PRMEND => CM_DControl.ind(AREP, ControlBlock)	W_ARDY
145	W_ARDY	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_ARDY

#	Current State	Event /Condition =>Action	Next State
146	W_ARDY	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ARDY
147	W_ARDY	RM_Release.ind(AREP, ControlBlock) => CM_Release.ind(AREP, ControlBlock)	W_RRES
148	W_ARDY	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	W_ARDY
149	W_ARDY	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ARDY
150	W_ARDY	CM_Set_Prov_Status.req (AREP,CREP,D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP,D_Status)	W_ARDY
151	W_ARDY	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) /Multicast_Comm == FALSE => RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock)	W_RDYC
152	W_ARDY	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) /Multicast_Comm == TRUE && NRMC_Timeout == FALSE && CHECK_ALL_MC_INDATA() == TRUE => Stop NRMC_Timer RM_ApplReady := TRUE RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock)	W_RDYC
153	W_ARDY	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) /Multicast_Comm == TRUE && NRMC_Timeout == FALSE && CHECK_ALL_MC_INDATA() == FALSE => Store AREP, ControlBlock, ModuleDiffBlock CM_ApplReady := TRUE	W_ARDY
154	W_ARDY	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) /Multicast_Comm == TRUE && NRMC_Timeout == TRUE && CHECK_ALL_MC_INDATA() == TRUE => RM_ApplReady := TRUE RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock)	W_RDYC

#	Current State	Event /Condition =>Action	Next State
155	W_ARDY	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) /Multicast_Comm == TRUE && NRMC_Timeout == TRUE && CHECK_ALL_MC_INDATA() == FALSE => RM_ApplReady := TRUE For all M-Consumer-CRs part of AR do: if (MC_InData(CREP) == FALSE) do: SubmoduleState.ARInfo := ApplicationReadyPending ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock) if (MC_InData(CREP) == FALSE) do: DiagnosisEvent.ind(AREP, CREP, API, Diagnosis Data)	W_RDYC
156	W_ARDY	NRMC_InData.ind(CREP) /NRMC_Timeout == TRUE CHECK_ALL_MC_INDATA() == FALSE => ignore	W_ARDY
157	W_ARDY	NRMC_InData.ind(CREP) /NRMC_Timeout == FALSE && CHECK_ALL_MC_INDATA() == TRUE && CM_ApplReady := TRUE => Stop NRMC_Timer RM_ApplReady := TRUE RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock)	W_RDYC
158	W_ARDY	NRMC_InData.ind(CREP) /NRMC_Timeout == FALSE && CHECK_ALL_MC_INDATA() == TRUE && CM_ApplReady := FALSE => Stop NRMC_Timer	W_ARDY
159	W_ARDY	Timeout_NRMC_Timer /CM_ApplReady == FALSE => NRMC_Timeout := TRUE	W_ARDY
160	W_ARDY	Timeout_NRMC_Timer /CM_ApplReady == TRUE => RM_ApplReady := TRUE For all M-Consumer-CRs do: if (MC_InData(CREP) == FALSE) do: SubmoduleState.ARInfo := ApplicationReadyPending RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock)	W_RDYC
161	W_ARDY	RM_CControl.cnf(+)(AREP, ControlBlock) => CM_CControl.cnf(+)(AREP, ControlBlock)	W_ARDY
162	W_ARDY	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_ARDY
163	W_ARDY	CM_Abort.req(AREP) => RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
164	W_ARDY	CPM_Stop.ind(CREP) /CREP != CREP.MCPM => RM_Abort.req(AREP)	REMOVE
165	W_ARDY	CPM_New_Data.ind(CREP.CPM, APDU_Status) => ignore	W_ARDY
166	W_ARDY	PPM_Error.ind => RM_Abort.req(AREP)	REMOVE
167	W_RDYC	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
168	W_RDYC	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_RDYC
169	W_RDYC	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RDYC
170	W_RDYC	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RDYC
171	W_RDYC	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_RDYC
172	W_RDYC	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RDYC
173	W_RDYC	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RDYC
174	W_RDYC	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_RDYC
175	W_RDYC	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RDYC

#	Current State	Event /Condition =>Action	Next State
176	W_RDYC	RM_DControl.ind(AREP, ControlBlock) /PRMEND => Retrieve ControlBlock RM_DControl.rsp(+)(AREP, ControlBlock)	W_RDYC
177	W_RDYC	RM_DControl.ind(AREP, ControlBlock) /!PRMEND => CM_DControl.ind(AREP, ControlBlock)	W_RDYC
178	W_RDYC	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_RDYC
179	W_RDYC	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RDYC
180	W_RDYC	RM_Release.ind(AREP, ControlBlock) => CM_Release.ind(AREP, ControlBlock)	W_RRES
181	W_RDYC	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	W_RDYC
182	W_RDYC	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RDYC
183	W_RDYC	CM_Set_Prov_Status.req(AREP, CREP, D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req(CREP, D_Status)	W_RDYC
184	W_RDYC	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => RM_CControl.req(AREP, ControlBlock; ModuleDiffBlock)	W_RDYC
185	W_RDYC	RM_CControl.cnf(+)(AREP, ControlBlock) => First := TRUE D_Status.DataValid := TRUE CM_CControl.cnf(+)(AREP, ControlBlock) for all MI-CRs part of the AR do: PPM_Set_Prov_Status.req(CREP, D_Status)	DATA
186	W_RDYC	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) RM_Abort.req(AREP)	REMOVE
187	W_RDYC	CM_Abort.req(AREP) => RM_Abort.req(AREP)	REMOVE
188	W_RDYC	CPM_Stop.ind(CREP) /CREP != CREP.MCPM => RM_Abort.req(AREP)	REMOVE
189	W_RDYC	PPM_Error.ind => RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
190	DATA	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE
191	DATA	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	DATA
192	DATA	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	DATA
193	DATA	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => CM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data)	DATA
194	DATA	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => RM_Write.rsp(+)(AREP, Multiple, SeqNumber, AddData1, AddData2)	DATA
195	DATA	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
196	DATA	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => CM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	DATA
197	DATA	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => RM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	DATA
198	DATA	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
199	DATA	RM_DControl.ind(AREP, ControlBlock) /PRMEND => Retrieve ControlBlock RM_DControl.rsp(+)(AREP, ControlBlock)	DATA
200	DATA	RM_DControl.ind(AREP, ControlBlock) /!PRMEND => CM_DControl.ind(AREP, ControlBlock)	DATA
201	DATA	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	DATA
202	DATA	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	DATA

#	Current State	Event /Condition =>Action	Next State
203	DATA	RM_Release.ind(AREP, ControlBlock) => CM_Release.ind(AREP, ControlBlock)	W_RRES
204	DATA	CM_Release.rsp(+)(AREP, ControlBlock) => ignore	DATA
205	DATA	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	DATA
206	DATA	CM_Set_Prov_Status.req (AREP,CREP,D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP,D_Status)	DATA
207	DATA	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
208	DATA	RM_CControl.cnf(+)(AREP, ControlBlock) => ignore	DATA
209	DATA	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	DATA
210	DATA	CM_Abort.req (AREP) => RM_Abort.req(AREP)	REMOVE
211	DATA	CPM_Stop.ind(CREP) /CREP != CREP.MCPM => RM_Abort.req(AREP)	REMOVE
212	DATA	CPM_New_Data.ind(CREP.CPM, APDU_Status) /First => First := FALSE CM_In_Data.ind(CREP.CPM) CM_New_Data.ind(CREP.CPM, APDU_Status)	DATA
213	DATA	CPM_New_Data.ind(CREP.CPM, APDU_Status) /!First => CM_New_Data.ind(CREP.CPM, APDU_Status)	DATA
214	DATA	PPM_Error.ind => RM_Abort.req(AREP)	REMOVE
215	W_RRES	RM_Connect.ind(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq) => ErrorDecode:= PNIOBW ErrorCode1 := state conflict RM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
216	W_RRES	CM_Connect.rsp(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => ignore	W_RRES
217	W_RRES	CM_Connect.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RRES
218	W_RRES	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => RM_Abort.req(AREP)	REMOVE
219	W_RRES	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	W_RRES
220	W_RRES	CM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RRES
221	W_RRES	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => RM_Abort.req(AREP)	REMOVE
222	W_RRES	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	W_RRES
223	W_RRES	CM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RRES
224	W_RRES	RM_DControl.ind(AREP, ControlBlock) => RM_Abort.req(AREP)	REMOVE
225	W_RRES	CM_DControl.rsp(+)(AREP, ControlBlock) => ignore	W_RRES
226	W_RRES	CM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RRES
227	W_RRES	RM_Release.ind(AREP, ControlBlock) => RM_Abort.req(AREP)	REMOVE
228	W_RRES	CM_Release.rsp(+)(AREP, ControlBlock) => RM_Abort.req(AREP)	REMOVE
229	W_RRES	CM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Release.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2) RM_Abort.req(AREP)	REMOVE

#	Current State	Event /Condition =>Action	Next State
230	W_RRES	CM_Set_Prov_Status.req (AREP,CREP,D_Status) => Rem_Sts(CREP), D_Status := D_Status & Inv_Msk PPM_Set_Prov_Status.req (CREP,D_Status)	W_RRES
231	W_RRES	CM_CControl.req(AREP, ControlBlock, ModuleDiffBlock) => ErrorDecode:= PNORW ErrorCode1 := state conflict CM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RRES
232	W_RRES	RM_CControl.cnf(+)(AREP, ControlBlock) => ignore	W_RRES
233	W_RRES	RM_CControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => ignore	W_RRES
234	W_RRES	CM_Abort.req (AREP) => RM_Abort.req(AREP)	REMOVE
235	W_RRES	CPM_Stop.ind(CREP) => RM_Abort.req(AREP)	REMOVE
236	W_RRES	CPM_New_Data.ind(CREP.CPM, APDU_Status) => CM_New_Data.ind(CREP.CPM, APDU_Status)	W_RRES
237	W_RRES	PPM_Error.ind => RM_Abort.req(AREP)	REMOVE
238	REMOVE	RM_Abort.cnf(AREP) /Multicast_Comm == FALSE => CPM_Close.req(CREP.CPM)	W_CPM_C
239	REMOVE	RM_Abort.cnf(AREP) /Multicast_Comm == TRUE => NRMC_Close.req(CREP)	W_NRMC_C
240	REMOVE	RM_Write.ind(AREP, API, SlotNumber, SubslotNumber, Index, Multiple, SeqNumber, Length, Data) => ErrorDecode:= PNORW ErrorCode1 := state conflict RM_Write.rsp(-)(AREP, Multiple, SeqNumber, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	REMOVE
241	REMOVE	CM_Write.rsp(+)(AREP, Multiple, Seq Number, AddData1, AddData2) => ignore	REMOVE
242	REMOVE	RM_Read.ind(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNORW ErrorCode1 := state conflict RM_Read.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	REMOVE
243	REMOVE	CM_Read.rsp(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => ignore	REMOVE

#	Current State	Event /Condition =>Action	Next State
244	REMOVE	RM_DControl.ind(AREP, ControlBlock) => Error := Wrong_State RM_DControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2)	REMOVE
245	REMOVE	RM_Release.ind(AREP, ControlBlock) => ignore	REMOVE
246	REMOVE	CM_Abort.req (AREP) => ignore	REMOVE
247	REMOVE	CPM_Stop.ind(CREP) => ignore	REMOVE
248	REMOVE	CPM_New_Data.ind(CREP.CPM, APDU_Status) => ignore	REMOVE
249	REMOVE	PPM_Error.ind => ignore	REMOVE
250	W_NRMC_C	NRMC_Close.cnf(CREP) => CPM_Close.req(CREP.CPM)	W_CPM_C
251	W_CPM_C	CPM_Close.cnf(CREP.CPM) => PPM_Close.req(CREP.PPM)	W_PPM_C
252	W_PPM_C	PPM_Close.cnf(CREP.PPM) => ALPMI_Close.req(CREP.ALPM)	W_ALPMI_C
253	W_ALPMI_C	ALPMI_Close.cnf(CREP.ALPM) => remove (CREP.PPM) remove (CREP.CPM) remove (CREP.ARPM) remove (CREP.NRMC) CM_Abort.ind(AREP)	W_CIND
254	any State	PPM_Set_Prov_Status.cnf(+) (CREP) => CM_Set_Prov_Status.cnf(+) (AREP, CREP)	same State
255	any State	PPM_Set_Prov_Status.cnf(-) (CREP,ERRCLS,ERRCODE) => CM_Set_Prov_Status.cnf(-) (AREP,CREP,ERRCLS,ERRCODE)	same State

5.6.9.4 Functions

There are no functions or macros defined for CMDEV.

5.6.10 NRMC

5.6.10.1 Primitive definitions

5.6.10.1.1 Primitives exchanged between NRMC and CMDEV

Table 347 shows the service primitives including their associated parameters issued by CM of the IO device (CMDEV) and received by NRMC.

Table 347 — Primitives issued by CMDEV to NRMC

Primitive name	Source	Associated parameters	Functions
NRMC_Activate.req	CMDEV	CREP ProviderStationName CR_Parameter	
NRMC_Close.req	CMDEV	CREP	

Table 345 shows the service primitives including their associated parameters issued by NRMC and received by CM of the IO device (CMDEV).

Table 348 — Primitives issued by NRMC to CMDEV

Primitive name	Source	Associated parameters	Functions
NRMC_Activate.cnf(+)	NRMC	CREP	
NRMC_Activate.cnf(-)	NRMC	CREP Err	
NRMC_Close.cnf	NRMC	CREP	
NRMC_InData.ind	NRMC	CREP	
DiagnosisEvent.ind	NRMC	AREP CREP Diagnosis Data	

5.6.10.1.2 Primitives exchanged between NRMC and CPM

Table 349 shows the service primitives including their associated parameters issued by CPM and received by NRMC.

Table 349 — Primitives issued by CPM to NRMC

Primitive name	Source	Associated parameters	Functions
CPM_Activate.cnf (-)	CPM	CREP ERRCLS ERRCODE	
CPM_Activate.cnf (+)	CPM	CREP	
CPM_Close.cnf	CPM	CREP	
CPM_New_Cons_Data.ind	CPM	CREP APDU_Status	
CPM_Stop.ind	CPM	CREP	

Table 350 shows the service primitives including their associated parameters issued by NRMC and received by CPM.

Table 350 — Primitives issued by NRMC to CPM

Primitive name	Source	Associated parameters	Functions
CPM_Activate.req	NRMC	CREP DA SA Frameld Prio VLAN Exp_Lenth StartTon Timeout_Base WatchDogTime DataHoldTime Default_Value Default_Status	
CPM_Close.req	NRMC	CREP	

5.6.10.1.3 Primitives exchanged between NRMC and other machines

Table 351 shows the service primitives including their associated parameters issued by other machines and received by NRMC.

Table 351 — Primitives issued by other machines to NRMC

Primitive name	Source	Associated parameters	Functions
DCP_Identify.cnf(-)	DCP	CREP ERRCLS ERRCODE	
DCP_Identify.cnf(+)	DCP	CREP	
DCP_IdentifyQ.ind	DCP	CREP SA ListOfData	
DCPMCS_Activate.cnf (-)	DCPMS	CREP ERRCLS ERRCODE	
DCPMCS_Activate.cnf (-)	DCPMS	CREP	

5.6.10.1.4 Parameters of NRMC primitives

The parameters used with the primitives exchanged between the NRMC and the CMDEV are described in FAL Service Definition.

5.6.10.2 State machine description

The NRMC machine is present for each multicast consumer.

Local Variables

RM_AppIReady

This local variable shows that RM_CControl.req from CMDEV was always initiated.

MC_InData(CREP)

Via CREP a valid frame was received the first time.

5.6.10.3 NRMC state table

Table 352 contains the complete description of the NRMC state machine.

Table 352 — NRMC state table

#	Current State	Event /Condition =>Action	Next State
1	W-START	NRMC_Activate.req(CREP, ProviderStationName, CR_Parameter) => Store CRParameter DCPMCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPMC_Timeout)	ACT1
2	W-START	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP)	W-START
3	W-START	CPM_Close.cnf(CREP) => ignore	W-START
4	W-START	DCP_IdentifyQ.ind(CREP,SA,ListOfData) => ignore	W-START
5	W-START	DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE) => ignore	W-START
6	W-START	DCP_Identify.cnf(+)(CREP) => ignore	W-START
7	W-START	CPM_Activate.cnf(+)(CREP) => ignore	W-START

#	Current State	Event /Condition =>Action	Next State
8	W-START	CPM_New_Cons_Data.ind(CREP, APDU_Status) => ignore	W-START
9	W-START	CPM_Stop.ind(CREP) => ignore	W-START
10	ACT1	DCPMCS_Activate.cnf (-) (CREP, ERRCLS, ERRCODE) => NRMC_Activate.cnf(-) (CREP,Err=No_DCPact)	W-START
11	ACT1	DCPMCS_Activate.cnf (+)(CREP) => MCName := ProviderStationName DA := DCPMC_add ListofFilter := DeviceProperties, NameOfStation, MCName ResponseDelay := 1 NRMC_Activate.cnf(+)(CREP) DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay)	W_DCP_I
12	W_DCP_I	DCP_IdentifyQ.ind(CREP,SA,ListOfData) => store SA	W_DCP_T
13	W_DCP_I	DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE) => DA := DCPMC_add ListofFilter := DeviceProperties, NameOfStation, MCName ResponseDelay := 1 DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay)	W_DCP_I
14	W_DCP_I	DCP_Identify.cnf(+)(CREP) => DA := DCPMC_add ListofFilter := DeviceProperties, NameOfStation, MCName ResponseDelay := 1 ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := AddressResolutionFailed DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay) DiagnosisEvent.ind(AREP, CREP, Diagnosis Data)	W_DCP_I
15	W_DCP_I	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) DCPMCS_Close.req(CREP)	W-START
16	W_DCP_T	DCP_IdentifyQ.ind(CREP,SA,ListOfData) => ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := AddressResolutionFailed DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay) DiagnosisEvent.ind(AREP, CREP, Diagnosis Data)	W_DCP_I
17	W_DCP_T	DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE) => DA := DCPMC_add ListofFilter := DeviceProperties, NameOfStation, MCName ResponseDelay := 1 DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay)	W_DCP_I

#	Current State	Event /Condition =>Action	Next State
18	W_DCP_T	DCP_Identify.cnf(+)(CREP) => CREP := create(CPM) MC_InData(CREP) := FALSE CPM_Activate.req(CREP, DA, SA, FrameId, Prio, VLAN, Exp_Lenth, StartTon, Timeout_Base, WatchDogTime, DataHoldTime, Default_Value, Default_Status)	WF_ACTC
19	W_DCP_T	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) DCPMCS_Close.req(CREP)	W-START
20	WF_ACTC	CPM_Activate.cnf(+)(CREP) =>	WF_NRMC_ID
21	WF_ACTC	CPM_Activate.cnf(-)(CREP,ERRCLS,ERRCODE) => CPM_Close.req(CREP) DCPMCS_Close.req(CREP)	W-START
22	WF_ACTC	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) CPM_Close.req(CREP) DCPMCS_Close.req(CREP)	W-START
23	WF_NRMC_ID	CPM_New_Cons_Data.ind(CREP, APDU_Status) /RM_ApplReady == FALSE => MC_InData(CREP) := TRUE NRMC_InData.ind(CREP)	NRMC_ID
24	WF_NRMC_ID	CPM_New_Cons_Data.ind(CREP, APDU_Status) /RM_ApplReady == TRUE => MC_InData(CREP) := TRUE ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x02 ChannelErrorType := Multicast CR Mismatch DiagnosisEvent.ind(AREP, CREP, Diagnosis Data)	NRMC_ID
25	WF_NRMC_ID	CPM_Stop.ind(CREP) => ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out DiagnosisEvent.ind(AREP, CREP, Diagnosis Data) CPM_Close.req(CREP) CM_Stop.ind(CREP)	WF_CLC1
26	WF_NRMC_ID	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) CPM_Close.req(CREP) DCPMCS_Close.req(CREP)	W-START
27	NRMC_ID	CPM_Stop.ind(CREP) /RM_ApplReady == FALSE => MC_InData(CREP) := FALSE CPM_Close.req(CREP) CM_Stop.ind(CREP)	WF_CLC1

#	Current State	Event /Condition =>Action	Next State
28	NRMC_ID	CPM_Stop.ind(CREP) /RM_ApplReady == TRUE => ChannelNumber := 0x8000 ChannelProperties.Type := 0x00 ChannelProperties.Specifier := 0x01 ChannelErrorType := Multicast CR Mismatch ExtChannelErrorType := Multicast Consumer CR timed out CPM_Close.req(CREP) CM_Stop.ind(CREP) DiagnosisEvent.ind(AREP, CREP, Diagnosis Data)	WF_CLC2
29	NRMC_ID	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) CPM_Close.req(CREP) DCPMCS_Close.req(CREP)	W-START
30	WF_CLC1	CPM_Close.cnf(CREP) => DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay)	W_DCP_I
31	WF_CLC1	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) CPM_Close.req(CREP) DCPMCS_Close.req(CREP)	W-START
32	WF_CLC2	CPM_Close.cnf(CREP) => DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay)	W_DCP_I
33	WF_CLC2	NRMC_Close.req(CREP) => NRMC_Close.cnf(CREP) DCPMCS_Close.req(CREP)	W-START

5.6.10.4 Functions

There are no functions or macros defined for NRMC.

5.6.11 CMCTL

5.6.11.1 Primitive definitions

5.6.11.1.1 Primitives exchanged between FSPMCTL and CMCTL

Table 353 shows the service primitives including their associated parameters issued by FSPM of the IO controller (FSPMCTL) and received by CMCTL.

Table 353 — Primitives issued by FSPMCTL to CMCTL

Primitive name	Source	Associated parameters	Functions
Init.req	FSPM		The service Init initializes the CMCTL.
Release.req	FSPM	AREP ReleaseBlock	The service Release closes an existing connection.
Connect.req	FSPM	AREP ARBlockReq ListOfIOCRBlockReq AlarmCRBlockReq ListOfExpectedSubmoduleBlockReq PrmCount ListOfWriteBlockReq	The service Connect opens an application relationship.
Read.req	FSPM	AREP SlotNumber SubslotNumber Index SeqNumber Length	The service Read requests the value of record data.

Primitive name	Source	Associated parameters	Functions
Write.req	FSPM	AREP SlotNumber SubslotNumber Index SeqNumber Length Data	The service Write conveys the value of record data.
Abort.req	FSPM	AREP	The service Abort terminates the AR.
DControl.req	FSPM	AREP ControlBlockConnect	The service DControl conveys end of parameter data.

Table 354 shows the service primitives including their associated parameters issued by CMCTL and received by FSPM of the IO controller (FSPMCTL).

Table 354 — Primitives issued by CMCTL to FSPMCTL

Primitive name	Source	Associated parameters	Functions
Release.cnf(+)	CM	AREP	This service primitive indicates that the Release of an application relationship succeeded.
Release.cnf (-)	CM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive indicates that the Release of an application relationship failed.
Connect.cnf(+)	CM	AREP	This service primitive is a positive confirmation to establish an application relationship.
Connect.cnf(-)	CM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative confirmation and the requested application relationship is not established.
Read.cnf(+)	CM	AREP Length Data	This service primitive indicates the positive confirmation of the Read service.
Read.cnf(-)	CM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative confirmation of the Read service.
Write.cnf(+)	CM	AREP	This service primitive indicates the positive confirmation of the Write service.
Write.cnf(-)	CM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative confirmation of the Write service.
CControl.rsp(-)	CM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative response of the application ready service.
DControl.cnf(-)	CM	AREP ErrorDecode ErrorCode1 ErrorCode2 AddData1 AddData2	This service primitive is a negative confirmation of the end of parameter service.
New_Input_Data.ind	CM	AREP	This service primitive indicates the reception of new data.
InData.ind	CM	AREP	This service primitive indicates the first reception of new data.

5.6.11.1.2 Parameters of CMCTL primitives

The parameters used with the primitives exchanged between the FSPM and the CMCTL are described in PROFINET IO Service Definition.

5.6.11.2 State machine description

The CM Controller protocol machine is present for every AR of an IO controller.

5.6.11.3 CMCTL state table

Table 355 contains the complete description of the CMCTL state machine.

Table 355 — CMCTL state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	CM_CTL_Init.req(AREP) =>	CONNECT
2	CONNECT	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => Release_Pending := FALSE Store Args CREP.PPM := create (PPM) CREP.CPM := create (CPM) CREP.ALPMR := create (ALPMR) RM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq)	W_CCON
3	CONNECT	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	CONNECT
4	CONNECT	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	CONNECT
5	CONNECT	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	CONNECT
6	CONNECT	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	CONNECT
7	CONNECT	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	CONNECT
8	CONNECT	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	CONNECT
9	CONNECT	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	CONNECT

#	Current State	Event /Condition =>Action	Next State
10	CONNECT	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	CONNECT
11	CONNECT	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	CONNECT
12	CONNECT	RM_DControl.cnf(+)(AREP, ControlBlock) =>	CONNECT
13	CONNECT	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	CONNECT
14	CONNECT	CM_Release.req(AREP, ControlBlock) => CM_Release.cnf(+)(AREP, ControlBlock)	CONNECT
15	CONNECT	RM_Release.cnf(+)(AREP, ControlBlock) =>	CONNECT
16	CONNECT	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	CONNECT
17	CONNECT	CM_CControl.rsp(+)(AREP, ControlBlock) =>	CONNECT
18	CONNECT	CM_CControl.rsp(-)(AREP, ControlBlock) =>	CONNECT
19	CONNECT	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict RM_CControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	CONNECT
20	CONNECT	CM_Abort.req(AREP) => CM_Abort.cnf(AREP)	CONNECT
21	W_CCON	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CCON
22	W_CCON	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) => Store Result StartTon := TRUE CPM_Activate.req(CREP.CPM, DA, SA, FrameID, Prio, VLAN, StartTon, Timeout_Base, WatchdogFactor, DataHoldFactor, Default_Value, Default_Status)	W_CPM_O

#	Current State	Event /Condition =>Action	Next State
23	W_CCON	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	CONNECT
24	W_CCON	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CCON
25	W_CCON	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	W_CCON
26	W_CCON	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CCON
27	W_CCON	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CCON
28	W_CCON	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	W_CCON
29	W_CCON	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CCON
30	W_CCON	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CCON
31	W_CCON	RM_DControl.cnf(+)(AREP, ControlBlock) =>	W_CCON
32	W_CCON	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CCON
33	W_CCON	CM_Release.req(AREP, ControlBlock) => Release_Pending := TRUE RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
34	W_CCON	RM_Release.cnf(+)(AREP, ControlBlock) =>	W_CCON
35	W_CCON	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CCON
36	W_CCON	CM_CControl.rsp(+)(AREP, ControlBlock) =>	W_CCON

#	Current State	Event /Condition =>Action	Next State
37	W_CCON	CM_CCControl.rsp(-)(AREP, ControlBlock) =>	W_CCON
38	W_CCON	RM_CCControl.ind(AREP, ControlBlock, ModulDiffBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict RM_CCControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CCON
39	W_CCON	CM_Abort.req (AREP) => RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
40	W_CPM_O	CPM_Activate.cnf(CREP.CPM) => PPM_Activate.req(CREP.PPM, DA, SA, FrameID, Prio, VLAN, SoftSync, TxOption, ReductionRatio, Phase, Sequence, Default_Values, Default_Status)	W_PPM_O
41	W_PPM_O	PPM_Activate.cnf(CREP.PPM) => ALPMR_Activate.req(CREP.ALPMR, DA, SA, VLAN, RTTimeoutFactor, MRetry)	W_ALPMR_O
42	W_ALPMR_O	ALPMR_Activate.cnf(CREP.ALPMR) /Prm > 0 => Prm_c := Prm – 1 Get Write Parameter from ListOfWriteParameter RM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	W_PCON
43	W_ALPMR_O	ALPMR_Activate.cnf(CREP.ALPMR) /Prm = 0 => ControlBlockConnect.PrmEnd := TRUE ArdyFlag := FALSE RM_DControl.req(AREP, ControlBlock)	W_ECON
44	W_PCON	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PCON
45	W_PCON	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_PCON
46	W_PCON	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_PCON
47	W_PCON	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PCON

#	Current State	Event /Condition =>Action	Next State
48	W_PCON	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) /Prm > 0 => Store Result Prm_c := Prm_c – 1 Get Write Parameter from ListOfWriteParameter RM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	W_PCON
49	W_PCON	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) /Prm = 0 => Store Result ControlBlockConnect.PrmEnd := TRUE ArdyFlag := FALSE RM_DControl.req(AREP, ControlBlock)	W_ECON
50	W_PCON	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) /Prm > 0 => Store Result Prm_c := Prm_c – 1 Get Write Parameter from ListOfWriteParameter RM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	W_PCON
51	W_PCON	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) /Prm = 0 => Store Result ControlBlockConnect.PrmEnd := TRUE ArdyFlag := FALSE RM_DControl.req(AREP, ControlBlock)	W_ECON
52	W_PCON	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PCON
53	W_PCON	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	W_PCON
54	W_PCON	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_PCON
55	W_PCON	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PCON
56	W_PCON	RM_DControl.cnf(+)(AREP, ControlBlock) =>	W_PCON
57	W_PCON	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_PCON

#	Current State	Event /Condition =>Action	Next State
58	W_PCON	CM_Release.req(AREP, ControlBlock) => Release_Pending := TRUE RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
59	W_PCON	RM_Release.cnf(+) (AREP, ControlBlock) =>	W_PCON
60	W_PCON	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_PCON
61	W_PCON	CM_CControl.rsp(+)(AREP, ControlBlock) =>	W_PCON
62	W_PCON	CM_CControl.rsp(-)(AREP, ControlBlock) =>	W_PCON
63	W_PCON	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict RM_CControl.rsp(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_PCON
64	W_PCON	CM_Abort.req (AREP) => RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
65	W_PCON	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Cancel.req(AREP, CancelSeqNumber) CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
66	W_PCON	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) =>	W_PCON
67	W_PCON	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Cancel.req(AREP, CancelSeqNumber) CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
68	W_ECON	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ECON
69	W_ECON	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_ECON
70	W_ECON	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ECON

#	Current State	Event /Condition =>Action	Next State
71	W_ECON	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ECON
72	W_ECON	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	W_ECON
73	W_ECON	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ECON
74	W_ECON	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ECON
75	W_ECON	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	W_ECON
76	W_ECON	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ECON
77	W_ECON	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ECON
78	W_ECON	RM_DControl.cnf(+)(AREP, ControlBlock) /!ArdyFlag =>	W_ARDY
79	W_ECON	RM_DControl.cnf(+)(AREP, ControlBlock) /ArdyFlag => ARBlockRes := Stored Results, ListOfIOCRBlockRes := Stored Results, AlarmCRBlockRes := Stored Results, ModuleDiffBlock := Stored Results RM_CCControl.rsp(+)(AREP, ControlBlock) CM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	W_IDATA
80	W_ECON	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Abort.req(AREP) CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	ABORT
81	W_ECON	CM_Release.req(AREP, ControlBlock) => Release_Pending := TRUE RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
82	W_ECON	RM_Release.cnf(+)(AREP, ControlBlock) =>	W_ECON

#	Current State	Event /Condition =>Action	Next State
83	W_ECON	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ECON
84	W_ECON	CM_CControl.rsp(+)(AREP, ControlBlock) =>	W_ECON
85	W_ECON	CM_CControl.rsp(-)(AREP, ControlBlock) =>	W_ECON
86	W_ECON	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /ARDY => ArdyFlag := TRUE Store Result RM_CControl.rsp(+)(AREP, ControlBlock)	W_ECON
87	W_ECON	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /!ARDY => CM_CControl.ind(AREP, ControlBlock, ModulDiffBlock)	W_ECON
88	W_ECON	CM_Abort.req (AREP) => RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
89	W_ECON	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Cancel.req(AREP, CancelSeqNumber) CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
90	W_ECON	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) =>	W_ECON
91	W_ECON	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Cancel.req(AREP, CancelSeqNumber) CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
92	W_ARDY	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ARDY
93	W_ARDY	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_ARDY
94	W_ARDY	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ARDY

#	Current State	Event /Condition =>Action	Next State
95	W_ARDY	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ARDY
96	W_ARDY	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	W_ARDY
97	W_ARDY	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ARDY
98	W_ARDY	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ARDY
99	W_ARDY	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	W_ARDY
100	W_ARDY	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ARDY
101	W_ARDY	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_ARDY
102	W_ARDY	RM_DControl.cnf(+)(AREP, ControlBlock) =>	W_ARDY
103	W_ARDY	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ARDY
104	W_ARDY	CM_Release.req(AREP, ControlBlock) => RM_Release.req(AREP, ControlBlock)	W_RCON
105	W_ARDY	RM_Release.cnf(+)(AREP, ControlBlock) =>	W_ARDY
106	W_ARDY	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_ARDY
107	W_ARDY	CM_CControl.rsp(+)(AREP, ControlBlock) =>	W_ARDY
108	W_ARDY	CM_CControl.rsp(-)(AREP, ControlBlock) =>	W_ARDY
109	W_ARDY	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /ARDY => RM_CControl.rsp(+)(AREP, ControlBlock)	W_IDATA

#	Current State	Event /Condition =>Action	Next State
110	W_ARDY	RM_CCControl.ind(AREP, ControlBlock, ModulDiffBlock) //!ARDY => CM_CCControl.ind(AREP, ControlBlock, ModulDiffBlock)	W_ARDY
111	W_ARDY	CM_Abort.req (AREP) => RM_Abort.req(AREP)	ABORT
112	W_ARDY	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Abort.req(AREP)	ABORT
113	W_ARDY	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) =>	W_ARDY
114	W_ARDY	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Abort.req(AREP)	ABORT
115	W_IDATA	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDATA
116	W_IDATA	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_IDATA
117	W_IDATA	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_IDATA
118	W_IDATA	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDSND
119	W_IDATA	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	W_IDATA
120	W_IDATA	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_IDATA
121	W_IDATA	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => RM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	W_IDSND
122	W_IDATA	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	W_IDATA

#	Current State	Event /Condition =>Action	Next State
123	W_IDATA	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_IDATA
124	W_IDATA	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDATA
125	W_IDATA	RM_DControl.cnf(+)(AREP, ControlBlock) => CM_DControl.cnf(+)(AREP, ControlBlock)	W_IDATA
126	W_IDATA	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDATA
127	W_IDATA	CM_Release.req(AREP, ControlBlock) => RM_Release.req(AREP, ControlBlock)	W_RCON
128	W_IDATA	RM_Release.cnf(+)(AREP, ControlBlock) =>	W_IDATA
129	W_IDATA	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_IDATA
130	W_IDATA	CM_CControl.rsp(+)(AREP, ControlBlock) => RM_CControl.rsp(+)(AREP, ControlBlock)	W_IDATA
131	W_IDATA	CM_CControl.rsp(-)(AREP, ControlBlock) => RM_CControl.rsp(-)(AREP, ControlBlock)	W_IDATA
132	W_IDATA	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /ARDY => RM_CControl.rsp(+)(AREP, ControlBlock)	W_IDATA
133	W_IDATA	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /!ARDY => CM_CControl.ind(AREP, ControlBlock, ModulDiffBlock)	W_IDATA
134	W_IDATA	CM_Abort.req (AREP) => RM_Abort.req(AREP)	ABORT
135	W_IDATA	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Abort.req(AREP)	ABORT
136	W_IDATA	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) => CM_InData.ind(AREP)	DATA
137	W_IDATA	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Abort.req(AREP)	ABORT

#	Current State	Event /Condition =>Action	Next State
138	W_IDSND	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDSND
139	W_IDSND	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_IDSND
140	W_IDSND	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_IDSND
141	W_IDSND	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDSND
142	W_IDSND	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) => CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	W_IDATA
143	W_IDSND	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDATA
144	W_IDSND	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDSND
145	W_IDSND	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	W_IDATA
146	W_IDSND	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDATA
147	W_IDSND	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDSND
148	W_IDSND	RM_DControl.cnf(+)(AREP, ControlBlock) => CM_DControl.cnf(+)(AREP, ControlBlock)	W_IDSND
149	W_IDSND	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_IDSND

#	Current State	Event /Condition =>Action	Next State
150	W_IDSND	CM_Release.req(AREP, ControlBlock) => Release_Pending := TRUE RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
151	W_IDSND	RM_Release.cnf(+) (AREP, ControlBlock) =>	W_IDSND
152	W_IDSND	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_IDSND
153	W_IDSND	CM_CControl.rsp(+)(AREP, ControlBlock) => RM_CControl.rsp(+)(AREP, ControlBlock)	W_IDSND
154	W_IDSND	CM_CControl.rsp(-)(AREP, ControlBlock) => RM_CControl.rsp(-)(AREP, ControlBlock)	W_IDSND
155	W_IDSND	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /ARDY => RM_CControl.rsp(+)(AREP, ControlBlock)	W_IDSND
156	W_IDSND	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /!ARDY => CM_CControl.ind(AREP, ControlBlock, ModulDiffBlock)	W_IDSND
157	W_IDSND	CM_Abort.req (AREP) => RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
158	W_IDSND	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
159	W_IDSND	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) => CM_InData.ind(AREP)	DSND
160	W_IDSND	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
161	DATA	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
162	DATA	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	DATA
163	DATA	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	DATA

#	Current State	Event /Condition =>Action	Next State
164	DATA	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => RM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data)	DSND
165	DATA	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	DATA
166	DATA	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	DATA
167	DATA	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => RM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length)	DSND
168	DATA	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	DATA
169	DATA	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	DATA
170	DATA	CM_DControl.req(AREP, ControlBlock) => RM_DControl.req(AREP, ControlBlock)	DATA
171	DATA	RM_DControl.cnf(+)(AREP, ControlBlock) => CM_DControl.cnf(+)(AREP, ControlBlock)	DATA
172	DATA	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
173	DATA	CM_Release.req(AREP, ControlBlock) => RM_Release.req(AREP, ControlBlock)	W_RCON
174	DATA	RM_Release.cnf(+)(AREP, ControlBlock) =>	DATA
175	DATA	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	DATA
176	DATA	CM_CControl.rsp(+)(AREP, ControlBlock) => RM_CControl.rsp(+)(AREP, ControlBlock)	DATA
177	DATA	CM_CControl.rsp(-)(AREP, ControlBlock) => RM_CControl.rsp(-)(AREP, ControlBlock)	DATA
178	DATA	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /ARDY => ARBlockRes := Stored Results, ListOfIOCRBlockRes := Stored Results, AlarmCRBlockRes := Stored Results, ModuleDiffBlock := Stored Results RM_CControl.rsp(+)(AREP, ControlBlock) CM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	DATA
179	DATA	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /!ARDY => CM_CControl.ind(AREP, ControlBlock, ModulDiffBlock)	DATA

#	Current State	Event /Condition =>Action	Next State
180	DATA	CM_Abort.req (AREP) => RM_Abort.req(AREP)	ABORT
181	DATA	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Abort.req(AREP)	ABORT
182	DATA	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) => CM_New_Input_Data.ind(AREP)	DATA
183	DATA	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Abort.req(AREP)	ABORT
184	DSND	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DSND
185	DSND	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	DSND
186	DSND	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	DSND
187	DSND	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DSND
188	DSND	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) => CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	DATA
189	DSND	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
190	DSND	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DSND
191	DSND	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	DATA

#	Current State	Event /Condition =>Action	Next State
192	DSND	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DATA
193	DSND	CM_DControl.req(AREP, ControlBlock) => RM_DControl.req(AREP, ControlBlock)	DSND
194	DSND	RM_DControl.cnf(+) (AREP, ControlBlock) => CM_DControl.cnf(+) (AREP, ControlBlock)	DSND
195	DSND	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	DSND
196	DSND	CM_Release.req(AREP, ControlBlock) => Release_Pending := TRUE RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
197	DSND	RM_Release.cnf(+) (AREP, ControlBlock) =>	DSND
198	DSND	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	DSND
199	DSND	CM_CControl.rsp(+)(AREP, ControlBlock) => RM_CControl.rsp(+)(AREP, ControlBlock)	DSND
200	DSND	CM_CControl.rsp(-)(AREP, ControlBlock) => RM_CControl.rsp(-)(AREP, ControlBlock)	DSND
201	DSND	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /ARDY => ARBlockRes := Stored Results, ListOfIOCRBlockRes := Stored Results, AlarmCRBlockRes := Stored Results, ModuleDiffBlock := Stored Results RM_CControl.rsp(+) (AREP, ControlBlock) CM_Connect.cnf(+) (AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock)	DSND
202	DSND	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) /!ARDY => CM_CControl.ind(AREP, ControlBlock, ModulDiffBlock)	DSND
203	DSND	CM_Abort.req (AREP) => RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
204	DSND	CPM_Stop.ind (CREP.CPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := time out RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL
205	DSND	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) => CM_New_Input_Data.ind(AREP)	DSND
206	DSND	PPM_Error.ind(CREP.PPM) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := no data send RM_Cancel.req(AREP, CancelSeqNumber)	W_CANCEL

#	Current State	Event /Condition =>Action	Next State
207	W_CANCEL	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
208	W_CANCEL	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_CANCEL
209	W_CANCEL	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CANCEL
210	W_CANCEL	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
211	W_CANCEL	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) => CM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2)	W_CANCEL
212	W_CANCEL	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
213	W_CANCEL	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
214	W_CANCEL	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) => CM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data)	W_CANCEL
215	W_CANCEL	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
216	W_CANCEL	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
217	W_CANCEL	RM_DControl.cnf(+)(AREP, ControlBlock) =>	W_CANCEL
218	W_CANCEL	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CANCEL

#	Current State	Event /Condition =>Action	Next State
219	W_CANCEL	CM_Release.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Release.cnf (-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_CANCEL
220	W_CANCEL	RM_Release.cnf(+)(AREP, ControlBlock) =>	W_CANCEL
221	W_CANCEL	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_CANCEL
222	W_CANCEL	CM_CControl.rsp(+)(AREP, ControlBlock) =>	W_CANCEL
223	W_CANCEL	CM_CControl.rsp(-)(AREP, ControlBlock) =>	W_CANCEL
224	W_CANCEL	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) =>	W_CANCEL
225	W_CANCEL	RM_Cancel.cnf(AREP) /Release_Pending => RM_Release.req(AREP, ControlBlock)	W_RCON
226	W_CANCEL	RM_Cancel.cnf(AREP) /!Release_Pending => RM_Abort.req(AREP)	ABORT
227	W_CANCEL	CM_Abort.req(AREP) =>	W_CANCEL
228	W_CANCEL	CPM_Stop.ind(CREP.CPM) =>	W_CANCEL
229	W_CANCEL	CPM_New_Cons_Data.ind(CREP.CPM, APDU_Status) =>	W_CANCEL
230	W_CANCEL	PPM_Error.ind(CREP.PPM) =>	W_CANCEL
231	W_RCON	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RCON
232	W_RCON	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	W_RCON
233	W_RCON	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_RCON

#	Current State	Event /Condition =>Action	Next State
234	W_RCON	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RCON
235	W_RCON	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	W_RCON
236	W_RCON	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_RCON
237	W_RCON	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RCON
238	W_RCON	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	W_RCON
239	W_RCON	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_RCON
240	W_RCON	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RCON
241	W_RCON	RM_DControl.cnf(+)(AREP, ControlBlock) =>	W_RCON
242	W_RCON	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	W_RCON
243	W_RCON	CM_Release.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	W_RCON
244	W_RCON	RM_Release.cnf(+)(AREP, ControlBlock) => RM_Release.cnf(+)(AREP, ControlBlock) RM_Abort.req(AREP)	ABORT
245	W_RCON	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) RM_Abort.req(AREP)	ABORT
246	W_RCON	CM_CControl.rsp(+)(AREP, ControlBlock) =>	W_RCON

#	Current State	Event /Condition =>Action	Next State
247	W_RCON	CM_CCControl.rsp(-)(AREP, ControlBlock) =>	W_RCON
248	W_RCON	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) =>	W_RCON
249	W_RCON	CM_Abort.req (AREP) =>	W_RCON
250	W_RCON	CPM_Stop.ind (CREP.CPM) =>	W_RCON
251	W_RCON	CPM_New_Cons_Data.ind (CREP.CPM, APDU_Status) =>	W_RCON
252	W_RCON	PPM_Error.ind(CREP.PPM) =>	W_RCON
253	ABORT	CM_Connect.req(AREP, ARBlockReq, ListOfIOCRBlockReq, AlarmCRBlockReq, ListOfExpectedSubmoduleBlockReq, PrmCount, ListOfWriteBlockReq) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	ABORT
254	ABORT	RM_Connect.cnf(+)(AREP, ARBlockRes, ListOfIOCRBlockRes, AlarmCRBlockRes, ModuleDiffBlock) =>	ABORT
255	ABORT	RM_Connect.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	ABORT
256	ABORT	CM_Write.req(AREP, SlotNumber, SubslotNumber, Index, SeqNumber, Length, Data) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	ABORT
257	ABORT	RM_Write.cnf(+)(AREP, Seq Number, AddData1, AddData2) =>	ABORT
258	ABORT	RM_Write.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	ABORT
259	ABORT	CM_Read.req(AREP, API, TargetUUID, SlotNumber, SubslotNumber, Index, SeqNumber, Length) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	ABORT
260	ABORT	RM_Read.cnf(+)(AREP, Seq Number, AddData1, AddData2, Length, Data) =>	ABORT

#	Current State	Event /Condition =>Action	Next State
261	ABORT	RM_Read.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	ABORT
262	ABORT	CM_DControl.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	ABORT
263	ABORT	RM_DControl.cnf(+)(AREP, ControlBlock) =>	ABORT
264	ABORT	RM_DControl.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) =>	ABORT
265	ABORT	CM_Release.req(AREP, ControlBlock) => ErrorDecode:= PNIO ErrorCode1 := CMCTL ErrorCode2 := state conflict CM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2)	ABORT
266	ABORT	RM_Release.cnf(+)(AREP, ControlBlock) => RM_Release.cnf(+)(AREP, ControlBlock) RM_Abort.req(AREP)	ABORT
267	ABORT	RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) => RM_Release.cnf(-)(AREP, ErrorDecode, ErrorCode1, ErrorCode2, AddData1, AddData2) RM_Abort.req(AREP)	ABORT
268	ABORT	CM_CControl.rsp(+)(AREP, ControlBlock) =>	ABORT
269	ABORT	CM_CControl.rsp(-)(AREP, ControlBlock) =>	ABORT
270	ABORT	RM_CControl.ind(AREP, ControlBlock, ModulDiffBlock) =>	ABORT
271	ABORT	CM_Abort.req(AREP) =>	ABORT
272	ABORT	CPM_Stop.ind(CREP.CPM) =>	ABORT
273	ABORT	CPM_New_Cons_Data.ind(CREP.CPM, APDU_Status) =>	ABORT
274	ABORT	PPM_Error.ind(CREP.PPM) =>	ABORT
275	ABORT	RM_Abort.cnf(AREP) => CPM_Close.cnf(CREP.CPM)	W_CPM_C
276	W_CPM_C	CPM_Close.cnf(CREP.CPM) => PPM_Close.req(CREP.PPM)	W_PPM_C

#	Current State	Event /Condition =>Action	Next State
277	W_PPM_C	PPM_Close.cnf(CREP.PPM) => ALPMR_Close.req(CREP.ALPMR)	W_ALPMR_C
278	W_ALPMR_C	ALPMR_Close.cnf(CREP.ALPMR) => remove (CREP.PPM) remove (CREP.CPM) remove (CREP.ALPMR)	CONECT

5.6.11.4 Functions

There are no functions or macros defined for CMCTL.

5.7 RPC

5.7.1 Primitive definitions

5.7.1.1 Primitives exchanged between RPC and NRPM

Table 356 shows the primitives issued by the NRPM to the RPC.

Table 356 — Primitives issued by NRPM to RPC

Primitive name	Source	Associated parameters	Functions
RPC_Alloc.req	NRPM	Arg	This is the service to allocate memory.
RPC_CControl.rsp(-)	NRPM	Arg	This is the service to convey the negative CControl response.
RPC_CControl.rsp(+)	NRPM	Arg	This is the service to convey the positive CControl response.
RPC_Connect.req	NRPM	Arg	This is the service to convey the Connect request.
RPC_DControl.req	NRPM	Arg	This is the service to convey the DControl request.
RPC_Read.req	NRPM	Arg	This is the service to convey the Read request.
RPC_Release.req	NRPM	Arg	This is the service to convey the Release request.
RPC_Write.req	NRPM	Arg	This is the service to convey the Write request.

Table 357 shows the primitives issued by the RPC to the NRPM.

Table 357 — Primitives issued by RPC to NRPM

Primitive name	Source	Associated parameters	Functions
RPC_CControl.ind	RPC	Arg	This is the service to convey the CControl indication.
RPC_Connect.cnf(-)	RPC	Arg	This is the service to convey the negative Connect confirmation.
RPC_Connect.cnf(+)	RPC	Arg	This is the service to convey the positive Connect confirmation.
RPC_DControl.cnf(-)	RPC	Arg	This is the service to convey the negative DControl confirmation.
RPC_DControl.cnf(+)	RPC	Arg	This is the service to convey the positive DControl confirmation.
RPC_Read.cnf(-)	RPC	Arg	This is the service to convey the negative Read confirmation.
RPC_Read.cnf(+)	RPC	Arg	This is the service to convey the positive Read confirmation.
RPC_Release.cnf(-)	RPC	Arg	This is the service to convey the negative Release confirmation.
RPC_Release.cnf(+)	RPC	Arg	This is the service to convey the positive Release confirmation.
RPC_Write.cnf(-)	RPC	Arg	This is the service to convey the negative Write confirmation.
RPC_Write.cnf(+)	RPC	Arg	This is the service to convey the positive Write confirmation.

5.8 DLL Mapping Protocol Machines (DMPMs)

5.8.1 Overview

The DLL Mapping Protocol consists of several protocol machines:

- Clock and time synchronization (GenSync) including
 - the Receive Control (RCTL) machine for delay measurement
 - the Time Master Management (TMM) machine for clock synchronization
- Handling of the services invoked by other Application Layer entities (LMPM)
- Forwarding for RT class 3, synchronization frames, and other frames (IR-FW, SYN-FW, Relay)
- Mapping to Media Access Control interface (MMAC) and send/receive functions for synchronization messages

The Interface between the service handler (LMPM) and Media Access Control interface (MMAC) consists of a set of Queues and a Data-Resource. The IR-FW uses queues.

Figure 53 illustrates the structuring of the protocol machines for the DMPM for a device with a single port.

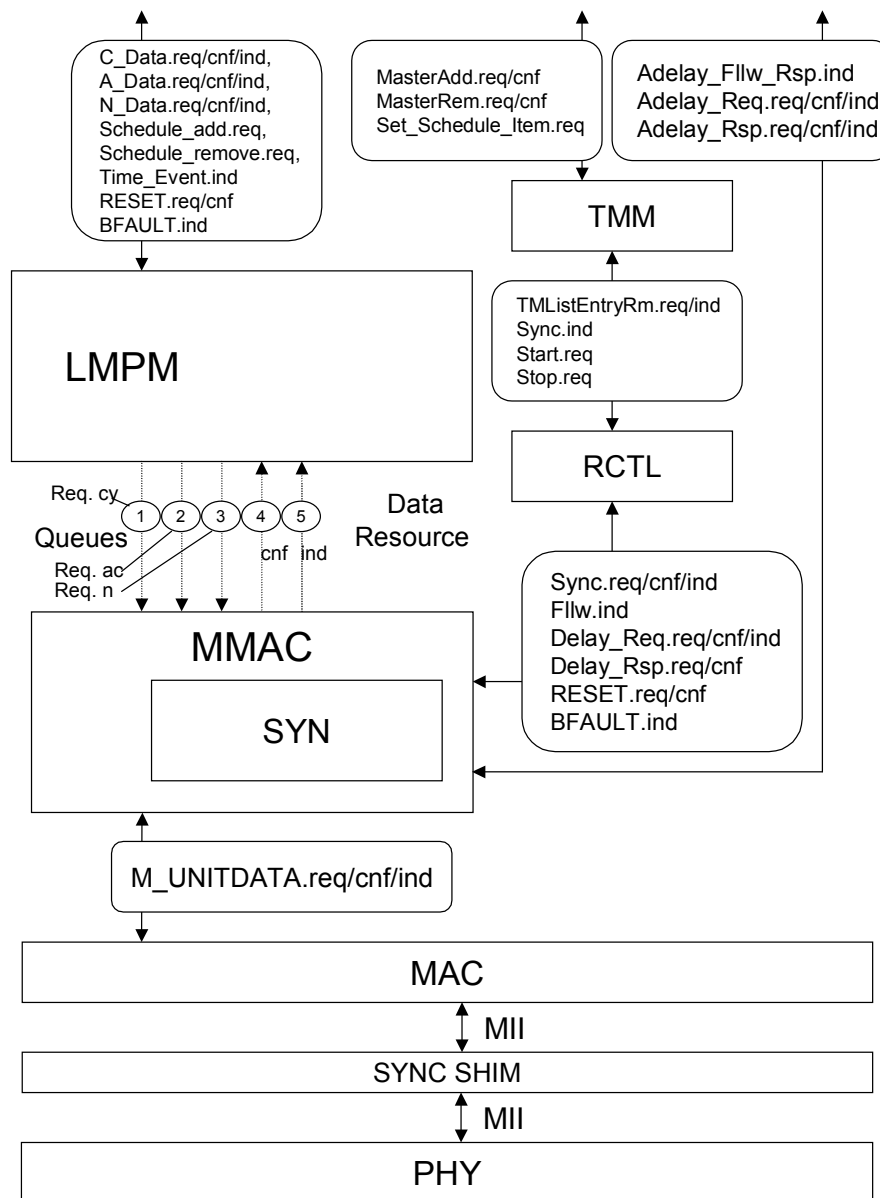


Figure 53 — Structuring of the protocol machines within the DMPM (single port)

Figure 54 illustrates the structuring of the protocol machines for the DMPM for a bridging device.

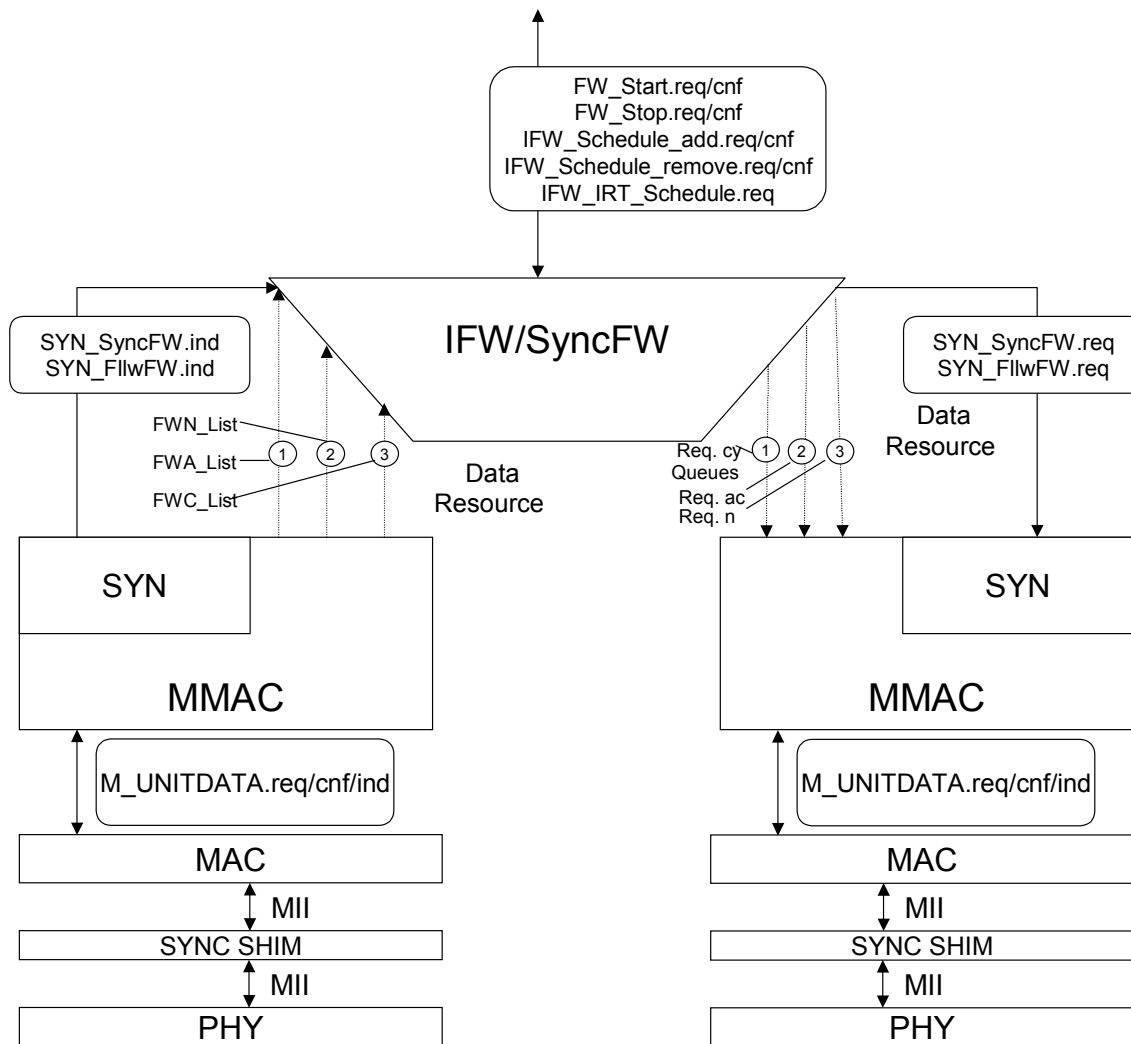


Figure 54 — Structuring of the protocol machines within the DMPM (bridge)

5.8.2 Generic Synchronization

5.8.2.1 Synchronization

The synchronization is done with a Sync frame as described in PTP **with additional fields for delay calculation**. The synchronization information is split into two fields that form together the time base:

- A reference value to the master's time
- A delay value counting all delays from sampling at the source to destination

The type of reference value can be an absolute time source as GPS or a locally generated sync time. A DTE shall be within a single Sync domain. There can be several master providing absolute time information.

Sync and delay measurement can be handled separately in switched networks, but for optimization and control procedures sync shall be used as trigger for delay request. The line delay between 2 bridges in an Ethernet switched network is independent of the time master. This is different with shared media with several master at one port. This requires different delay values for different master.

A PTP bridge can detect non PTP neighbors by checking the line delay between nodes. As a switch delay is in the range of several (>2) us, the delay between two nodes connected with 100 m Twisted Pair cabling (100 Base TX) is about 500 ns. A point-to-point link measurement ensures to detect such errors.

The adaptation of a *PTP-slave* to that of the *PTP-master* is soft. This results in a control loop.

The synchronization scheme as described in Figure 56 requires transferring the exact sending time of the Sync as initial value of the delay (this is the time between sampling and sending out of Sync). Some implementations of ISO/IEC 8802-3 do not allow modifying a frame short before transmission. This can be avoided when the exact delay of a node will be sent in a second frame. This principle will be called follow up.

A clock synchronization master who is not able to put the exact sending time in a Sync frame will use follow up. Clock synchronization slaves do not need follow up.

If follow up is used, a Flag in the Sync frame will be set to indicate this. A sequence number shall be used to assign the follow up to the associated Sync.

5.8.2.2 Delay measurement

The delay measurement between peer nodes is accomplished by measuring the round trip of two adjacent DL-frames (SYNC and DELAY-REQ-frame). There are four time stamps needed to calculate the line delay (including the delays of the transceivers). The delay time measurement procedure between two nodes is described in Figure 55 in principle. The quartz tolerance can influence the precision of the delay measurement and shall be compensated to achieve the required accuracy.

NOTE1 The local time stamps have a possible inaccuracy due to the quartz tolerance. But this is negligible if the time between receiving and returning the time frame is short. If the local delay between Sync indication and Delay request is 100 µs and quartz deviation of transmitter and receiver is 100 PPM, the possible error is 10 ns. If the local delay is 10 ms, the possible error is 1 µs.

NOTE2 The diagram shows the principle flow of data and the information sampling procedures. Additional protocol elements may be needed to accomplish this task.

Other reasons are errors are inaccuracies in time stamping and non symmetrical delays between two stations. There may be other sources or combinations of the inaccuracies.

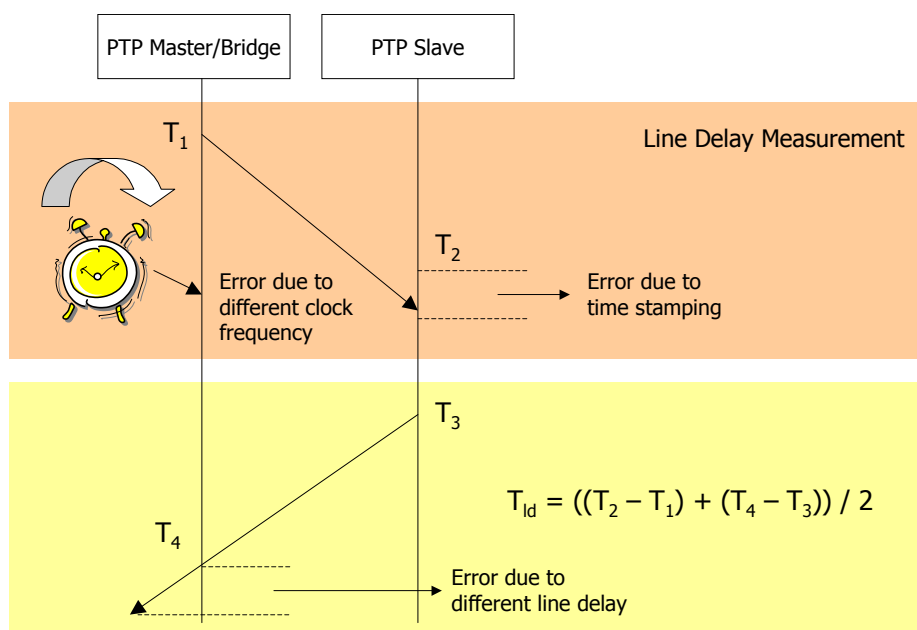


Figure 55 — Line delay measurement

The calculated line delay is stored and is therefore available for the line delay accumulation.

If a delay value is available it will be propagated to the peer node. This allows fast recovery in case of redundancy switchover procedures.

5.8.2.3 Sequence of Operation

Figure 56 shows the sequence of Operation.

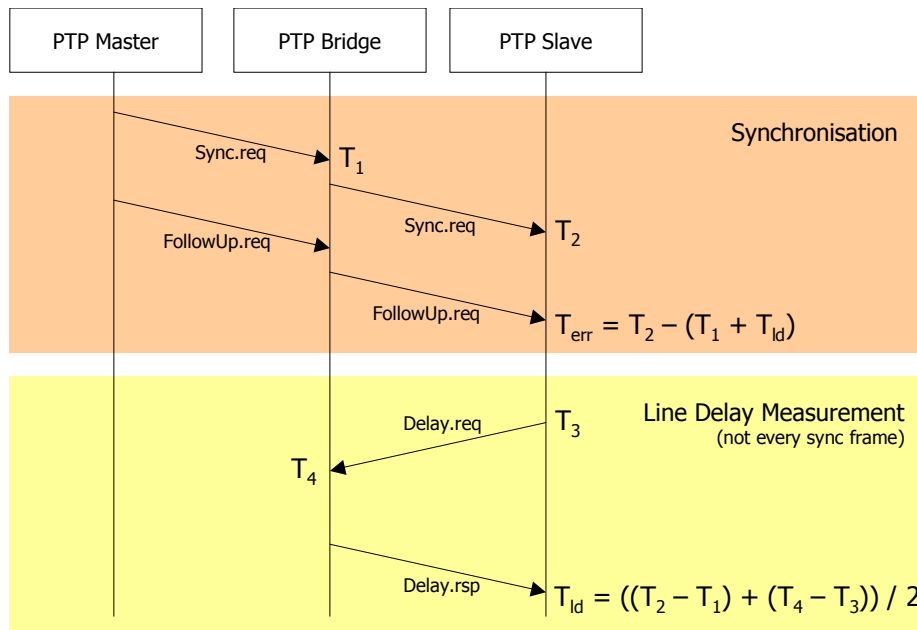


Figure 56 — Synchronization and line delay measurement

5.8.2.4 PTP Master Mode

A clock synchronization master DTE shall proceed as described in 5.8.2.3 and in Table 358.

Table 358 — Procedures of clock synchronization master

State	Event/ Condition	Action	Comment
Init	uct	Start Reference Counter Start Timeout Init	
Wait Activity	Better Master active	Start Timeout Init	
	Other Master with same stratum active	Start Timeout Init Adjust time	
	Time Adjusted or Timeout	Start Timeout Sync Sample time Start delay counter SYNC with sampled time + delay counter at send	
Operation	Timer expired	Start Timeout Sync Sample time Start delay counter SYNC with sampled time + delay counter at send	Sample Time & Start delay counter shall be atomic operations without unknown delays between them
	Timer expired – n SYNC completed	Issue link delay measurement together with SYNC	
	Link down		Continue with Init
	Better Master active		Continue with Init

The main issue is the best master clock algorithm which shall operate in the same way in every master.

5.8.2.5 PTP Slave Mode

A clock synchronization slave DTE can proceed as described in Table 359.

Table 359 — Example Procedures of clock synchronization slave

State	Event/ Condition	Action	Comment
Init		Start Reference Counter	

State	Event/ Condition	Action	Comment
Await Sync Frame	SYNC	Local1 = sample local timer Remote1 = sendCC from SYNC Start Timeout Sync Set Master Address	Sample local timer & SYNC receipt shall be atomic operations without unknown delays between them
Adjust Frequency Deviation	n SYNC (same Master Add)	Local2 = Sample local Timer Remote2 = sendCC from SYNC Set local time $\text{Drift} = (\text{Remote2} - \text{Remote1}) / (\text{Local2} - \text{Local1}) - 1$ Add a tic every 1 /Drift tics of the local clock (subtract if Drift is negative) Init Delay Request (time at send out is Local3)	Sample local timer & SYNC receipt shall be an atomic operation without unknown delays between them The distance between start and calculation shall be in a Range, that the possible maximum drift error is at least 10 times greater as a possible jitter (e.g. if drift is 100 PPM and jitter is 1µs a difference of 100ms or greater shall be used to make the calculation) The drift adjustment should be done at least every cycle to avoid additional sources of error
	SYNC (better Master)	Local1 = sample local timer Remote1 = sendCC from SYNC Start Timeout Sync Set Master Address	Sample local timer & SYNC receipt shall be atomic operations without unknown delays between them Continue with Adjust Frequency Deviation
	Timer expired		
Get Delay	Delay Response	Remote3 = rcvCC from DelayRes $\text{Delay} = ((\text{Local2} - \text{Remote2}) + (\text{Remote3} - \text{Local3})) / 2$	
Synchronized	SYNC(same Master Add)	Local2 = sample local timer Remote2 = sendCC from SYNC Adjust Drift Get Time = Absolute Time + Delay + Line Delay	
	SYNC (same Master Add w. DelayR)	Local2 = Sample local Timer Remote2 = sendCC from SYNC Init Delay Request (time at send out is Local3)	
	SYNC (better Master)	Local1 = sample local timer Remote1 = sendCC from SYNC Start Timeout Sync Set Master Address	Sample local timer & SYNC receipt shall be atomic operations without unknown delays between them Continue with Adjust Frequency Deviation
	Timer expired		Continue with Init

The adaptation of a PTP-slave to that of the PTP-master is soft. This results in a control loop. Follow up principle is not shown in the table (this may be handled at a subordinated driver).

5.8.2.6 Best Master Clock

A clock at startup listens for a time SYNC_RECEIPT_TIMEOUT.

A master clock (clock in the PTP_MASTER state) issues periodic Sync messages (period is called the sync_interval).

A master clock may receive Sync messages from other clocks (who for the moment think they are master) which it calls 'foreign masters'.

Each master clock uses the Best Master Clock algorithm to determine whether it should remain master or yield to a foreign master.

Each non-master clock uses the Best Master Clock algorithm to determine whether it should become a master or to select the Best Master

The decision is done in the following way:

- 1) 1A master clock 'A' can receive Sync messages from other potential master clocks-'B', 'C',...

- 2) Clock 'A' decides:
 - a) Which of the clocks 'B', 'C' ,... is the 'best' clock
 - b) Whether clock 'A' is better than the best of 'B', 'C' ,...
- 3) It does this by pair wise comparisons of the data sets describing each of the clocks using the Best Master Clock algorithm, BMC.
- 4) Based on the results of this comparison the BMC returns a recommended clock state: in simple situations either master or slave.
- 5) All clocks operate on the same information and therefore arrive at consistent results.
- 6) Data for these comparisons logically is maintained by each clock in one of several data sets.

The PER CLOCK data sets supported are:

- Default data set: Properties of the local clock that determine its behavior and performance when it is the grandmaster clock (only potential master)
- Global time properties data set: Time base properties
- Current data set: Current synchronization and topological operational properties

The PER CLOCK PORT data sets supported are:

- Parent data set: Properties of the parent and grandmaster
- Port configuration data set: Clock port properties
- Foreign master data set: Identification of Sync messages from potential master clocks-part of a qualification scheme to reduce thrashing

Principal items used by the BMC:

- Based on primary source of time, e.g. GPS, local oscillator...
- Accuracy
- Variance
- Preferred set membership
- Type: Boundary clock spans subnets or ordinary clock
- PTP-UUID

Main steps to decide the best master:

- a) Select the clock that derives its time from the better grandmaster
- b) If the grandmasters are equivalent choose the nearest grandmaster (this is an option which shall not used for synchronization schemes that need exactly one master in the system)
- c) If the above fail to indicate a choice use tiebreaking (PTP-UUID)

The hierarchy is:

- a. Preferred: designates a set of masters from which it is selected
- b. Stratum: primary or secondary standard
- c. Identifier: accuracy of clock's time base
- d. Variance: stability and noise of clock
- e. 'Closest': shall not apply for systems which need a single clock
- f. PTP-UUID tiebreaker

5.8.2.7 Bridge Protocols

5.8.2.7.1 Preconditions

Forwarding is active only if there is a valid delay measurement.

5.8.2.7.2 Time Forwarding

The principle of synchronization through a sequence of PTP-bridges can be seen in Figure 57. The node that originally places the time frames for synchronization in the network is known as the PTP-master. All other nodes that receive and/or pass on these frames are known as PTP-bridge or PTP-slave. A PTP-master uses a reference to a local time system or a global time source.

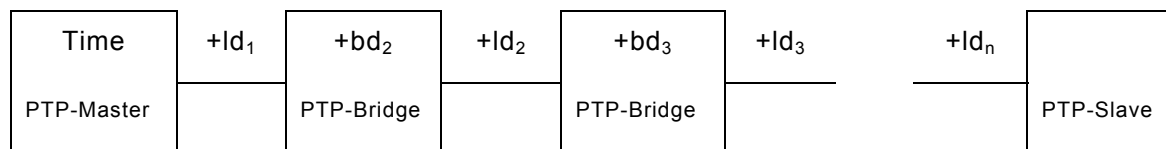


Figure 57 — Delay accumulation

The sync frames are forwarded like other frames in principle. In addition the time had to be adjusted in every bridge by:

- adding its internal bridge delay (bd -for handling time frames, refer to 5.8.2.8);
- and adding the line delay ld to the previous node (for line delay measurement, refer to 5.8.2.1),

to the `cc_del` field of the time frame as it passes through. Each PTP-bridge knows the delay of the time frame from the master. Since the time of arrival `cc_rcv` is also entered in the frame, it can also determine drift compared

- to the previous PTP-bridge,
- to the PTP-master.

The delay of a time message in a PTP bridge is only a critical issue for time synchronization if the delay within a PTP bridge is significant (several 100 us). Significant delay values should be corrected by a drift compensation to achieve high precision synchronization.

EXAMPLE

If a frame of 12000 Bittimes will be send out the delay of a subsequent time frame is about 120 μs at fast Ethernet speed. If there is a clock drift of 100 PPM and the PTP-Master has a clock drift of -100 PPM this would result in an error of 24ns. If there is no priority for PTP frames and there are 10 frames queued, this could result in an additional error of 240ns which may be unacceptable. That means that a large delay value means less precision of the delay value.

Each clock sampling at a node is always associated with a synchronization jitter. In addition to this, there is always control inaccuracy for each bridge. The maximum time difference between any two nodes of a network is dependent on the topology – cascading of bridges result in an inaccuracy depending on the level of cascading.

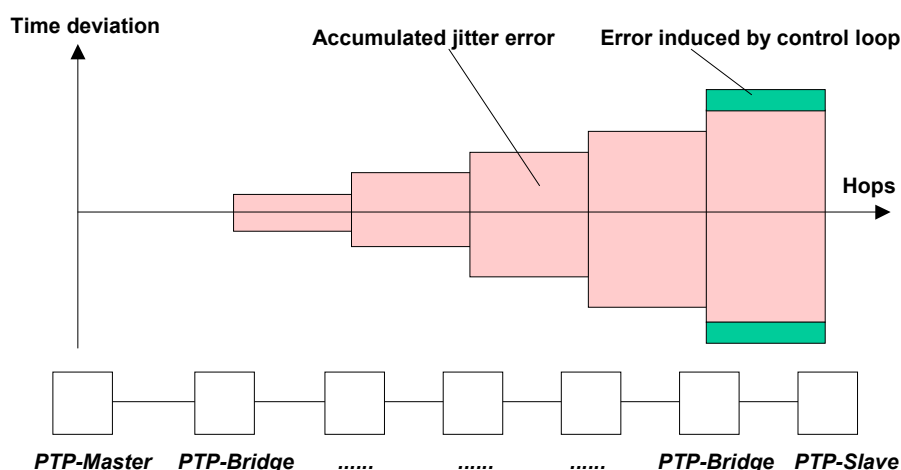


Figure 58 — Worst case Time deviation of Synchronization

The highest required accuracy for synchronization applies only to two adjacent nodes.

In a non-redundant system, only one time frame is used for synchronization. In redundant systems, two situations can be distinguished:

- Ring: The PTP-master exists only once. The synchronization is sent in all directions. PTP bridges forward the most recent one (identified by the sequence number). This will guarantee consistent sequence numbers needed for follow up.
- Redundant PTP-master : Each PTP master runs the Best Master Clock (BMC) Algorithm which selects one PTP master. Passive PTP master will monitor the active and restart their PTP Master activity on timeout. Inactive time master can send frames marked as being inactive to keep the schedule constant. Inactive frames are not used for control. To avoid time changes when there is a changeover to the time frames of the standby master, it is advisable to keep the number of PTP bridges between the PTP-master and the standby master as low as possible. To switch from one PTP Master to another PTP Master within a short period of time, the second Master shall use the drift and change it smoothly at a rate of 1 PPM per 10 s to avoid synchronization errors.

The two constellations can also be combined.

5.8.2.8 Processing Time Frames

This frame structure is an example and may not be mandatory for this approach although the placement of some fields determines the performance of the approach. All time frames should have a similar structure. It consists of a set of fields that allows a fast processing while forwarding.

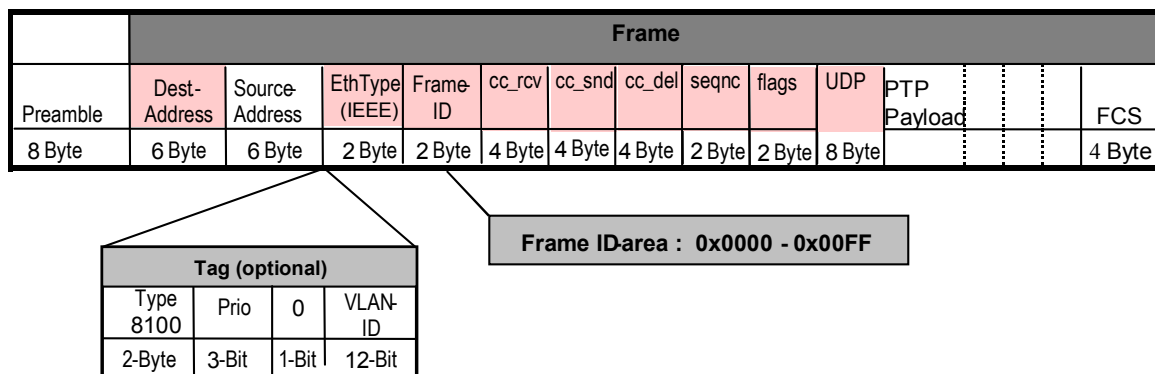


Figure 59 — Structure of a Time Frame

The fields are used in the following way:

Vlan Tag

Assigning priorities to the frame. A Priority of 6 would be recommended for PTP. The VLAN ID is not used in PTP.

EtherType

A field assigned by IEEE to identify the PTP protocol. Currently 0x8892 (PI Ethertype) is used.

FrameID

This field is used to identify the PTP frames and their version. For different use there are several ID reserved for Sync, FollowUp, DelayReq and DelayRsp. The version number provides a means for selecting a specific set.

cc_rcv, cc_snd, cc_del

These fields contain the clock counter values used within the protocol. Not all of these fields may be used in any service primitive.

The Field cc_del contains the accumulated delay of the sync frame in sync and follow up. If a follow up sequence is used a bridge may add the delay either in Sync or in FollowUp. This allows a flexible action for all types of systems. A time receiver has to add the cc_del values of Sync and Follow Up to adjust the time reference. DelayReq PDU may contain optionally the delay between receipt of Sync and issuing DelayReq.

The line delay between two nodes and the internal delay is measured by a 32-bit *clock_counter*. This resolution is a tradeoff between time range and accuracy. A small time range will limit the delay between PTP Master and PTP Slave and the counter frequency will

limit the precision of the clock signal. 100 MHz will produce an error of 2 clock ticks as error (one for Sync and one for Delay).

The calculation of the delay value at Sync reception and transmission is shown in Figure 60.

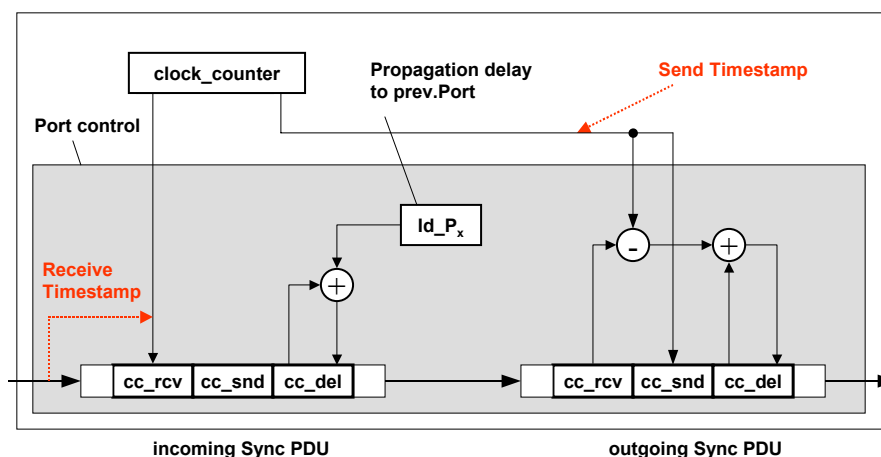


Figure 60 — Hardware Arrangement for Processing Sync PDU

$$cc\_del = cc\_snd - cc\_rcv + cc\_del + Pd\_Px \quad (5)$$

Seqnc

This field contains the sequence number and is used to assign Sync, FollowUp and Delay. Only FollowUp frames with match in Sequence number can be processed. The Sequence ID in the PTP payload is irrelevant in this case. The sequence number is incremented by the PTP master.

Flags

Fields of general interest are placed here (e.g. flag indicating follow up, Delay Measuring request).

Three 4 Bit fields are used to extend the cc_rcv, cc_snd, cc_del counters to a ns range.

Bit 0 – 3: cc_del_ext

Shall be used to express the ns of the cc_del field.

Allowed values 0 to 15.

Bit 4 – 7: cc_snd_ext

Shall be used to express the ns of the cc_snd field.

Allowed values 0 to 15.

Bit 8 – 11: cc_rcv_ext

Shall be used to express the ns of the cc_rcv field.

Allowed values 0 to 15.

Bit 12: FU

Shall be set to 0 to indicate that there is no follow up to this Sync.

Shall be set to 1 to indicate a follow up sequence.

Bit 13: DLY

Shall be set to 0 to indicate that there is no DelayReq required in this Sync interval.

Shall be set to 1 to request a DelayReq.

Bit 14-15: reserved

Shall be set to 0.

5.8.2.9 Start up

A sequence of operation is necessary. A Sync Bridge will enter forwarding state if there is a delay value to the way back to the Sync Master. This means a sequential Start Up starting at the time Master. At least one Cycle is needed to establish a link.

The sequence of Operation is shown in Figure 61. Sync can be done together with Follow Up. DRQ and DRS are Delay Request and Delay Response. The dotted line shows the activation of the forwarding procedure within a bridge.

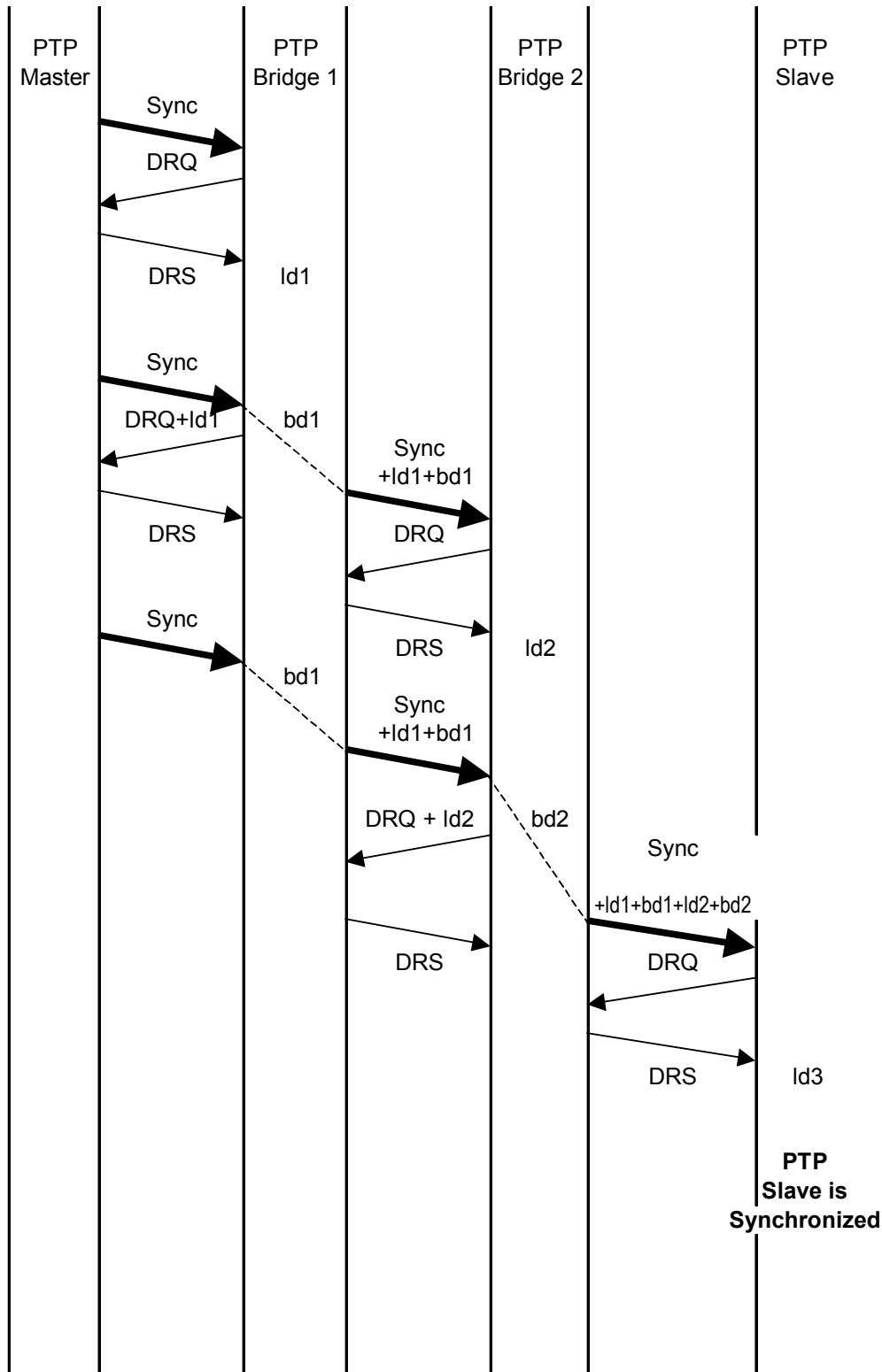


Figure 61 — Start up sequence

5.8.2.10 Error Recovery

A link down will delete the line delay. A time out will close the forwarding.

To deal with redundancy switchover in case of redundant paths, a Sync 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 Req and Delay Resp
- monitoring of Sync: allowed time between two subsequent Sync.

The timeout of the delay measurement shall be higher as the timeout of the time master and is calculated independently at every update of the 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 BMC algorithm will cancel the activities of a Sync master.

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

5.8.2.11 TMM

5.8.2.11.1 Primitive definitions

5.8.2.11.1.1 Primitives exchanged between FSPM and TMM

Table 360 shows all primitives issued by FSPM to the TMM.

Table 360 — Primitives issued by FSPM to TMM

Primitive Name	Source	Associated Parameters	Functions
TMM_MasterAdd.req		SA ProtVar MS_list MS_status	
TMM_MasterRem.req		SA ProtVar MS_list MS_status	
TMM_Set_Schedule_Item.req		Port Sched_list	

Table 361 shows all primitives issued by the TMM to the FSPM.

Table 361 — Primitives issued by TMM to the FSPM

Primitive Name	Source	Associated Parameters	Functions
TMM_MasterAdd.cnf	TMM	SA PortVar MS_list MS_status	
TMM_MasterRem.cnf	TMM	SA PortVar MS_list MS_status	

5.8.2.11.1.2 Primitives exchanged between RCTL and TMM

Table 362 shows all primitives issued by RCTL to the TMM.

Table 362 — Primitives issued by RCTL to TMM

Primitive Name	Source	Associated Parameters	Functions
RCTL_Sync.ind	RCTL	Port DA SA CS_list CS_status	
RCTL_TMListEntryRm.ind	RCTL	SA rt_id Status	

Table 363 shows all primitives issued by the TMM to the RCTL.

Table 363 — Primitives issued by TMM to the RCTL

Primitive Name	Source	Associated Parameters	Functions
RCTL_Start.req	TMM	-	

5.8.2.11.1.3 Primitives exchanged between TMM and other machines

Table 364 shows all primitives issued by other to the TMM.

Table 364 — Primitives issued by other machines to TMM

Primitive Name	Source	Associated Parameters	Functions
SYN_Sync.cnf	MMAC SYNC_SR	Port DA SA CS_list CS_status	

Table 365 shows all primitives issued by other to the TMM.

Table 365 — Primitives issued by TMM to other machines

Primitive Name	Source	Associated Parameters	Functions
IFW_IRT_Schedule.req	TMM	Port	
LMPM_IRT_Schedule.req	TMM	CREP	
LMPM_TI_Send_Clock.req	TMM	-	
MMAC_IRT_Set_Interval.req	TMM	Port Interv IntervLen	

5.8.2.11.1.4 Parameters of TMM primitives

Table 366 shows all parameters used with primitives between the FSPM, RCTL and the TMM.

Table 366 — Parameters used with primitives exchanged between FSPM, RCTL and TMM

Parameter Name	Description
CS_list	Cycle Sync List
CS_status	Cycle Sync Status
DA	Destination Address
MS_list	
MS_status	
Port	
ProtVar	Protocol variant for time synchronisation
rt_id	
SA	Source Address
Status	

5.8.2.11.2 State machine description

- MIDDLE: Master-Idle-State

PTP-Master monitors PTP.Domain for other active PTP-Master. If MIDELE (Master-Idle-Time) expires without Sync indications of PTP-Master with higher ranking this station starts sending Sync and enters state OPEN_M.

- DRIFT-A

PTP-Slave enters this state to measure drift from PTP-Master.

- PLLW-A

After drift compensation a deviation from PTP-Master time will be reduced to a certain level.

- OPEN_S

After drift and deviation compensation a PTP Slave enters OPEN_S. OPEN_S indicates that a PTP-Slave is synchronized with active PTP-Master.

- OPEN_ST

PTP-Slave detects PTP-Master change and tries to adopt this change (OPEN_ST means Open-Slave-Transient). This state can be left when adopted to the new Master (duration of a Sync-Control-Loop except permanent PTP-Master-Change)

- PLLW_T

PLLW_T (PLL-Window-Transient) will be used during PTP-Master change if a deviation is beyond a defined threshold.

- DRIFT_T

DRIFT_T (DRIFT-Transient) will be used during PTP-Master change if a drift is beyond a defined threshold.

In States PLLW_T and DRIFT_T PTP-Slave it is not synchronized to active PTP-Master in States PLLW_T and DRIFT_T.

- OPEN_MT

If a PTP-Slave becomes active PTP-Master it enters OPEN_M via OPEN_MT (Master-Transient). This state is used to compensate drift to PTP-Master. But the station itself sends already Sync frames.

- OPEN_M

OPEN_M (Master) PTP-Master sends cyclic Sync-Frames without drift adaption.

- t-state

states with prefix OPEN and the transient states DRIFT-T and PLLW-T (in these two states the PTP-Slave have an inconsistent time base compared to the active PTP-Master).

5.8.2.11.3 TMM state table

Table 367 contains the complete description of the TMM state table.

Table 367 — TMM state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	=> AMList.ActCntEntry := 0 AMList.SA := NIL AMList.ProtVar := NIL AMList.Dferr := 0 AMList.Pherr := 0 AMList.SLOOPT.ini AMList.MIDLET.ini AMList.DRCRRT.ini AMList.PLLWT.ini AMList.SYNCHT.ini AMList.TRANST.ini RCTL_Start.req ()	CLOSED
2	CLOSED	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /BM_SA := BMCsetRmo(SA, CS_list.RT_ID) && BM_SA != NIL && BM_SA != Lcoal_SA => AMList.SA := BM_SA AMList.PortVar := CS_list.RT_ID AMList.Dferr := 0 AMList.Pherr := 0 AMList.SLOOPT.start AMList.DFCRRT.start INIT_DFCRT_LOOP() INIT_PHCRT_LOOP()	DRIFT-A
3	CLOSED	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /BM_SA := BMCsetRmo(SA, CS_list.RT_ID) && BM_SA != NIL && BM_SA == Lcoal_SA => AMList.SA := BM_SA AMList.PortVar := CS_list.RT_ID AMList.Dferr := 0 AMList.Pherr := 0 AMList.SLOOPT.start AMList.MIDLET.start	MIDLE
4	CLOSED	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /BM_SA := BMCsetRmo(SA, CS_list.RT_ID) && BM_SA == NIL =>	CLOSED
5	CLOSED	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) /BM_SA := BMCsetLoc(SA, MS_list.RT_ID) && BM_SA != NIL && BM_SA != Local_SA => AMList.SA := BM_SA AMList.PortVar := CS_list.RT_ID AMList.Dferr := 0 AMList.Pherr := 0 AMList.SLOOPT.start AMList.DFCRRT.start INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	DRIFT-A

#	Current State	Event /Condition =>Action	Next State
6	CLOSED	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) /BM_SA := BMCsetLoc(SA, MS_list.RT_ID) && BM_SA != NIL && BM_SA == Local_SA => AMList.SA := Local_SA AMList.PortVar := MS_list.RT_ID AMList.Dferr := 0 AMList.Pherr := 0 AMList.SLOOPT.start BMList.MIDLET.start MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	MIDLE
7	CLOSED	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) /BM_SA := BMCsetLoc(SA, MS_list.RT_ID) && BM_SA == NIL => MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	CLOSED
8	CLOSED	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => MS_status := SNV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	CLOSED
9	CLOSED	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => ignore	CLOSED
10	MIDLE	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) => BMCsetRmo(SA, CS_list.RT_ID)	DRIFT-A
11	MIDLE	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	MIDLE
12	MIDLE	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	MIDLE
13	MIDLE	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	MIDLE
14	MIDLE	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == Local_SA => AMList.SLOOPT.start	MIDLE
15	MIDLE	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != Local_SA => AMList.MIDLET.stop AMList.SA := BM_SA AMList.Dferr := 0 AMList.Pherr := 0 AMList.SLOOPT.start AMList.DFCRRT.start INIT_DFCRT_LOOP() INIT_PHCRT_LOOP()	DRIFT-A
16	MIDLE	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.MIDLET.stop AMList.SA := NIL	CLOSED

#	Current State	Event /Condition =>Action	Next State
17	MIDLE	AMList.MIDLET.expired /AMList.Dferr == 0 && AMList.Pherr == 0 => AMList.TRANST.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_M
18	MIDLE	AMList.MIDLET.expired /Abs(AMList.Dferr) > 0) (Abs(AMList.Pherr) > 0) => AMList.TRANST.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_MT
19	DRIFT-A	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => BMCsetRmo(SA, CS_list.RT_ID) AMList.Dferr := DF_CTL_ERR(CS_list) AMList.Pherr := PH_CTL_ERR_(CS_list)	DRIFT-A
20	DRIFT-A	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	DRIFT-A
21	DRIFT-A	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	DRIFT-A
22	DRIFT-A	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	DRIFT-A
23	DRIFT-A	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	DRIFT-A
24	DRIFT-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.ProtVar) && BM_SA != NIL && BM_SA == AMList.SA => DF_CTL_LOOP_DRIFT_A(AMList.Dferr) PH_CTL_LOOP_DRIFT_A(AMList.Pherr) AMList.SLOOPT.start	DRIFT-A
25	DRIFT-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.ProtVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA != Local_SA => AMList.DFCRRT.restart AMList.SA := BM_SA AMList.Dferr := 0 AMList.Pherr := 0 INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() AMList.SLOOPT.start	DRIFT-A

#	Current State	Event /Condition =>Action	Next State
26	DRIFT-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.ProtVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA == AMList.SA => DF_CTL_LOOP_DRIFT_A(AMList.Dferr) PH_CTL_LOOP_DRIFT_A(AMList.Pherr) AMList.SLOOPT.start	DRIFT-A
27	DRIFT-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.ProtVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA != AMList.SA => AMList.DFCRRT.restart AMList.SA := BM_RMO_SA AMList.Dferr := 0 AMList.Pherr := 0 INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() AMList.SLOOPT.start	DRIFT-A
28	DRIFT-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.ProtVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA == NIL => AMList.DFCRRT.stop AMList.SA := Local_SA AMList.Dferr := 0 AMList.Pherr := 0 INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() BMList.MIDLET.start AMList.SLOOPT.start	MIDLE
29	DRIFT-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.DFCRRT.stop AMList.SA := NIL	CLOSED
30	DRIFT-A	AMList.DFCRRT.expired /Abs(AMList.Dferr) < DRIFTA_W => AMList.PLLWT.start	PLLW-A
31	DRIFT-A	AMList.DFCRRT.expired /Abs(AMList.Dferr) >= DRIFTA_W => AMList.DFCRRT.start	DRIFT-A
32	PLLW-A	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => BMCsetRmo(SA, CS_list.RT_ID) AMList.Dferr := DF_CTL_ERR(CS_list) AMList.Pherr := PH_CTL_ERR(CS_list)	PLLW-A

#	Current State	Event /Condition =>Action	Next State
33	PLLW-A	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	PLLW-A
34	PLLW-A	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	PLLW-A
35	PLLW-A	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	PLLW-A
36	PLLW-A	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	PLLW-A
37	PLLW-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA => DF_CTL_LOOP_PLLW_A(AMList.Dferr) PH_CTL_LOOP_PLLW_A(AMList.Pherr) AMList.SLOOPT.start	PLLW-A
38	PLLW-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA != Local_SA => AMList.PLLWT.stop AMList.SA := BM_SA AMList.Dferr := 0 AMList.Pherr := 0 INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() AMList.DFCRRT.start AMList.SLOOPT.start	DRIFT-A
39	PLLW-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA == AMList.SA => DF_CTL_LOOP_PLLW_A(AMList.Dferr) PH_CTL_LOOP_PLLW_A(AMList.Pherr) AMList.SLOOPT.start	PLLW-A

#	Current State	Event /Condition =>Action	Next State
40	PLLW-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA != AMList.SA => AMList.PLLWT.stop AMList.SA := Local_SA AMList.Dferr := 0 AMList.Pherr := 0 INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() AMList.MIDDLE.start AMList.SLOOPT.start	MIDLE
41	PLLW-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA == NIL => AMList.PLLWT.stop AMList.SA := Local_SA AMList.Dferr := 0 AMList.Pherr := 0 INIT_DFCRT_LOOP() INIT_PHCRT_LOOP() AMList.MIDDLE.start AMList.SLOOPT.start	MIDLE
42	PLLW-A	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.PLLWT.stop AMList.SA := NIL	CLOSED
43	PLLW-A	AMList.PLLWT.expired /ABS(AMList.Dferr) < DRIFTA_W && Abs(AMList.Pherr) < PLLA_W => AMList.SYNCHT.start	OPEN_S
44	PLLW-A	AMList.PLLWT.expired /Abs(AMList.Dferr) < DRIFTA_W Abs(AMList.Pherr) >= PLLA_W => AMList.PLLWT.start	PLLW-A
45	PLLW-A	AMList.PLLWT.expired /Abs(AMList.Dferr) >= DRIFTA_W => AMList.DFCRRT.start	DRIFT-A
46	OPEN_S	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => BMCsetRmo(SA, CS_list.RT_ID) AMList.Dferr := DF_CTL_ERR(CS_list) AMList.Pherr := PH_CTL_ERR_(CS_list)	OPEN_S
47	OPEN_S	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	OPEN_S

#	Current State	Event /Condition =>Action	Next State
48	OPEN_S	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	OPEN_S
49	OPEN_S	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	OPEN_S
50	OPEN_S	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	OPEN_S
51	OPEN_S	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA => DF_CTL_LOOP_OPEN_S(AMList.Dferr) PH_CTL_LOOP_OPEN_S(AMList.Pherr) AMList.SLOOPT.start	OPEN_S
52	OPEN_S	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA != Local_SA => AMList.SYNCHT.stop AMList.SA := BM_SA DF_CTL_LOOP_OPEN_S(AMList.Dferr) PH_CTL_LOOP_OPEN_S(AMList.Pherr) AMList.SLOOPT.start	OPEN_ST
53	OPEN_S	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA => AMList.SYNCHT.stop AMList.SA := Local_SA DF_CTL_LOOP_OPEN_S(AMList.Dferr) PH_CTL_LOOP_OPEN_S(AMList.Pherr) AMList.TRANST.start AMList.SLOOPT.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_MT
54	OPEN_S	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.SYNCHT.stop AMList.SA := NIL	CLOSED
55	OPEN_S	AMList.SYNCHT.expired /Abs(AMList.Dferr) < DRIFTA_W && Abs(AMList.Pherr) < PLLA_W => AMList.SYNCHT.start	OPEN_S
56	OPEN_S	AMList.SYNCHT.expired /Abs(AMList.Dferr) < DRIFTA_W Abs(AMList.Pherr) >= PLLA_W => AMList.PLLWT.start	PLLW-A
57	OPEN_S	AMList.SYNCHT.expired /Abs(AMList.Dferr) >= DRIFTA_W => AMList.DFCRRT.start	DRIFT-A

#	Current State	Event /Condition =>Action	Next State
58	OPEN_ST	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => BMCsetRmo(SA, CS_list.RT_ID) AMList.Dferr := DF_CTL_ERR(CS_list) AMList.Pherr := PH_CTL_ERR(CS_list)	OPEN_ST
59	OPEN_ST	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	OPEN_ST
60	OPEN_ST	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	OPEN_ST
61	OPEN_ST	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	OPEN_ST
62	OPEN_ST	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	OPEN_ST
63	OPEN_ST	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA && Abs(AMList.Dferr) < DRIFTA_W && Abs(AMList.Pherr) < PLLA_W => AMList.SYNCHT.start DF_CTL_LOOP_OPEN_ST(AMList.Dferr) PH_CTL_LOOP_OPEN_ST(AMList.Pherr) AMList.SLOOPT.start	OPEN_S
64	OPEN_ST	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA && Abs(AMList.Dferr) < DRIFTA_W && Abs(AMList.Pherr) >= PLLA_W => DF_CTL_LOOP_OPEN_ST(AMList.Dferr) PH_CTL_LOOP_OPEN_ST(AMList.Pherr) AMList.PLLWT.start AMList.SLOOPT.start	PLLW-T
65	OPEN_ST	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA && Abs(AMList.Dferr) > DRIFTA_W => DF_CTL_LOOP_OPEN_ST(AMList.Dferr) PH_CTL_LOOP_OPEN_ST(AMList.Pherr) AMList.DFCRRT.start AMList.SLOOPT.start	DRIFT-T
66	OPEN_ST	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != Local_SA && BM_SA != AMList.SA => AMList.SA := BM_SA DF_CTL_LOOP_OPEN_ST(AMList.Dferr) PH_CTL_LOOP_OPEN_ST(AMList.Pherr) AMList.SLOOPT.restart	OPEN_ST

#	Current State	Event /Condition =>Action	Next State
67	OPEN_ST	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != AMList.SA && BM_SA == Local_SA => AMList.SA := Local_SA DF_CTL_LOOP_OPEN_ST(AMList.Dferr) PH_CTL_LOOP_OPEN_ST(AMList.Pherr) AMList.SLOOPT.start AMList.TRANST.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_MT
68	OPEN_ST	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.SA := NIL	CLOSED
69	PLLW-T	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => BMCsetRmo(SA, CS_list.RT_ID) AMList.Dferr := DF_CTL_ERR(CS_list) AMList.Pherr := PH_CTL_ERR(CS_list)	PLLW-T
70	PLLW-T	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	PLLW-T
71	PLLW-T	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	PLLW-T
72	PLLW-T	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	PLLW-T
73	PLLW-T	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	PLLW-T
74	PLLW-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA => DF_CTL_LOOP_PLLW_T(AMList.Dferr) PH_CTL_LOOP_PLLW_T(AMList.Pherr) AMList.SLOOPT.start	PLLW-T
75	PLLW-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != Local_SA && BM_SA != AMList.SA => AMList.SA := BM_SA DF_CTL_LOOP_PLLW_T(AMList.Dferr) PH_CTL_LOOP_PLLW_T(AMList.Pherr) AMList.PLLWT.restart AMList.SLOOPT.start	PLLW-T

#	Current State	Event /Condition =>Action	Next State
76	PLLW-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA == AMList.SA => DF_CTL_LOOP_PLLW_T(AMList.Dferr) PH_CTL_LOOP_PLLW_T(AMList.Pherr) AMList.SLOOPT.start	PLLW-T
77	PLLW-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA != AMList.SA => AMList.SA := BM_SA DF_CTL_LOOP_PLLW_T(AMList.Dferr) PH_CTL_LOOP_PLLW_T(AMList.Pherr) AMList.PLLWT.restart AMList.SLOOPT.start	PLLW-T
78	PLLW-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA == NIL => AMList.PLLWT.stop AMList.SA := Local_SA DF_CTL_LOOP_PLLW_T(AMList.Dferr) PH_CTL_LOOP_PLLW_T(AMList.Pherr) AMList.TRANST.start AMList.SLOOPT.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_MT
79	PLLW-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.PLLWT.stop AMList.SA := NIL	CLOSED
80	PLLW-T	AMList.PLLWT.expired /Abs(AMList.Dferr) < DRIFTA_W && Abs(AMList.Pherr) < PLLA_W => AMList.SYNCHT.start	OPEN_S
81	PLLW-T	AMList.PLLWT.expired /Abs(AMList.Dferr) < DRIFTA_W Abs(AMList.Pherr) >= PLLA_W => AMList.PLLWT.start	PLLW-T
82	PLLW-T	AMList.PLLWT.expired /Abs(AMList.Dferr) >= DRIFTA_W => AMList.DFCRRT.start	DRIFT-T
83	DRIFT-T	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => BMCsetRmo(SA, CS_list.RT_ID) AMList.Dferr := DF_CTL_ERR(CS_list) AMList.Pherr := PH_CTL_ERR(CS_list)	DRIFT-T

#	Current State	Event /Condition =>Action	Next State
84	DRIFT-T	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	DRIFT-T
85	DRIFT-T	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	DRIFT-T
86	DRIFT-T	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	DRIFT-T
87	DRIFT-T	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	DRIFT-T
88	DRIFT-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == AMList.SA => DF_CTL_LOOP_DRIFT_T(AMList.Dferr) PH_CTL_LOOP_DRIFT_T(AMList.Pherr) AMList.SLOOPT.start	DRIFT-T
89	DRIFT-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != Local_SA && AMList.SA != BM_SA => AMList.SA := BM_SA DF_CTL_LOOP_DRIFT_T(AMList.Dferr) PH_CTL_LOOP_DRIFT_T(AMList.Pherr) AMList.DFCRRT.restart AMList.SLOOPT.start	DRIFT-T
90	DRIFT-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA == AMList.SA => DF_CTL_LOOP_DRIFT_T(AMList.Dferr) PH_CTL_LOOP_DRIFT_T(AMList.Pherr) AMList.SLOOPT.start	DRIFT-T
91	DRIFT-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL && BM_RMO_SA != AMList.SA => AMList.SA := BM_RMO_SA DF_CTL_LOOP_DRIFT_T(AMList.Dferr) PH_CTL_LOOP_DRIFT_T(AMList.Pherr) AMList.DFCRRT.restart AMList.SLOOPT.start	DRIFT-T

#	Current State	Event /Condition =>Action	Next State
92	DRIFT-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && BM_RMO_SA := BMCgetRmo(AMList.ProtVar) && BM_RMO_SA != NIL => AMList.DRCRRT.stop AMList.SA := Local_SA DF_CTL_LOOP_DRIFT_T(AMList.Dferr) PH_CTL_LOOP_DRIFT_T(AMList.Pherr) AMList.TRANST.start AMList.SLOOPT.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_MT
93	DRIFT-T	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.DFCRRT.stop AMList.SA := NIL	CLOSED
94	DRIFT-T	AMList.DFCRRT.expired /Abs(AMList.Dferr) < DRIFTA_W => AMList.PLLWT.start	PLLW-T
95	DRIFT-T	AMList.DFCRRT.expired /Abs(AMList.Dferr) >= DRIFTA_W => AMList.DFCRRT.restart	DRIFT-T
96	OPEN_MT	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => ignore	OPEN_MT
97	OPEN_MT	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	OPEN_MT
98	OPEN_MT	SYN_Sync.cnf (Port, DA, SA, CS_list, CS_status) => ignore	OPEN_MT
99	OPEN_MT	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	OPEN_MT
100	OPEN_MT	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	OPEN_MT
101	OPEN_MT	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	OPEN_MT
102	OPEN_MT	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && ((Abs(AMList.Dferr) > 0) (Abs(AMList.Pherr) > 0)) => DF_CTL_LOOP_OPEN_MT(AMList.Dferr) PH_CTL_LOOP_OPEN_MT(AMList.Pherr) AMList.SA := Local_SA AMList.SLOOPT.start	OPEN_MT

#	Current State	Event /Condition =>Action	Next State
103	OPEN_MT	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA && Abs(AMList.Dferr) == 0 && Abs(AMList.Pherr) == 0 => AMList.SLOOPT.start	OPEN_M
104	OPEN_MT	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != Local_SA => AMList.TRANST.stop AMList.SA := BM_SA AMList.SLOOPT.start	OPEN_ST
105	OPEN_MT	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.TRANST.stop AMList.SLOOPT.stop AMList.SA := NIL	CLOSED
106	OPEN_MT	AMList.TRANST.expired => AMList.TRANST.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_MT
107	OPEN_M	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA == AMList.SA => ignore	OPEN_M
108	OPEN_M	RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) /SA != AMList.SA => BMCsetRmo(SA, CS_list.RT_ID)	OPEN_M
109	OPEN_M	SYN_Sync.cnf (Port, DA, SA, CS_list, CS_status) => ignore	OPEN_M
110	OPEN_M	TMM_MasterAdd.req (SA, ProtVar, MS_list, MS_status) => BMCsetLoc(SA, MS_list.RT_ID) MS_status := SV TMM_MasterAdd.cnf(SA, PortVar, MS_list, MS_status)	OPEN_M
111	OPEN_M	TMM_MasterRem.req (SA, ProtVar, MS_list, MS_status) => BMCremLoc(SA, MS_list.RT_ID) MS_Status := SV TMM_MasterRem.cnf(SA, PortVar, MS_list, MS_status)	OPEN_M
112	OPEN_M	RCTL_TMListEntryRm.ind (SA, rt_id, Status) => BMCremRmo(SA, rt_id)	OPEN_M
113	OPEN_M	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA == Local_SA => AMList.SLOOPT.start	OPEN_M
114	OPEN_M	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA != NIL && BM_SA != Local_SA => AMList.TRANST.stop AMList.SA := BM_SA AMList.SLOOPT.start	OPEN_ST

#	Current State	Event /Condition =>Action	Next State
115	OPEN_M	AMList.SLOOPT.expired /BM_SA := BMCget(AMList.PortVar) && BM_SA == NIL => AMList.TRANST.stop AMList.SLOOPT.stop AMList.SA := NIL	CLOSED
116	OPEN_M	AMList.TRANST.expired => AMList.TRANST.start SYN_Sync.req (Port, DA, SA, CS_list, CS_status)	OPEN_M
117	t-state	Send_Clock pulse of TMM => phase := (phase mod xx) +1, i := 1 Start LMPM_TI_Timer(SchedDB.Impm.SendClockOf) if (SchedDB.Impm(Phase) != NIL) Start LMPM_IRT_Timer(SchedDB.Impm(Phase)(i).TimeOf) For all Ports do j(Port) := 1 if (SchedDB.fw(Port)(Phase) != NIL) Start IFW_IRT_Timer(Port)(SchedDB.fw(Port)(Phase).TimeOf) if (SchedDB.mmac(Port) != NIL) Start MMAC_Interval_Timer(Port)(SchedDB.mmac(Port)(j(Port)).interval)	same state
118	t-state	LMPM_TI_Timer expired => LMPM_TI_Send_Clock.req	same state
119	t-state	LMPM_IRT_Timer expired /SchedDB.Impm(Phase)(i+1) != NIL => CREP := SchedDB.Impm(Phase)(i).CREP i++ Start LMPM_IRT_Timer(SchedDB.Impm(Phase)(i).TimeOf LMPM_IRT_Schedule.req(CREP)	same state
120	t-state	LMPM_IRT_Timer expired /SchedDB.Impm(Phase)(i+1) == NIL => CREP := SchedDB.Impm(Phase)(i).CREP i++ Start LMPM_IRT_Timer(SchedDB.Impm(Phase)(i).TimeOf	same state
121	t-state	IFW_IRT_Timer(Port) expired => IFW_IRT_Schedule.req(Port)	same state
122	t-state	MMAC_Interval_Timer(Port) expired /j(Port) > 3 => ignore	same state
123	t-state	MMAC_Interval_Timer(Port) expired /j(Port) =< 3 => Interv := SchedDB.mmac(Port)(j(Port)).Interv j(Port) ++ IntervLen := SchedDB.mmac(Port)(j(Port)).interval Start MMAC_Interval_Timer(Port)(SchedDB.mmac(Port)(j(Port)).interval) MMAC_IRT_Set_Interval.req (Port, Interv, IntervLen)	same state
124	any state	TMM_Set_Schedule_Item.req(Port, Sched_list) => Store Sched_list in SchedDB	same state

5.8.2.12 RCTL

5.8.2.12.1 Primitive definitions

5.8.2.12.1.1 Primitives exchanged between TMM and RCTL

Table 368 shows all primitives issued by TMM to the RCTL.

Table 368 — Primitives issued by TMM to RCTL

Primitive Name	Source	Associated Parameters	Functions
RCTL_Start.req	TMM	-	
RCTL_Stop.req	TMM	-	
RCTL_TMListEntryRm.req	TMM	SA rt_id Status	

Table 369 shows all primitives issued by the RCTL to the TMM

Table 369 — Primitives issued by RCTL to the TMM

Primitive Name	Source	Associated Parameters	Functions
RCTL_Sync.ind	RCTL	Port DA SA CS_list CS_status	
RCTL_TMListEntryRm.ind	RCTL	SA rt_id Status	

5.8.2.12.1.2 Primitives exchanged between SYN and RCTL

Table 370 shows all primitives issued by SYN to the RCTL.

Table 370 — Primitives issued by SYN to RCTL

Primitive Name	Source	Associated Parameters	Functions
SYN_Delay_Req.cnf	SYN	Port DA SA DQ_list DQ_status	
SYN_Delay_Req.ind	SYN	Port DA SA DQ_list DQ_status	
SYN_Delay_Rsp.cnf	SYN	Port DA SA DS_list DS_status	
SYN_Delay_Rsp.ind	SYN	Port DA SA DS_list DS_status	
SYN_Fllw.ind	SYN	Port DA SA FU_list FU_status	
SYN_Sync.ind	SYN	Port DA SA CS_list CS_status	

Table 371 shows all primitives issued by the RCTL to the SYN .

Table 371 — Primitives issued by RCTL to SYN

Primitive Name	Source	Associated Parameters	Functions
SYN_Delay_Req.req	RCTL	Port DA SA DQ_list DQ_status	
SYN_Delay_Rsp.req	RCTL	Port DA SA DS_list DS_status	

5.8.2.12.1.3 Parameters of RCTL primitives

Table 372 shows all parameters used with primitives between the TMM, SYN, FW and the RCTL.

Table 372 — Parameters used with primitives exchanged between TMM, SYN, FW and RCTL

Parameter Name	Description
DA	Destination Address
SA	Source Address
Port	
CS_list	Cycle Sync List
CS_status	Cycle Sync Status
DS_list	Delay Response List
DS_status	Delay Response Status
TMList	TimeMaster List
Status	
rt_id	FrameID
DQ_list	Delay Request List
DQ_status	Delay Request Status
FU_list	Follow Up List
FU_status	Follow Up Status

5.8.2.12.2 RCTL state table

The RCTL state table is shown in Table 373.

Table 373 — RCTL state table

#	Current State	Event /Condition =>Action	Next State
1	POWER-ON	<pre> => for m := TMList.MaxCnEntry do { for i := all Ports do { TMList[m].Delay[i].Value := 0 TMList[m].DelayReq[i] := FALSE TMList[m].Rctl[Port].CS_list := NIL } } </pre>	CLOSED
2	CLOSED	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) => ignore	CLOSED
3	CLOSED	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) => ignore	CLOSED

4	CLOSED	SYN_Delay_Req.ind (Port, DA, SA, DQ_list, DQ_status) => ignore	CLOSED
5	CLOSED	SYN_Delay_Rsp.cnf (Port, DA, SA, DS_list, DS_status) => ignore	CLOSED
6	CLOSED	SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status) => ignore	CLOSED
7	CLOSED	RCTL_Start.req () => start RCTL-SM SYN_FW_Start.req ()	OPEN
8	CLOSED	RCTL_Stop.req () => ignore	CLOSED
9	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /!({SA, ProtVar} in TMList) => ignore	OPEN
10	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /({SA, ProtVar} in TMList && CS_status == SV && ICS_list.S_SDU.Fwup && ICS_list.S_SDU.DelayReq => TMList{SA, ProtVar}.TMRCTL.start RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status)	OPEN
11	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /({SA, ProtVar} in TMList && CS_status == SV && CS_list.S_SDU.Fwup && ICS_list.S_SDU.DelayReq => TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Rctl[Port].CS_list := CS_list TMList{SA, ProtVar}.Rctl[Port].CS_status := CS_status	OPEN
12	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /({SA, ProtVar} in TMList && CS_status == SV && CS_list.S_SDU.DelayReq && ICS_list.S_SDU.Fwup && TMList{SA, ProtVar}.DelayReq[Port] => TMList{SA, ProtVar}.TMRCTL.start RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status)	OPEN

13	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{SA, ProtVar} in TMList && CS_status == SV && CS_list.S_SDU.DelayReq && !CS_list.S_SDU.Fwup && !TMList{SA, ProtVar}.Delay[Port].Req ==> TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Delay[Port].Req := TRUE TMList{SA, ProtVar}.Delay[Port].ReqCnf := FALSE TMList{SA, ProtVar}.Delay[Port].Fwup := TRUE TMList{SA, ProtVar}.Delay[Port].Rsp := FALSE TMList{SA, ProtVar}.Delay[Port].SyncT1 := CS_list.S_SDU.T1 TMList{SA, ProtVar}.Delay[Port].T2 := CS_list.S_SDU.T2 TMList{SA, ProtVar}.Delay[Port].Seq := CS_list.S_SDU.Seq TMList{SA, ProtVar}.Delay[Port].DYCTL.start DQ_list.S_SDU.Seq := CS_list.S_SDU.Seq DQ_status := SV DYRQ := TRUE RCTL_Sync.ind (Port, DA, SA, CS_list, CS_status) SYN_Delay_Req.req (Port, DA, SA, DQ_list, DQ_status)	OPEN
14	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{SA, ProtVar} in TMList && CS_status == SV && CS_list.S_SDU.DelayReq && CS_list.S_SDU.Fwup && !TMList{SA, ProtVar}.Delay[Port].Req ==> TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Rctl[Port].CS_list := CS_list TMList{SA, ProtVar}.Rctl[Port].CS_status := CS_status TMList{SA, ProtVar}.Delay[Port].Req := TRUE TMList{SA, ProtVar}.Delay[Port].ReqCnf := FALSE TMList{SA, ProtVar}.Delay[Port].Fwup := FALSE TMList{SA, ProtVar}.Delay[Port].Rsp := FALSE TMList{SA, ProtVar}.Delay[Port].SyncT1 := CS_list.S_SDU.T1 TMList{SA, ProtVar}.Delay[Port].T2 := CS_list.S_SDU.T2 TMList{SA, ProtVar}.Delay[Port].Seq := CS_list.S_SDU.Seq	OPEN
15	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{SA, ProtVar} in TMList && CS_status == SV && CS_list.S_SDU.DelayReq && CS_list.S_SDU.Fwup && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup ==> TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Rctl[Port].CS_list := CS_list TMList{SA, ProtVar}.Rctl[Port].CS_status := CS_status TMList{SA, ProtVar}.Delay[Port].Req := TRUE TMList{SA, ProtVar}.Delay[Port].SyncT1 := CS_list.S_SDU.T1 TMList{SA, ProtVar}.Delay[Port].T2 := CS_list.S_SDU.T2 TMList{SA, ProtVar}.Delay[Port].Seq := CS_list.S_SDU.Seq	OPEN
16	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{SA, ProtVar} in TMList && CS_status == SV && CS_list.S_SDU.Fwup && CS_list.S_SDU.DelayReq && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Fwup ==> TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Rctl[Port].CS_list := CS_list TMList{SA, ProtVar}.Rctl[Port].CS_status := CS_status	OPEN

17	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{(SA, ProtVar) in TMList && CS_status == NDY && !CS_list.S_SDU.Fwup && !TMList{SA, ProtVar}.Delay[Port].Req => TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Delay[Port].Req := TRUE TMList{SA, ProtVar}.Delay[Port].ReqCnf := FALSE TMList{SA, ProtVar}.Delay[Port].Fwup := TRUE TMList{SA, ProtVar}.Delay[Port].Rsp := FALSE TMList{SA, ProtVar}.Delay[Port].SyncT1 := CS_list.S_SDU.T1 TMList{SA, ProtVar}.Delay[Port].T2 := CS_list.S_SDU.T2 TMList{SA, ProtVar}.Delay[Port].Seq := CS_list.S_SDU.Seq TMList{SA, ProtVar}.Delay[Port].DYCTL.start DQ_list.S_SDU.Seq := CS_list.S_SDU.Seq DQ_status := SV DYRQ := TRUE SYN_Delay_Req.req (Port, DA, SA, DQ_list, DQ_status)	OPEN
18	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{(SA, ProtVar) in TMList && CS_status == NDY && !CS_list.S_SDU.Fwup && TMList{SA, ProtVar}.Delay[Port].Req => TMList{SA, ProtVar}.TMRCTL.start	OPEN
19	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{(SA, ProtVar) in TMList && CS_status == NDY && CS_list.S_SDU.Fwup && !TMList{SA, ProtVar}.Delay[Port].Req => TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Delay[Port].Req := TRUE TMList{SA, ProtVar}.Delay[Port].ReqCnf := FALSE TMList{SA, ProtVar}.Delay[Port].Fwup := FALSE TMList{SA, ProtVar}.Delay[Port].Rsp := FALSE TMList{SA, ProtVar}.Delay[Port].SyncT1 := CS_list.S_SDU.T1 TMList{SA, ProtVar}.Delay[Port].T2 := CS_list.S_SDU.T2 TMList{SA, ProtVar}.Delay[Port].Seq := CS_list.S_SDU.Seq	OPEN
20	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{(SA, ProtVar) in TMList && CS_status == NDY && CS_list.S_SDU.Fwup && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup => TMList{SA, ProtVar}.TMRCTL.start TMList{SA, ProtVar}.Delay[Port].Req := TRUE TMList{SA, ProtVar}.Delay[Port].SyncT1 := CS_list.S_SDU.T1 TMList{SA, ProtVar}.Delay[Port].T2 := CS_list.S_SDU.T2 TMList{SA, ProtVar}.Delay[Port].Seq := CS_list.S_SDU.Seq	OPEN
21	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{(SA, ProtVar) in TMList && CS_status == NDY && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Fwup => TMList{SA, ProtVar}.TMRCTL.start	OPEN
22	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{(SA, ProtVar) in TMList && CS_status == NTM => TMList{SA, ProtVar}.TMRCTL.start	OPEN

23	OPEN	SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) /{SA, ProtVar} in TMList && CS_status != NDY CS_status != SV CS_status != NTM => ignore	OPEN
24	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList => ignore	OPEN
25	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list != NIL && !TMList{SA, ProtVar}.Delay[Port].Req => TMList{SA, ProtVar}.Rctl[Port].CS_list.S_SDU.Delay += FU_list.S_SDU.Delay TMList{SA, ProtVar}.Rctl[Port].CS_list := NIL RCTL_Sync.ind (Port, DA, SA, TMList{SA, ProtVar}.Rctl[Port].CS_list, TMList{SA, ProtVar}.Rctl[Port].CS_status)	OPEN / W_DYRS
26	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list == NIL && !TMList{SA, ProtVar}.Delay[Port].Req => ignore	OPEN / W_DYRS
27	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list != NIL && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Fwup => TMList{SA, ProtVar}.Rctl[Port].CS_list.S_SDU.Delay += FU_list.S_SDU.Delay TMList{SA, ProtVar}.Rctl[Port].CS_list := NIL RCTL_Sync.ind (Port, DA, SA, TMList{SA, ProtVar}.Rctl[Port].CS_list, TMList{SA, ProtVar}.Rctl[Port].CS_status)	OPEN / W_DYRS
28	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list == NIL && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Fwup => ignore	OPEN / W_DYRS
29	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list != NIL && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup && TMList{SA, ProtVar}.Delay[Port].Seq != FU_list.S_SDU.Seq => TMList{SA, ProtVar}.Rctl[Port].CS_list.S_SDU.Delay += FU_list.S_SDU.Delay TMList{SA, ProtVar}.Rctl[Port].CS_list := NIL RCTL_Sync.ind (Port, DA, SA, TMList{SA, ProtVar}.Rctl[Port].CS_list, TMList{SA, ProtVar}.Rctl[Port].CS_status)	OPEN / W_DYRS

30	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list == NIL && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup && TMList{SA, ProtVar}.Delay[Port].Seq != FU_list.S_SDU.Seq => ignore	OPEN / W_DYRS
31	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list != NIL && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup && TMList{SA, ProtVar}.Delay[Port].Seq := FU_list.S_SDU.Seq => TMList{SA, ProtVar}.Rctl[Port].CS_list.S_SDU.Delay += FU_list.S_SDU.Delay TMList{SA, ProtVar}.Rctl[Port].CS_list := NIL TMList{SA, ProtVar}.Delay[Port].Fwup := TRUE TMList{SA, ProtVar}.Delay[Port].FwupT1 := FU_list.S_SDU.T1 DQ_list.S_SDU.Seq := FS_list.S_SDU.Seq DQ_status := SV DYRQ := TRUE RCTL_Sync.ind (Port, DA, SA, TMList{SA, ProtVar}.Rctl[Port].CS_list, TMList{SA, ProtVar}.Rctl[Port].CS_status) SYN_Delay_Req.req (Port, DA, SA, DQ_list, DQ_status)	OPEN / W_DYRS
32	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == SV && TMList{SA, ProtVar}.Rctl[Port].CS_list == NIL && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup && TMList{SA, ProtVar}.Delay[Port].Seq := FU_list.S_SDU.Seq => TMList{SA, ProtVar}.Delay[Port].Fwup := TRUE TMList{SA, ProtVar}.Delay[Port].FwupT1 := FU_list.S_SDU.T1 DQ_list.S_SDU.Seq := FS_list.S_SDU.Seq DQ_status := SV DYRQ := TRUE SYN_Delay_Req.req (Port, DA, SA, DQ_list, DQ_status)	OPEN / W_DYRS
33	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == NDY && !TMList{SA, ProtVar}.Delay[Port].Req => ignore	OPEN / W_DYRS
34	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == NDY && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Fwup => ignore	OPEN / W_DYRS
35	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == NDY && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup && TMList{SA, ProtVar}.Delay[Port].Seq != FU_list.S_SDU.Seq => ignore	OPEN / W_DYRS

36	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && FU_status == NDY && TMList{SA, ProtVar}.Delay[Port].Req && !TMList{SA, ProtVar}.Delay[Port].Fwup && TMList{SA, ProtVar}.Delay[Port].Seq := FU_list.S_SDU.Seq => TMList{SA, ProtVar}.Delay[Port].Fwup := TRUE TMList{SA, ProtVar}.Delay[Port].FwupT1 := FU_list.S_SDU.T1 DQ_list.S_SDU.Seq := FS_list.S_SDU.Seq DQ_status := SV DYRQ := TRUE SYN_Delay_Req.req (Port, DA, SA, DQ_list, DQ_status)	OPEN / W_DYRS
37	OPEN	SYN_Filw.ind (Port, DA, SA, FU_list, FU_status) /{SA, ProtVar} in TMList && (CS_status != NDY CS_status != SV) => ignore	OPEN
38	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ => ignore	OPEN
39	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ && !({SA, ProtVar} in TMList) => DYRQ := FALSE	OPEN
40	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ && {SA, ProtVar} in TMList && DQ_status != SV => ignore	OPEN
41	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ && {SA, ProtVar} in TMList && DQ_status == SV && !TMList{SA, ProtVar}.Delay[Port].Req => ignore	OPEN
42	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ && {SA, ProtVar} in TMList && DQ_status == SV && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Seq != DQ_list.S_SDU.Seq => ignore	W_DYRS
43	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ && {SA, ProtVar} in TMList && DQ_status == SV && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Seq == DQ_list.S_SDU.Seq && !TMList{SA, ProtVar}.Delay[Port].Rsp => TMList{SA, ProtVar}.Delay[Port].ReqCnf := TRUE TMList{SA, ProtVar}.Delay[Port].T3 := DQ_list.S_SDU.T3 DYRQ := FALSE	OPEN

44	W_DYRS	SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status) /DYRQ && {SA, ProtVar} in TMList && DQ_status == SV && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Seq == DQ_list.S_SDU.Seq && TMList{SA, ProtVar}.Delay[Port].Rsp => TMList{SA, ProtVar}.Delay[Port].T3 := DQ_list.S_SDU.T3 if (TMList{SA, ProtVar}.Delay[Port].T1Sync) TMList{SA, ProtVar}.Delay[Port].Value := ((TMList{SA, ProtVar}.Delay[Port].T4 – TMList{SA, ProtVar}.Delay[Port].SyncT) - (TMList{SA, ProtVar}.Delay[Port].T3 -TMList{SA, ProtVar}.Delay[Port].T2) / 2) else TMList{SA, ProtVar}.Delay[Port].Value := ((TMList{SA, ProtVar}.Delay[Port].T4 – TMList{SA, ProtVar}.Delay[Port].FwupT) - (TMList{SA, ProtVar}.Delay[Port].T3 -TMList{SA, ProtVar}.Delay[Port].T2) / 2) TMList{SA, ProtVar}.Delay[Port].TMCTL.stop TMList{SA, ProtVar}.Delay[Port].Req := FALSE DYRQ := FALSE	OPEN
45	W_DYRS	RCTL_Start.req () => ignore	W_DYRS
46	W_DYRS	RCTL_Stop.req () => stop RCTL-SM FW_Stop.req ()	CLOSED
47	OPEN	SYN_Delay_Req.ind (Port, DA, SA, DQ_list, DQ_status) /DQ_status != SV => ignore	OPEN
48	OPEN	SYN_Delay_Req.ind (Port, DA, SA, DQ_list, DQ_status) /DQ_status == SV => if (!FU) DS_list.S_SDU.T1Sync := TRUE else DS_list.S_SDU.T1Sync := FALSE DS_list.S_SDU.Seq := DQ_list.S_SDU.Seq DS_status := SV SYN_Delay_Rsp.req (Port, DA, SA, DS_list, DS_status)	OPEN
49	W_DYRS	SYN_Delay_Rsp.cnf (Port, DA, SA, DS_list, DS_status) / => ignore	OPEN
50	OPEN	SYN_Delay_Req.ind (Port, DA, SA, DQ_list, DQ_status) /DS_status != SV => ignore	OPEN
51	OPEN	SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status) /DS_status == SV && !({SA, ProtVar} in TMList) => ignore	OPEN
52	OPEN	SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status) /DS_status == SV && {SA, ProtVar} in TMList && !TMList{SA, ProtVar}.Delay[Port].Req => ignore	OPEN

53	OPEN	SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status) /DS_status == SV && {SA, ProtVar} in TMList && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Seq != DQ_list.S_SDU.Seq => ignore	OPEN
54	OPEN	SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status) /DS_status == SV && {SA, ProtVar} in TMList && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Seq == DS_list.S_SDU.Seq && !TMList{SA, ProtVar}.Delay[Port].ReqCnf => TMList{SA, ProtVar}.Delay[Port].Rsp := TRUE TMList{SA, ProtVar}.Delay[Port].T1 := DS_list.S_SDU.DelayT1 TMList{SA, ProtVar}.Delay[Port].T4 := DS_list.S_SDU.T4	OPEN
55	OPEN	SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status) /DS_status == SV && {SA, ProtVar} in TMList && TMList{SA, ProtVar}.Delay[Port].Req && TMList{SA, ProtVar}.Delay[Port].Seq == DQ_list.S_SDU.Seq && TMList{SA, ProtVar}.Delay[Port].ReqCnf => TMList{SA, ProtVar}.Delay[Port].T3 := DS_list.S_SDU.T4 if (DS_list.S_SDU.DelayT1Sync) TMList{SA, ProtVar}.Delay[Port].Value := ((TMList{SA, ProtVar}.DelayT4[Port] - TMList{SA, ProtVar}.DelaySyncT1[Port]) - (TMList{SA, ProtVar}.DelayT3[Port] -TMList{SA, ProtVar}.DelayT2[Port]) / 2) else TMList{SA, ProtVar}.Delay[Port].Value := ((TMList{SA, ProtVar}.Delay[Port].T4 - TMList{SA, ProtVar}.Delay[Port].FwupT1) - (TMList{SA, ProtVar}.Delay[Port].T3 -TMList{SA, ProtVar}.Delay[Port].T2) / 2) TMList{SA, ProtVar}.Delay[Port].TMCTL.stop TMList{SA, ProtVar}.Delay[Port].Req := FALSE	OPEN
56	OPEN / WDYRS	/TMList{SA, ProtVar}.Delay[Port].DYCTL.expired => TMList{SA, ProtVar}.Delay[Port].Req := FALSE	OPEN
57	OPEN / WDYRS	/TMList{SA, ProtVar}.TMRCTL.expired => remove Entry from TMList{SA, ProtVar} TMListActCntEntry := TMListActCntEntry – 1 RCTL_TMListEntryRm.ind (SA, rt_id, Status)	OPEN
58	OPEN / WDYRS	RCTL_TMListEntryRm.req (SA, rt_id, Status) /{SA, ProtVar} in TMList => remove Entry from TMList{SA, ProtVar} TMListActCntEntry := TMListActCntEntry – 1 Status := SV RCTL_TMListEntryRm.cnf (SA, rt_id, Status)	OPEN
59	OPEN / WDYRS	RCTL_TMListEntryRm.req (SA, rt_id, Status) /{SA, ProtVar} in TMList => Status := NoEntry RCTL_TMListEntryRm.cnf (SA, rt_id, Status)	OPEN
60	OPEN	RCTL_Start.req () => ignore	OPEN

61	OPEN	RCTL_Stop.req () => stop RCTL-SM FW_Stop.req ()	CLOSED

5.8.3 DMPM common data resource

The Data Resource models a data interface for all state machines used to describe the DMPM. It contains queues and buffers for exchanging data between the sublayers (LMPM, MMAC) of the DMPM. Moreover it contains management information which have to be accessed by various sublayers. Table 374 illustrates all Data Resources.

Table 374 — Data resource

Name	Struct Element	Type	Range	Remark
Variables		S0		General Variables
Cy_List		S1		Cyclic Request List
Ac_List		S2		Acyclic Request List
N_List		S3		NonRTRequest List
C_List		S4		Confirmation List
I_List		S5		Indication List
FC_List		S6		Free Cyclic List to be used for Indication Buffers
FA_List		S6		Free Acyclic List to be used for Indication Buffers
FN_List		S6		Free NonRT List to be used for Indication Buffers
S0	Variables			
	TS	octed string	6 Octets	DL-address of this station
	Medium_redundancy		single; redundant	Availability of redundant media
	RT	Unsigned16		Tag for RT-Frames, LT
	ReductionRatioMax	Unsigned16	1..16384	Maximum of allowed ReductionRatio
	HW-Release		LE_HR; hardware release identification	Hardware release number
	SW-Release		LE_SR; software release identification	Software release number
S1	Cy_List			
	Num_entry	U16		Number of entries in the list
	First_entry	Ref to S7		Pointer to a list element
	Last_entry	Ref to S7		Pointer to a list element
	Insert()			Method to insert a new element after Last_entry
	Remove()			Method to remove element to which First-entry points to
S2	Ac_List			
	Num_entry	U16		Number of entries in the list
	First_entry	Ref to S8		Pointer to a list element
	Last_entry	Ref to S8		Pointer to a list element
	Insert()			Method to insert a new element after Last_entry
	Remove()			Method to remove element to which First-entry points to
S3	N_List			
	Num_entry	U16		Number of entries in the list
	First_entry	Ref to S9		Pointer to a list element
	Last_entry	Ref to S9		Pointer to a list element
	Insert()			Method to insert a new element after Last_entry
	Remove()			Method to remove element to which First-entry points to
S4	C_List			
	Num_entry	U16		Number of entries in the list
	First_entry	Ref to S10		Pointer to a list element

Name	Struct Element	Type	Range	Remark
	Last_entry	Ref to S10		Pointer to a list element
	Insert()			Method to insert a new element after Last_entry
	Remove()			Method to remove element to which First-entry points to
S5	I_List			
	Num_entry	U16		Number of entries in the list
	First_entry	Ref to S11		Pointer to a list element
	Last_entry	Ref to S11		Pointer to a list element
	Insert()			Method to insert a new element after Last_entry
	Remove()			Method to remove element to which First-entry points to
S6	F_List			
	Num_entry	U16		Number of entries in the list
	First_entry	Ref to S11		Pointer to a list element
	Last_entry	Ref to S11		Pointer to a list element
	Insert()			Method to insert a new element
	Remove()			Method to remove element
S7	Cy_List entry			
	Next entry	Ref to S7		Pointer to a list element
	CREP	Unsigned 16		Communication Reference
	DA		6 Octets	Destination Address
	SA		6 Octets	Source Address
	Prio		0..7	User Priority according to IEEE 802.1Q
	VLAN_Tag		0..4095	VID according to IEEE 802.1Q
	C_SDU	S12		Cyclic Service Data Unit
	APDU_Status	Unsigned16, Unsigned8, Unsigned8		Status bits of the sending device
	Port			
S8	Ac_List entry			
	Next entry	Ref to S8		Pointer to a list element
	CREP	Unsigned 16		Communication Reference
	DA		6 Octets	Destination Address
	SA		6 Octets	Source Address
	Prio		0..7	User Priority according to IEEE 802.1Q
	VLAN_Tag		0..4095	VID according to IEEE 802.1Q
	LT	Unsigned 16		
	A_SDU	S12		Acyclic Service Data Unit
S9	N_List entry			
	Next entry	Ref to S9		Pointer to a list element
	CREP			Communication Reference, Format not specified
	DA		6 Octets	Destination Address
	SA		6 Octets	Source Address
	Prio		0..7	User Priority according to IEEE 802.1Q
	VLAN_Tag		0..4095	VID according to IEEE 802.1Q
	N_SDU	S12		Acyclic Service Data Unit
S10	C_List entry			
	Next entry	Ref to S10		Pointer to a list element
	CREP			Communication Reference, Format not specified
	Function		C, A, N	Gives back to the function Cyclic, Acyclic, NonRealTime
S11	I_List entry			
	Next entry	Ref to S11		Pointer to a list element

Name	Struct Element	Type	Range	Remark
	CREP			Communication Reference, Format not specified
	Function		C, A, N	Gives back to the function Cyclic, Acyclic, NonRealTime
	DA		6 Octets	Destination Address
	SA		6 Octets	Source Address
	Prio		0..7	User Priority according to IEEE 802.1Q
	VLAN_Tag		0..4095	VID according to IEEE 802.1Q
	LT	Unsigned 16		Cyclic Service Data Unit
	DLSDU	S12		Cyclic Service Data Unit
	APDU_Status	Unsigned16, OctetString		Status Infos induced by switches and sending devices
	Port			
S12	DLSDU			
	Len	U16	0..1500	Restrictions for A_SDU, C_SDU
	Data	OctetString		

5.8.4 LMPM

5.8.4.1 Primitive definitions

5.8.4.1.1 Primitives exchanged between LMPM-User and LMPM

Table 375 shows all primitives issued by LMPM-User to the LMPM.

Table 375 — Primitives issued by LMPM-User to LMPM

Primitive Name	Associated Parameters
LMPM_C_Data.req	CREP Port DA SA Prio VLAN_Tag C_SDU APDU_Status
LMPM_A_Data.req	CREP DA SA Prio A_SDU
LMPM_N_Data.req	CREP DA SA Prio LT N_SDU
LMPM_Schedule_add.req	CREP ReductionRatio Phase Sequence
LMPM_Schedule_remove.req	CREP
LMPM_IRT_Schedule.req	CREP
LMPM_TI_SendClock.req	-

Table 376 shows all primitives issued by the LMPM to the LMPM-User.

Table 376 — Primitives issued by LMPM to the LMPM-User

Primitive Name	Associated Parameters
LMPM_C_Data.cnf	CREP LMPM_status
LMPM_A_Data.cnf	CREP LMPM_status

Primitive Name	Associated Parameters
LMPM_N_Data.cnf	CREP LMPM_status
LMPM_Schedule_add.cnf(+/-)	CREP
LMPM_C_Data.ind	CREP Port DA SA Prio C_SDU APDU_Status
LMPM_A_Data.ind	CREP DA SA Prio A_SDU
LMPM_N_Data.ind	DA SA Prio VLAN_Tag LT N_SDU
LMPM_Time_Event.ind	CREP Cycle

5.8.4.1.2 Parameters of LMPM primitives

Table 377 shows all parameters used with primitives between the LMPM-User and the LMPM.

Table 377 — Parameters used with primitives exchanged between LMPM-User and LMPM

Parameter Name	Description
A_SDU	Acyclic realtime service data unit
APDU_Status	Status of a C_PDU consist of Cycle Number, Data Status, Transfer Status
C_SDU	Cyclic realtime service data unit
CREP	Communication Reference unique for that Station (can be deduced from Received Data with RT-Ethertype)
DA	Destination Address
LMPM_status	IV = invalid Parameter, LS = no local Resource available, OK = no error
LT	Type field of a 8802-3 PDU
N_SDU	Non realtime service data unit
Phase	Offset of the start of a send interval between 1 and ReductionRatio
Prio	Priority according to IEEE 802.1Q
ReductionRatio	This parameter multiplied with the SendClockTime is the send interval.
SA	Source Address
Sequence	Sequence is relative number within a phase (= 0 means no numbering)
VLAN_Tag	VID according to IEEE 802.1Q

5.8.4.2 State machine description

The LMPM forms the interface between MMAC and LMPM-User for various data and management services. In addition LMPM is the source for time events for the various protocols.

The services are all handled in the the READY-State.

The service request will be validated first. A negative validation will result in a negative confirmation. Otherwise the service will be put into the appropriate priority service queue. Services executed by MMAC will be moved from the request queues to the confirmation queues. All valid incoming services will be put into the indication queue.

Local Variables

The LMPM does not use local variables. All variables are contained in the Data Resource.

5.8.4.3 LMPM state table

The LMPM State Table is shown in Table 378.

Table 378 — LMPM state table

#	Current State	Event /Condition =>Action	Next State
1	READY	LMPM_N_Data.req (CREP, DA, SA, Prio, VLAN_Tag, LT, N_SDU) /!CHECK_PAR_N => LMPM_status := IV LMPM_N_Data.cnf (CREP, LMPM_status)	READY
2	READY	LMPM_N_Data.req (CREP, DA, SA, Prio, VLAN_Tag, LT, N_SDU) /CHECK_PAR_N && !RESOURCE_N(N_SDU.Len) => LMPM_status := LS LMPM_N_Data.cnf (CREP, LMPM_status)	READY
3	READY	LMPM_N_Data.req (CREP, DA, SA, Prio, VLAN_Tag, LT, N_SDU) /CHECK_PAR_N && RESOURCE_N(N_SDU.Len) && (DA in ADD_DB) => For all Ports in ADD_DB PUT_NREQ (Port)	READY
4	READY	LMPM_N_Data.req (CREP, DA, SA, Prio, VLAN_Tag, LT, N_SDU) /CHECK_PAR_N && RESOURCE_N(N_SDU.Len) && !(DA in ADD_DB) => For all Ports/Port PUT_NREQ (Port)	READY
5	READY	LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU) /!CHECK_PAR_A => LMPM_status := IV LMPM_A_Data.cnf (CREP, LMPM_status)	READY
6	READY	LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU) /CHECK_PAR_A && !RESOURCE_A(A_SDU.Len) => LMPM_status := LS LMPM_A_Data.cnf (CREP, LMPM_status)	READY
7	READY	LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU) /CHECK_PAR_A && RESOURCE_A(A_SDU.Len) && (DA in ADD_DB) => For all Ports in ADD_DB PUT_AREQ (Port)	READY
8	READY	LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU) /CHECK_PAR_A && RESOURCE_A(A_SDU.Len) && !(DA in ADD_DB) => For all Ports/Port PUT_AREQ (Port)	READY
9	READY	LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) /!CHECK_PAR_C => LMPM_status := IV LMPM_C_Data.cnf (CREP, LMPM_status)	READY
10	READY	LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) /CHECK_PAR_C && !RESOURCE_C(C_SDU.Len) => LMPM_status := LS LMPM_C_Data.cnf (CREP, LMPM_status)	READY

#	Current State	Event /Condition =>Action	Next State
11	READY	LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) /CHECK_PAR_C && RESOURCE_C(C_SDU.Len) && Port != NIL => APDU_Status.cycle := Idx*SendClock PUT_CREQ (Port)	READY
12	READY	LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) /CHECK_PAR_C && RESOURCE_C(C_SDU.Len) && Port == NIL && DA in ADD_DB => APDU_Status.cycle := Idx*SendClock For all Ports in ADD_DB PUT_CREQ (Port)	READY
13	READY	LMPM_C_Data.req (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status) /CHECK_PAR_C && RESOURCE_C(C_SDU.Len) && Port == NIL && !(DA in ADD_DB) => APDU_Status.cycle := Idx*SendClock For all Ports/Port PUT_CREQ (Port)	READY
14	READY	/C_List.Num_entry<>0 && C_List.First_entry.Function = N => LMPM_status := OK GET_CNF() LMPM_N_Data.cnf (CREP, LMPM_status)	READY
15	READY	/C_List.Num_entry<>0 && C_List.First_entry.Function = Ac => LMPM_status := OK GET_CNF() LMPM_A_Data.cnf (CREP, LMPM_status)	READY
16	READY	/C_List.Num_entry<>0 && C_List.First_entry.Function = Cy => LMPM_status := OK GET_CNF() LMPM_C_Data.cnf (CREP, LMPM_status)	READY
17	READY	/I_List.Num_entry<>0 && I_List.First_entry.Function = N => GET_N_IND() LMPM_N_Data.ind (DA, SA, Prio, VLAN_Tag, LT, N_SDU)	READY
18	READY	/I_List.Num_entry<>0 && I_List.First_entry.Function = Ac => GET_A_IND() LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU)	READY
19	READY	/I_List.Num_entry<>0 && I_List.First_entry.Function = Cy => GET_C_IND() APDU_Status.cycle := Idx*SendClock LMPM_C_Data.ind (CREP, Port, DA, SA, Prio, C_SDU, APDU_Status)	READY

#	Current State	Event /Condition =>Action	Next State
20	READY	LMPM_TI_SendClock.req => Cycle := Cycle + SendClockOffset Idx := (Idx+1) mod (ReducMax) For each Element in Time_CList(Idx) LMPM_Time_Event.ind(CREP) For each Element in Time_AList(Idx) LMPM_Time_Event.ind(CREP) Time_AList.Remove()	READY
21	READY	LMPM_Schedule_add.req (CREP,ReductionRatio,Phase, Sequence) /Phase == NIL => Idy:= (Idx + ReductionRatio) mod (ReducMax) Time_AList.Insert(Idy, CREP, Sequence) LMPM_Schedule_add.cnf (+) (CREP)	READY
22	READY	LMPM_Schedule_add.req (CREP,ReductionRatio,Phase, Sequence) /Phase != NIL && (Phase < ReductionRatio) && (ReducMax mod ReductionRatio = 0) => For each n := 0 to (ReducMax/ReductionRatio-1) Do Time_CList.Insert(n*ReductionRatio +Phase, CREP, Sequence) LMPM_Schedule_add.cnf (+) (CREP)	READY
23	READY	LMPM_Schedule_add.req (CREP,ReductionRatio,Phase, Sequence) /Phase != NIL && (Phase >= ReductionRatio) (ReducMax mod ReductionRatio > 0) => LMPM_Schedule_add.cnf (-) (CREP)	READY
24	READY	LMPM_Schedule_remove.req (CREP) => For each Time_AList entry of CREP Do Time_AList.Remove() For each Time_CList entry of CREP Do Time_Clist.Remove()	READY
25	READY	LMPM_IRT_Schedule.req(CREP) => LMPM_Time_Event.ind(CREP)	READY

5.8.4.4 Functions

The LMPM Functions are summarized in Table 379.

NOTE rt_id is an alias for FrameID

Table 379 — LMPM function table

Function name	Operations
CHECK_PAR_N	Check that all parameters of Service LMPM_N_Data.req are valid. Number of parameters must be 7. LT != RT SA shall be the one of own station address N_SDU.Len: 0..1500 N_SDU.Data: according to DLSDU.Len
RESOURCE_N	Check that local resources are available to handle the requested data
PUT_NREQ	Put a service request to the high prior service queue to MMAC. N_List.Insert() N_List.First_Entry.CREP := CREP N_List.First_Entry.DA := DA N_List.First_Entry.SA := SA N_List.First_Entry.Prio := Prio N_List.First_Entry.VLAN_Tag := VLAN_Tag N_List.First_Entry.LT := LT N_List.First_Entry.N_SDU := N_SDU

Function name	Operations
	N_List.Num_entry++
CHECK_PAR_A	Check that all parameters of Service LMPM_A_Data.req are valid. Number of parameters must be 4. SA shall be the one of own station address A_SDU.Len: 0..1440 A_SDU.Data: according to DLSDU.Len
RESOURCE_A	Check that local resources are available to handle the requested data
PUT_AREQ	Put a service request to the high prior service queue to MMAC. Ac_List.Insert() Ac_List.First_Entry.CREP := CREP Ac_List.First_Entry.DA := DA Ac_List.First_Entry.SA := SA Ac_List.First_Entry.A_SDU := A_SDU Ac_List.Num_entry++
CHECK_PAR_C	Check that all parameters of Service LMPM_C_Data.req are valid. Number of parameters must be 6. SA shall be the one of own station address C_SDU.Len: 0..1440 C_SDU.Data: according to DLSDU.Len
RESOURCE_C	Check that local resources are available to handle the requested data
PUT_CREQ	Put a service request to the high prior service queue to MMAC. Cy_List.Insert() Cy_List.First_Entry.CREP := CREP Cy_List.First_Entry.DA := DA Cy_List.First_Entry.SA := SA Cy_List.First_Entry.Prio := Prio Cy_List.First_Entry.A_SDU := A_SDU Cy_List.First_Entry.APDU_Status := APDU_Status Cy_List.Num_entry++
GET_CNF ()	Get a service confirmation from the confirmation queue to MMAC. C_List.Num_entry— CREP := C_List.First_Entry.CREP C_List.Remove()
GET_N_IND ()	Get a service indication from the indication queue to MMAC. I_List.Num_entry— DA := I_List.First_Entry.DA SA := I_List.First_Entry.SA Prio := I_List.First_Entry.Prio VLAN_Tag := I_List.First_Entry.VLAN_Tag LT := I_List.First_Entry.LT N_SDU := I_List.First_Entry.DLSDU I_List.Remove() FN_List.Ibuffer.Insert() FN_List.Ibuffer.Num_entry++
GET_A_IND ()	Get a service indication from the indication queue to MMAC. I_List.Num_entry— DA := I_List.First_Entry.DA SA := I_List.First_Entry.SA Prio := I_List.First_Entry.Prio CREP.rt_id, A_SDU := I_List.First_Entry.DLSDU I_List.Remove() FA_List.Ibuffer.Insert() FA_List.Ibuffer.Num_entry++
GET_C_IND ()	Get a service indication from the indication queue to MMAC. I_List.Num_entry— DA := I_List.First_Entry.DA SA := I_List.First_Entry.SA Prio := I_List.First_Entry.Prio CREP.rt_id, C_SDU, APDU_Status := I_List.First_Entry.DLSDU I_List.Remove()

Function name	Operations
	FC_List.Ibuffer.Insert() FC_List.Ibuffer.Num_entry++
RESET_LIST	Reset all Lists with queued services to LMPM in the data interface to MMAC sublayer. I_List.Num_entry-- I_List.Remove() until empty C_List.Num_entry-- C_List.Remove() until empty

5.8.5 Bridges, Forwarding and Scheduling

5.8.5.1 Specific issues in bridged networks

The term bridge is used according to ISO/IEC 8802-3 and is synonym to switch. The term Real Time Ethernet (RTE) is used in this clause as abbreviated term of “switched network with Real Time Ethernet”.

Bridges are used because they can connect different network technologies at MAC level with high throughput and little configuration effort. They provide means for traffic reduction by forwarding frames selectively while repeater technologies will flood the whole network with every single frame. Thus bridges can be used in networks with a high number of stations with little performance degradation. This basic bridge feature allows an RTE-System to be independent of the size of the network. The protocol is designed to make fully use of highest possible speed even if a part of the system use links with lower performance.

Some features of this RTE may be used in combination with other types of networks but this specification assumes a network with full-duplex links of ISO/IEC 8802-3 without collisions. Bridges connect the point-to-point links. Two nodes cannot be directly connected and there is always at least one bridge function in between. Disadvantages of bridges are the additional delay/jitter and the reduced availability due to active components. Thus, traditional approaches in bridging do not provide solutions for all RT requirements.

The bridge approach should fulfill the following requirements for a RT network:

- Protocol has no restrictions in structuring of an automation system (star <=> line, number of nodes)
- Topologies with redundant links shall be used for enhanced communication availability with no significant delays in case of a single disturbance or failure in the network
- Efficient use of bandwidth
- Time synchronized actions for any group of nodes – the precision depends on the number and capabilities of the bridges between the nodes (the approach shall allow to synchronize more than 20 nodes in a line with a deviation between clocks below 1 µs)
- Any DTE according to ISO/IEC 8802-3 can be connected to this type of network without specific RT restrictions on communication implementation and protocol usage.

5.8.5.2 Time-synchronized actions

Time-synchronized actions are an absolute necessity in many tasks in industrial automation. Time-synchronized actions cannot be directly triggered by sync frames in a bridge-based network because

- the delay in bridges is within the us range. Thus, the number of bridges between sender and receiver will cause an time offset.
- as ISO/IEC 8802.3 is non preemptive, a non RT frame can cause a delay of more than 100 us on a single link.

To achieve the required synchronization of nodes, special support is required in the DTEs and bridges to

- measure the propagation delay of a frame between two nodes
- allow calculation of all external and internal delays from a clock synchronization master to any node

Each node adapts its own local time base to that of a clock synchronization master.

5.8.5.3 Real-time communication (RT class 3)

Real-time communication should guarantee transfers within a bounded time frame in all situations, including

- any load or even overload situation resulting from other Ethernet/Internet communication
- any network topology including large numbers of bridges connected in series

Real time communication shall support cyclic transfer of data. This means, that a DTE can send out data in fixed cycles. Each cycle is subdivided into intervals. An interval within a cycle is reserved for cyclic RT communication as shown in Figure 62. To fulfill the performance requirement a TDMA (time division multiple access) based approach can be used which consists of two intervals:

- Scheduled: “Red interval”; only cyclic frames can be sent and
- Unscheduled: “Green interval”; any type of frames can be sent

There shall be a section at the end of the green interval that is called “Preparation for RTE” – section or orange section. Frames have to complete within this section.

Figure 62 shows the basic principles of the control flow between unscheduled-, preparation-, and scheduled-interval.

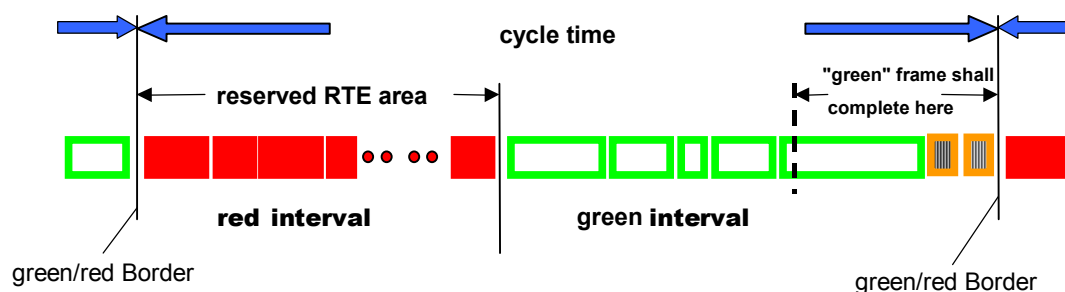


Figure 62 — Green and Red intervals and interval transitions

NOTE in the preparation section between green and red interval a non RT frame can be send if it can be completed at the green/red-Border – a RT frame can be sent out at any time.

Before any unscheduled communication is started, a check is necessary to determine whether this can be completed before the green/red-Border. If this is not possible, the transfer does not start. In the unscheduled interval, node behaves like any other node without RT features. As a frame can be started with the begin of the preparation section this interval shall allow the transfer of a frame with the maximum length.

The length of the scheduled interval depends solely on the amount of data to be transferred. This can vary from link to link. As a result, although all nodes of a RTE subnet shall be synchronized to a cycle of the same length, the distribution within a cycle is specific for each link. The scheduled interval shall be configured in that way that blocking of RT Frames at a single link is avoided if sending starts in a scheduled interval.

In a RTE network, this means that **any devices involved in Ethernet communication** can be connected to bridges with capability since they cannot influence communication in the scheduled interval.

5.8.5.4 Scheduling of Send list

See also PROFINET IO service definition. Scheduling of send list is different from the definitions of PROFINET IO service definition in the use of the send offset and the use of port identification.

In general, RT communication is based on a cyclic service of transmitting data. The frames are sent in a fixed, scheduled time scheme. The schedule shall be based on a basic timing signal.

A single DTE scheduling List of RT communication always applies to the sender of one link.

Communication can be distributed among cycles. RTE shall be able to support bundling cycles which are a multiple of the send clock. The use of this feature shall be described in the application layer protocol specification.

EXAMPLE if the basic timing signal is 1 ms, the macrocycles of 1, 2, 4, 5, or other natural multiples are allowed. Multiples like 1.5 or 2.7 are not allowed.

With RT all services scheduled in one cycle are transmitted with high priority as burst, according to IEEE 802.1D and IEEE 802.1Q.

The following rules apply for RT Class 3:

- If a path has no red interval, there is also no green/red border
- If a path has a red interval, the beginning of this interval is at a green/red border. A non RT communication begins only when it is guaranteed that it will be completed before this border is reached.
- The start of the red interval can be shifted within the cycle.

5.8.5.5 Control of reception of cyclic data

Receiving is done according to the standard rules for receiving a frame.

A DTE is able to receive RT data of the network. The selection is based on the type of RT communication. The reception of the data is monitored.

There may be different types of reception mechanisms: queued reception (as defined in IEEE 802.1D) and buffered reception. Buffered reception means that a frame is written into an overwriteable buffer. So, the next cyclic reception of the frame overwrites the previously received one. An application need not read each frame, but every time, an application reads the buffer, the most recently received frame will be read.

RT uses a database for filtering data with the items:

- Frame ID
- Length of the frame
- Optional Source MAC address

RT Class 3 can use a list of expected arrival. Thus, the data have also to arrive in the right order additionally.

5.8.5.6 Start up

A sequence of operation is necessary.

1. Time synchronization to a global time base
2. Load Scheduling information if needed
3. Establish the real time communication.

5.8.5.7 Error Recovery

The trouble shooting of a system needs events and faults for detection of problems and recovering based on errors. There are additional events/faults categories for recovery:

1. Synchronization of clocks
2. Scheduling
3. Redundancy status

In a redundant system is the strategy to try to keep the communication and in a non-redundant system to shed in an initial state.

5.8.5.8 Cyclic services forwarding for bridges

5.8.5.8.1 General

The guarantee that a specified number of RT frames will be transferred within a predefined time, not only requires reservation of the communication path but also reservation of buffer

space if buffering becomes necessary on the bridge. This buffer space is shared with the regular Ethernet communication that is subject to the mechanisms of the flow control. A reservation for RT traffic therefore requires that

- the mechanisms for flow control of RT traffic are disabled
- the size of the space to be reserved can be selected **locally** (no global configuration)

The maximum necessary storage space required on a node corresponds exactly to the amount incoming from more than one port.

5.8.5.8.2 MAC Address based forwarding optimization

The length of the RT interval is determined automatically in every bridge. The address-based RT traffic requires no configured, previously stored switching information. Nor is global configuration of the length of the RT communication necessary. For determining the required duration **locally**, two mechanisms are supported

- At the beginning, a RT communication of minimum length is set. A timeout counter is assigned. With each RT communication, this counter is automatically reset. If the counter expires, the RT communication is ended.
- Red interval length is set explicitly.

Which method should be used depends on the particular application or requirements.

5.8.5.8.3 Schedule-based RT Communication

A RT transmission between two endpoints of a connection path is possible only when there are information about sending (Destination Address, FrameID, Send-cycle, Source of Data) and receiving (Source Address, FrameID, Destination of Data) on both ends.

To transfer an RT Class 3 frame from a sender to a recipient, there are not only such control information on the sender and recipient but also on all nodes through which the frame passes. Source port and destination port are independent of the MAC addresses in the frame (neither learning nor forwarding procedures of IEEE 802.1D are used at schedule based RT Class 3 communication). The sender should enter a specific Multicast MAC address as destination as it would be necessary for address-based forwarding.

RT frames shall arrive in the expected **sequence**; in other words, the sequence at the recipient shall match a predefined schedule. Frames out of order will be discarded. Whether or not an arriving frame is the expected RT frame is checked based on the frame ID and, if present, on the MAC source address. The length is checked additionally.

The time specified in the schedule defines at which the frame is forwarded to one or more Ethernet ports. If the expected frame does not arrive at the time of monitoring, a diagnostic entry is generated and a substitute frame should be sent. A substitute frame is identified by a transfer status $\neq 0$. Since it is only possible to determine whether the expected frame has arrived after evaluating the FCS field, the transfer status shall be positioned at the end of the frame immediately before the (newly generated) FCS field to minimize forwarding delay. Substitute frames are formally correct frames that result in a diagnostic entry being made and support the localization of errors. For the user interface of a recipient, they have the same effect as a missing frame.

Within the red interval, it is acceptable for a frame to arrive earlier than expected. The buffering capability is a device feature and shall be greater than tbd Bytes (the actual size shall be a configuration parameter).

As long as the receive buffer does not overflow and as long as the sequence of scheduling is maintained, a RT frame can arrive, no matter how early, within the red interval. This type of buffering allows compensation of the timing inaccuracies shown in Figure 63.

There is a minimal time needed between Receive and Send of a message (this time is a device parameter)

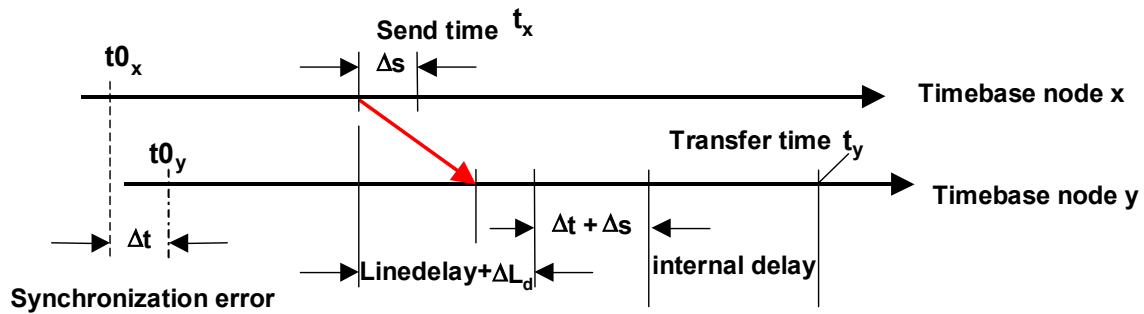


Figure 63 — Possible Time Inaccuracies

These include the following:

- Synchronization error Δt : Two neighboring nodes can never be exactly synchronized.
- Send error Δs : There is some jitter in the start of frame
- Propagation time error ΔL_d : The real line delay may vary from the delay calculated.
- Internal delay

5.8.5.9 Redundant Communication

The network is a key component of the reliability of an industrial automation system. The algorithms used for reliable switched networks are described in IEEE 802.1D. These protocols transfer a meshed network into a tree by blocking specific ports. There are some restrictions on recovery time in case of a failure, which are not acceptable in RT systems.

Redundant paths, combined with the forwarding protocols of the bridges, lead to circulating frames especially with broadcast/multicast addressing. For this reason, during startup a network is searched for possible circular routes and these are logically disconnected at a suitable location. The redundant path is not used for forwarding user data. If problems occur, communication has to reconfigure the paths. This reconfiguration includes a search for circular routes within the network and can run into tens of seconds (depending on the topology and size of the network). The method for identifying circular routes is the "Spanning Tree" or "Rapid Spanning Tree" known from IEEE 802.1D. The simpler the topology and the smaller the network, the faster a problem can be located and dynamic reconfiguration implemented. With ring topologies, special protocols allow recovery times in the 50 ms range. These recovery times are acceptable in some areas of application of such networks but such temporary breaks in operation are unacceptable for many real-time critical applications in automation.

In principle, this mechanism is not capable of achieving a completely smooth, transparent switchover. To avoid breaks, permanent redundant transmission on disjunctive paths is necessary. Permanent use of redundant paths is possible if the routing information for transferring a frame is stored in advance on all nodes involved in RT communication. Depending on the capabilities of the scheduling software, this is possible for any mesh network. For address-based forwarding, permanent transmission over redundant paths is supported only for simple ring topologies.

5.8.5.10 Redundancy support for bridges

In a network consisting of many point-to-point links, redundant paths are needed to have a reliability like shared media. Redundant paths are also necessary, for example, to avoid the most serious disadvantage of point-to-point links – the failure of one node means the failure of the entire communication system. A switchover with short reaction times is only possible when there is permanent redundant transmission on disjunctive paths. Redundant transmission automatically means greater load on the total bandwidth available in the network.

The enhancement of reliability of data transfer described here applies to ring structures and RT frames only. Others may use the mechanism as defined in 802.1Q.

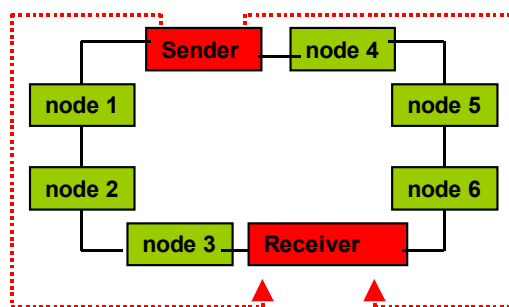


Figure 64 — Using Medium Redundancy

For schedule-based RT communication, in which the path for each frame sent is specified in the control information stored on every node involved, redundant transmission does not require any extensions of the protocol. Redundancy is specified by the scheduling. Any redundant network topologies are possible.

For address-based RT communication, in which the recipient is located solely based on the MAC destination address and in which redundant paths are “disabled” to avoid circulating frames, special support is necessary. Only a simple ring shall be supported as the redundant topology. Support consists of the following:

- transparent splitting into two redundant paths
- communication over disabled paths
- detection and elimination of circulating frames.

It should also be possible to implement redundant acyclic RT communication but multiple reception causes extra load on the recipient.

Redundant transmission should only take place on redundant paths. Two ports of a bridge that are part of a ring are identified as R-ports. All other ports are identified as nR-ports. The forwarding procedures are:

- If a frame is an redundant RT frame **and**
- If a destination port is an R-port (specified in the MAC address table entry) then **in addition to** this destination port, the frame is always passed on over the other R-port. A frame is never forwarded to the receiving port (no mirroring possible). As shown in Figure 65, this rule covers the transparent splitting, forwarding, and extraction from a ring.

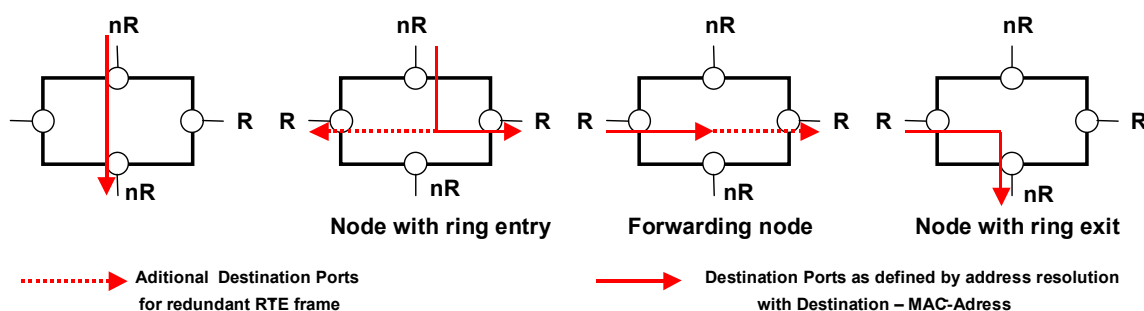


Figure 65 — Locating the Destination for redundant RT Frames

The detection (and elimination) of a circulating frame is based on the principle that a frame that is placed on a ring originally over a non-redundant port shall **not** later be received via a redundant port of the same node. If it is subsequently received by a redundant port it is destroyed. The detection of this situation is linked to the MAC-SA (source address) of the sending node because the MAC-SA of a frame shall be stored (learnt) at the point at which

the frame is placed on the network. At the point at which a redundant RT frame is placed on the network, and extra feed bit (F bit) is set in the MAC-SA table entry as an identifier. Entries with the F bit set, are also protected from removal if the address table is modified (reconfiguration).

To prevent redundant RT frames from circulating, they are also sent via a “disabled” port if this is an R-port.

5.8.5.11 Start up for bridges

A sequence of operation is necessary.

- Time synchronization to a global time base is to set
- Scheduling information if needed is to load
- Establish the real time communication

5.8.5.12 Error Recovery for bridges

The trouble shooting of a system needs events and faults for detection of problems and recovering based on errors. There are additional events/faults categories for recovery:

- Synchronization of clocks
- Scheduling
- Redundancy status

In a redundant system is the strategy to try to keep the communication and in a non-redundant system to shed in an initial state.

5.8.5.13 Configuration data for the red interval

Each reception or sending of a RT frame shall be configured.

The actions are split among different macro cycles thus the configuration has to be done for each cycle of each macrocycle. Each cycle of each macrocycle is represented by a list for each port with entries, which contain the following parameters:

- destinationPort: the port, on which the frame shall be sent. Set only on reception ports. There may be more than one destinationPort for a frame.
- local: determines if the frame shall be received locally on the bridge. Frames may be received locally and be forwarded.
- frameId: identifier of the frame. A frame is only valid, if the identifier of the frame has got a higher value than the one in the frame before. Wrap-around shall be considered.
- frameLength: length of the frame, if it shall be received locally.
- macAddress: the MAC address of the destination (for sending if a substitution frame has to be sent).
- time: the time when the frame shall be sent.
- checkSource(optional): the source MAC address to be checked at reception

5.8.5.14 Error cases

5.8.5.14.1 General

The behavior in case of error depends on the error itself. If possible, a substitute frame should be forwarded instead of the erroneous frame in case of schedule-based forwarding.

A substitute frame is a frame with a transfer state $\neq 0$. Since it is only possible to determine whether the expected frame has arrived after evaluating the FCS field, the transfer state shall be positioned at the end of the frame immediately before the (newly generated) FCS field to minimize forwarding delay. Substitute frames are formally correct frames that result in a diagnostic entry being made and support the localization of errors. A substitute frame is dropped at final destination. Substitute Frames are send with own MAC Address as source address and the RT Class 3 Multicast address as destination address and with the FrameID

as specified in the schedule list. If the frame has been send out before the error occurs, the data received so far correctly shall be send.

5.8.5.14.2 Frame received too early

A RT frame received too early shall be stored until its forwarding time and then it shall be forwarded regularly.

The buffering capability is a device feature and shall be at least tbd Bytes.

As long as the receive buffer does not overflow and as long as the sequence of scheduling is maintained, an RT frame can arrive, no matter how early, within the red interval.

5.8.5.14.3 Frame with wrong frameld

RT frames with a frameld out of order shall be discarded. To react on this, a bridge has two possibilities:

- 1) The bridge may look up in the configuration table for that frame ID, if it can be found, the bridge stores the frame and does the action configured for that frame. This may be a very time-consuming search that may cause other timing problems when the next frame is received.
- 2) The bridge sends a substitute frame instead. The RTClass3Error bit in transfer state shall be set.

5.8.5.14.4 Frame not received

If a frame dedicated for schedule-based forwarding is not received in time, the bridge shall send a substitute frame instead. The RTClass3Error bit in transfer state shall be set.

5.8.5.14.5 Frame received was too short

Frames dedicated for schedule-based forwarding that are too short shall be forwarded, but in the transfer state, the WrongLength bit shall be set.

5.8.5.14.6 Frame received has not been found in the configuration list with wrong frameld

Frames dedicated for schedule-based forwarding whose frameID and optional source MAC address do not match with one in the configuration list, shall be discarded on every node. Not all of the configuration list shall be checked.

5.8.5.14.7 Frame with wrong FCS was received

Frames dedicated for schedule-based forwarding received with wrong FCS or alignment shall be forwarded with re-calculated FCS and the proper alignment and the bit AlignFCheckError in the transfer state field set.

5.8.5.14.8 Frame was too long

Frames dedicated for schedule-based forwarding with a length different from their configured length shall be forwarded with the WrongLength bit set in the transfer state field.

5.8.5.14.9 Frame arrived too late

Frames dedicated for schedule-based forwarding not received on time shall be dealed like frames not received.

5.8.5.14.10 Frame with out of ring source address

All frames dedicated for address based forwarding with out-of-ring source addresses (i.e. in address table and marked as specific static entry) shall be discarded on every node. Such a reception may happen in ring topologies. In this case, the ring has been passed completely by the frame and the frame shall not pass the ring a second time.

5.8.5.15 FW

5.8.5.15.1 Primitive definitions

5.8.5.15.1.1 Primitives exchanged between FW and MMAC

Table 380 shows the primitives issued by the MMAC to the FW.

Table 380 — Primitives issued by MMAC to FW

Primitive name	Source	Associated parameters	Functions
SYN_FW_Start.req	RCTL	-	
SYN_FW_Stop.req	RCTL	-	
SYN_FllwFW.cnf	MMAC	Port DA SA FU_list FU_status	
SYN_FllwFW.ind	MMAC	Port DA SA FU_list FU_status	
SYN_SyncFW.cnf	MMAC	Port DA SA CS_list CS_status	
SYN_SyncFW.ind	MMAC	Port DA SA CS_list CS_status	

Table 381 shows the primitives issued by the FW to the MMAC

Table 381 — Primitives issued by FW to MMAC

Primitive name	Source	Associated parameters	Functions
SYN_FllwFW.req	FW	Port DA SA FU_list FU_status	
SYN_SyncFW.req	FW	Port DA SA CS_list CS_status	

5.8.5.15.1.2 Parameters of FW primitives

Table 382 shows all parameters used with primitives between the MMAC and the FW.

Table 382 — Parameters used with primitives exchanged between MMAC and FW

Parameter Name	Description
CS_list	Cycle Sync List
CS_status	Cycle Syn Status
DA	Destination Address
FU_list	Follow Up List
FU_status	Follow Up Status
Port	
SA	Source Address

5.8.5.15.2 FW state table

The FW state table is shown in Table 383.

Table 383 — FW state table

#	Current State	Event /Condition =>Action	Next State
1	P-ON	<pre> => if (FU) for m := TMList.MaxCnEntry do { for i := all Ports do { TMList[m].Fw[i].SyncDelay := 0 TMList[m].Fw[i].LocalDelay := 0 TMList[m].Fw[i].FwupInd := FALSE TMList[m].Fw[i].SyncReqCnf := TRUE TMList[m].Fw[i].SyncReq := FALSE } } </pre>	CLOSED
2	CLOSED	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) => ignore	CLOSED
3	CLOSED	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) => ignore	CLOSED
4	CLOSED	SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status) => ignore	CLOSED
5	CLOSED	SYN_FilwFW.cnf (Port, DA, SA, FU_list, FU_status) => ignore	CLOSED
6	CLOSED	SYN_FW_Start.req() => start FW-SM	OPEN
7	CLOSED	SYN_FW_Stop.req() => ignore	CLOSED
8	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /!({SA, ProtVar} in TMList) => ignore	OPEN
9	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /!FU && {SA, ProtVar} in TMList && Port <= 0 => for i := all op. Ports \ -Port do SYN_SyncFW.req (i, DA, SA, CS_list, CS_status)	OPEN

#	Current State	Event /Condition =>Action	Next State
10	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /!FU && {SA, ProtVar} in TMList && Port > 0 => SYN_SyncFW.req (Port, DA, SA, CS_list, CS_status)	OPEN
11	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && Port < 0 && CS_list.S_SDU.Fwup := FALSE => CS_list.S_SDU.Fwup := TRUE for i := all op. Ports \ -Port do if (TMList{SA, ProtVar}.Fw[i].SyncReqCnf && !TMList{SA, ProtVar}.Fw[i].FwupInd) { TMList{SA, ProtVar}.Fw[i].SyncReq := TRUE TMList{SA, ProtVar}.Fw[i].SyncReqCnf := FALSE TMList{SA, ProtVar}.Fw[i].FwupInd := TRUE TMList{SA, ProtVar}.Fw[i].SyncDelay := 0 SYN_SyncFW.req (i, DA, SA, CS_list, CS_status) }	OPEN
12	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && Port < 0 && CS_list.S_SDU.Fwup := TRUE => for i := all op. Ports \ -Port do if (TMList{SA, ProtVar}.Fw[i].SyncReqCnf[i] && !TMList{SA, ProtVar}.Fw[i].FwupInd[i]) { TMList{SA, ProtVar}.Fw[i].SyncReq[i] := TRUE TMList{SA, ProtVar}.Fw[i].SyncReqCnf[i] := FALSE SYN_SyncFW.req (i, DA, SA, CS_list, CS_status) }	OPEN
13	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && Port == 0 => CS_list.S_SDU.Fwup := TRUE for i := all op. Ports do if (TMList{SA, ProtVar}.Fw[i].SyncReqCnf && !TMList{SA, ProtVar}.Fw[i].FwupInd) { TMList{SA, ProtVar}.Fw[i].SyncReq := TRUE TMList{SA, ProtVar}.Fw[i].SyncReqCnf := FALSE TMList{SA, ProtVar}.Fw[i].FwupInd := TRUE TMList{SA, ProtVar}.Fw[i].SyncDelay := 0 SYN_SyncFW.req (Port, DA, SA, CS_list, CS_status) }	OPEN

#	Current State	Event /Condition =>Action	Next State
14	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /FU && Port > 0 && CS_list.S_SDU.Fwup := FALSE => CS_list.S_SDU.Fwup := TRUE if (TMList{SA, ProtVar}.Fw[Port].SyncReqCnf && !TMList{SA, ProtVar}.Fw[Port].FwupInd) { TMList{SA, ProtVar}.Fw[Port].SyncReq := TRUE TMList{SA, ProtVar}.Fw[Port].SyncReqCnf := FALSE TMList{SA, ProtVar}.Fw[Port].FwupInd := TRUE TMList{SA, ProtVar}.Fw[Port].SyncDelay := 0 SYN_SyncFW.req (Port, DA, SA, CS_list, CS_status) }	OPEN
15	OPEN	SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && Port > 0 && CS_list.S_SDU.Fwup := TRUE => if (TMList{SA, ProtVar}.Fw[Port].SyncReqCnf && !TMList{SA, ProtVar}.Fw[Port].FwupInd) { TMList{SA, ProtVar}.Fw[Port].SyncReq[Port] := TRUE TMList{SA, ProtVar}.Fw[Port].SyncReqCnf[i] := FALSE SYN_SyncFW.req (Port, DA, SA, CS_list, CS_status) }	OPEN
16	OPEN	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) /!({SA, ProtVar} in TMList) => ignore	OPEN
17	OPEN	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) /!FU && {SA, ProtVar} in TMList && Port <= 0 => for i := all op. Ports \ -Port do SYN_FilwFW.req (i, DA, SA, FU_list, FU_status)	OPEN
18	OPEN	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) /!FU && {SA, ProtVar} in TMList && Port > 0 => SYN_FilwFW.req (Port, DA, SA, FU_list, FU_status)	OPEN

#	Current State	Event /Condition =>Action	Next State
19	OPEN	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) /FU && {SA, ProtVar} in TMList && Port < 0 => for i := all op. Ports \ -Port do if (TMList{SA, ProtVar}.Fw[i].SyncReq) { if (TMList{SA, ProtVar}.Fw[i].SyncReqConf) { if (!TMList{SA, ProtVar}.Fw[i].FwupInd) { TMList{SA, ProtVar}.Fw[i].FwupInd := TRUE FU_list.S_SDU.Delay := FU_list.S_SDU.Delay + TMList{SA, ProtVar}.Fw[i].LocalDelay SYN_FilwFW.req (i, DA, SA, FU_list, FU_status) } } else { if (!TMList{SA, ProtVar}.Fw[i].FwupInd) { TMList{SA, ProtVar}.Fw[i].FwupInd := TRUE TMList{SA, ProtVar}.Fw[i].SyncDelay := FU_list.S_SDU.Delay } } } 	OPEN
20	OPEN	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) /FU && {SA, ProtVar} in TMList && Port > 0 => if (TMList{SA, ProtVar}.Fw[Port].SyncReq) { if (TMList{SA, ProtVar}.Fw[Port].SyncReqConf) { if (!TMList{SA, ProtVar}.Fw[Port].FwupInd) { TMList{SA, ProtVar}.Fw[Port].FwupInd := TRUE FU_list.S_SDU.Delay := FU_list.S_SDU.Delay + TMList{SA, ProtVar}.Fw[Port].LocalDelay SYN_FilwFW.req (Port, DA, SA, FU_list, FU_status) } } else { if (!TMList{SA, ProtVar}.Fw[Port].FwupInd) { TMList{SA, ProtVar}.Fw[Port].FwupInd := TRUE TMList{SA, ProtVar}.Fw[Port].SyncDelay := FU_list.S_SDU.Delay } } } 	OPEN
21	OPEN	SYN_FilwFW.ind (Port, DA, SA, FU_list, FU_status) /!FU && {SA, ProtVar} in TMList && Port == 0 => ignore	OPEN

#	Current State	Event /Condition =>Action	Next State
22	OPEN	SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status) /!FU => ignore	OPEN
23	OPEN	SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status) /!(SA, ProtVar} in TMList) => ignore	OPEN
24	OPEN	SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && !TMList{SA, ProtVar}.Fw[Port].SyncReqCnf && !TMList{SA, ProtVar}.Fw[Port].FwupInd => TMList{SA, ProtVar}.Fw[Port].FWLocalDelay := CS_llist.S_SDU.T1 – CS_list.S_SDU.T2 TMList{SA, ProtVar}.Fw[Port].SyncReqCnf := TRUE	OPEN
25	OPEN	SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && !TMList{SA, ProtVar}.Fw[Port].SyncReqCnf && TMList{SA, ProtVar}.Fw[Port].FwupInd => TMList{SA, ProtVar}.Fw[Port].SyncReqCnf := TRUE FU_list.S_SDU.Delay := TMList{SA, ProtVar}.Fw[Port].SyncDelay + CS_llist.S_SDU.T1 – CS_list.S_SDU.T2 FU_status := SV SYN_FllwFW.req (Port, DA, SA, FU_list, FU_status)	OPEN
26	OPEN	SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status) /FU && {SA, ProtVar} in TMList && TMList{SA, ProtVar}.Fw[Port].SyncReqCnf => ignore	OPEN
27	OPEN	SYN_FllwFW.cnf (Port, DA, SA, FU_list, FU_status) /!FU => ignore	OPEN
28	OPEN	SYN_FllwFW.cnf (Port, DA, SA, FU_list, FU_status) /!(SA, ProtVar} in TMList) => ignore	OPEN
29	OPEN	SYN_FllwFW.cnf (Port, DA, SA, FU_list, FU_status) /FU && {SA, ProtVar} in TMList => TMList{SA, ProtVar}.Fw[Port].SyncReq := FALSE TMList{SA, ProtVar}.Fw[Port].FwupInd := FALSE	OPEN
30	OPEN	SYN_FW_Start.req() => ignore	OPEN
31	OPEN	SYN_FW_Stop.req() => stop FW-SM	CLOSED

5.8.5.16 IFW

5.8.5.16.1 Primitive definitions

5.8.5.16.1.1 Primitives exchanged between IFW and FSPM

Table 384 shows the primitives issued by the FSPM to the IFW.

Table 384 — Primitives issued by FSPM to IFW

Primitive name	Source	Associated parameters	Functions
IFW_IRT_Schedule.req	FSPM	Port Phase	
IFW_Schedule_add.req	FSPM	Port Sched_list	
IFW_Schedule_remove.req	FSPM	Port	
IFW_SetFWState.req	FSPM	Port FWState	
IFW_SetLEState.req	FSPM	Port LEState	

Table 385 shows the primitives issued by the IFW to the FSPM

Table 385 — Primitives issued by IFW to FSPM

Primitive name	Source	Associated parameters	Functions
IFW_Schedule_add.cnf	IFW	Port	
IFW_Schedule_remove.cnf	IFW	Port	
IFW_Error.ind	IFW	CREP ERRCLS ERRCODE	

5.8.5.16.1.2 Parameters of IFW primitives

Table 386 shows all parameters used with primitives between the LMPM and the IFW.

Table 386 — Parameters used with primitives exchanged between LMPM and IFW

Parameter Name	Description
FWState	
LEState	
Phase	
Port	
Sched_List	

5.8.5.16.2 IFW state table

The IFW state table is shown in Table 387.

Table 387 — IFW state table

#	Current State	Event /Condition =>Action	Next State
1	NORMAL	/FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DA in ADD_DB => for all Ports in ADD_DB(FWC_List(Port).First_entry.DA) PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove()	NORMAL
2	NORMAL	/FWC_List(Port).Num_entry<>0 &&!(FWC_List(Port).First_entry.DA in ADD_DB) => for all Ports \ Port PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove()	NORMAL

#	Current State	Event /Condition =>Action	Next State
3	NORMAL	/FWC_List(Port).Num_entry ==0 && FWA_List(Port).Num_entry<>0 && FWA_List(Port).First_entry.DA in ADD_DB => for all Ports in ADD_DB(FWA_List(Port).First_entry.DA) PUT_AREQ (Port) FWA_List(Port).Num_entry-- FWA_List(Port).Remove()	NORMAL
4	NORMAL	/FWC_List(Port).Num_entry ==0 && FWA_List(Port).Num_entry<>0 &&!(FWA_List(Port).First_entry.DA in ADD_DB) => for all Ports \ Port PUT_AREQ (Port) FWA_List(Port).Num_entry-- FWA_List(Port).Remove()	NORMAL
5	NORMAL	/FWC_List(Port).Num_entry ==0 && FWA_List(Port).Num_entry ==0 && FWN_List(Port).Num_entry<>0 && FWN_List(Port).First_entry.DA in ADD_DB => for all Ports in ADD_DB(FWN_List(Port).First_entry.DA) PUT_NREQ (Port) FWN_List(Port).Num_entry-- FWN_List(Port).Remove()	NORMAL
6	NORMAL	/FWC_List(Port).Num_entry ==0 && FWA_List(Port).Num_entry ==0 && FWN_List(Port).Num_entry<>0 &&!(FWN_List(Port).First_entry.DA in ADD_DB) => for all Ports \ Port PUT_NREQ (Port) FWN_List(Port).Num_entry-- FWN_List(Port).Remove()	NORMAL
7	NORMAL	IFW_Schedule_add.req(Port, Sched_list) => Store Sched_list IFW_Schedule_add.cnf(Port)	NORMAL
8	NORMAL	IFW_Schedule_remove.req(Port) => Delete Sched_list IFW_Schedule_remove.cnf(Port)	NORMAL
9	NORMAL	IFW_IRT_Schedule.req(Port, Phase) /Sched_list(Phase) <> NIL => i := 1 Start IRT_Timer(Sched_List(Phase)(i).TimeOf)	IRTFW
10	NORMAL	IFW_IRT_Schedule.req(Port, Phase) /Sched_list(Phase) == NIL =>	NORMAL
11	NORMAL	IFW_SetFWState.req(Port, FWState) => FW_Stat := FWState	NORMAL
12	NORMAL	IFW_SetLEState.req(Port, LEState) => LE_Stat := LEState	NORMAL
13	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) <> NIL && FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DLSDU.rt_id == Sched_List(Phase)(i).rt_id && FWC_List(Port).First_entry.DLSDU.Len == Sched_List(Phase)(i).Len => for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove() i++ Start IRT_Timer(Sched_List(Phase)(i).TimeOf)	IRTFW

#	Current State	Event /Condition =>Action	Next State
14	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) == NIL && FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DLSDU.rt_id == Sched_List(Phase)(i).rt_id && FWC_List(Port).First_entry.DLSDU.Len == Sched_List(Phase)(i).Len => for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove()	NORMAL
15	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) <> NIL && FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DLSDU.rt_id == Sched_List(Phase)(i).rt_id && FWC_List(Port).First_entry.DLSDU.Len <> Sched_List(Phase)(i).Len => C_SDU.Len := Sched_List(Phase)(i).Len CREP.rt_id := Sched_List(Phase)(i).rt_id APDU_Status.TransferStatus := WRONG_LENGTH for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) ----- sdu is undefined FWC_List(Port).Num_entry-- FWC_List(Port).Remove() i++ Start IRT_Timer(Sched_List(Phase)(i).TimeOf) IFW_Error.ind(CREP, ERRCLS, ERRCODE)	IRTFW
16	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) == NIL && FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DLSDU.rt_id == Sched_List(Phase)(i).rt_id && FWC_List(Port).First_entry.DLSDU.Len <> Sched_List(Phase)(i).Len => C_SDU.Len := Sched_List(Phase)(i).Len CREP.rt_id := Sched_List(Phase)(i).rt_id APDU_Status.TransferStatus := WRONG_LENGTH for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove() IFW_Error.ind(CREP, ERRCLS, ERRCODE)	NORMAL
17	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) <> NIL && FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DLSDU.rt_id <> Sched_List(Phase)(i).rt_id => C_SDU.Len := Sched_List(Phase)(i).Len CREP.rt_id := Sched_List(Phase)(i).rt_id APDU_Status.TransferStatus := IRT_ERROR for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove() i++ Start IRT_Timer(Sched_List(Phase)(i).TimeOf) IFW_Error.ind(CREP, ERRCLS, ERRCODE)	IRTFW

#	Current State	Event /Condition =>Action	Next State
18	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) == NIL && FWC_List(Port).Num_entry<>0 && FWC_List(Port).First_entry.DLSDU.rt_id <> Sched_List(Phase)(i).rt_id => C_SDU.Len := Sched_List(Phase)(i).Len CREP.rt_id := Sched_List(Phase)(i).rt_id APDU_Status.TransferStatus := IRT_ERROR for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) FWC_List(Port).Num_entry-- FWC_List(Port).Remove() IFW_Error.ind(CREP, ERRCLS, ERRCODE)	NORMAL
19	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) <> NIL && FWC_List(Port).Num_entry==0 => C_SDU.Len := Sched_List(Phase)(i).Len CREP.rt_id := Sched_List(Phase)(i).rt_id APDU_Status.TransferStatus := IRT_ERROR for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) i++ Start IRT_Timer(Sched_List(Phase)(i).TimeOf) IFW_Error.ind(CREP, ERRCLS, ERRCODE)	IRTFW
20	IRTFW	IRT_Timer expired /Sched_List(Phase)(i+1) == NIL && FWC_List(Port).Num_entry==0 => C_SDU.Len := Sched_List(Phase)(i).Len CREP.rt_id := Sched_List(Phase)(i).rt_id APDU_Status.TransferStatus := IRT_ERROR for all Ports in Sched_List(Phase)(i).PortList PUT_CREQ (Port) IFW_Error.ind(CREP, ERRCLS, ERRCODE)	NORMAL

5.8.5.16.3 Functions

All functions of the IFW are summarized in Table 388.

Table 388 — IFW function table

Function name	Operations
PUT_NREQ	Put a service request to the service queue to MMAC. FWN_List.Num_entry-- N_List.Insert() N_List.Last_Entry.CREP := NIL N_List.Last_Entry.DA:=FWN_List.First_Entry.DA := DA N_List.Last_Entry.SA:=FWN_List.First_Entry.SA := SA N_List.Last_Entry.Prio:=FWN_List.First_Entry.Prio := Prio N_List.Last_Entry.VLAN_Tag:=FWN_List.First_Entry.VLAN_Tag := VLAN_Tag N_List.Last_Entry.LT:=FWN_List.First_Entry.LT := LT N_List.Last_Entry.N_SDU:=FWN_List.First_Entry.N_SDU := N_SDU FWN_List.Remove() N_List.Num_entry++
PUT_AREQ	Put a service request to the service queue to MMAC. FWA_List.Num_entry-- Ac_List.Insert() Ac_List.Last_Entry.CREP := NIL Ac_List.Last_Entry.DA:=FWA_List.First_Entry.DA Ac_List.Last_Entry.SA:=FWA_List.First_Entry.SA Ac_List.Last_Entry.Prio:=FWA_List.First_Entry.Prio := Prio Ac_List.Last_Entry.VLAN_Tag:=FWA_List.First_Entry.VLAN_Tag Ac_List.Last_Entry.ASDU:=FWA_List.First_Entry.A_SDU

Function name	Operations
	FWA_List.Remove() Ac_List.Num_entry++
PUT_CREQ	Put a service request to the service queue to MMAC. FWC_List.Num_entry-- Cy_List.Insert() Cy_List.Last_Entry.CREP := NIL Cy_List.Last_Entry.DA:=FWC_List.First_Entry.DA := DA Cy_List.Last_Entry.SA:=FWC_List.First_Entry.SA := SA Cy_List.Last_Entry.Prio:=FWC_List.First_Entry.Prio := Prio Cy_List.Last_Entry.VLAN_Tag:=FWC_List.First_Entry.VLAN_Tag := VLAN_Tag Cy_List.Last_Entry.LT:=FWC_List.First_Entry.APDU_Status := APDU_Status Cy_List.Last_Entry.N_SDU:=FWC_List.First_Entry.N_SDU := N_SDU Cy_List.Num_entry++ FWC_List.Remove()

5.8.6 MMAC

5.8.6.1 Primitive definitions

5.8.6.1.1 Primitives exchanged between LMPM and MMAC

Table 389 shows the primitives issued by the LMPM to the MMAC.

Table 389 — Primitives issued by LMPM to MMAC

Primitive Name	Associated Parameters
MMAC_RESET.req	none

Table 390 shows the primitives issued by the MMAC to the LMPM.

Table 390 — Primitives issued by MMAC to LMPM

Primitive Name	Associated Parameters
MMAC_RESET.cnf	none

5.8.6.1.2 Primitives exchanged between MMAC and MAC

Table 391 shows the service primitives including their associated parameters issued by the MMAC and received by the MAC.

Table 391 — Primitives issued by MMAC to MAC

Primitive name	Source	Associated parameters	Functions
M_UNITDATA.req	MMAC	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	

Table 392 shows the service primitives including their associated parameters issued by the MAC and received by the MMAC.

Table 392 — Primitives issued by MAC to MMAC

Primitive name	Source	Associated parameters	Functions
M_UNITDATA.ind	MAC	frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier, rif_information (optional)	
M_UNITDATA.cnf	MAC	-	

5.8.6.1.3 Primitives exchanged between MMAC and TMM

Table 393 shows the service primitives including their associated parameters issued by the TMM and received by the MMAC.

Table 393 — Primitives issued by TMM to MMAC

Primitive name	Source	Associated parameters	Functions
MMAC_IRT_Set_Interval.req	TMM	Port Interv IntervLen	
SYN_Sync.req	TMM	Port DA SA CS_list CS_status	

Table 394 shows the service primitives including their associated parameters issued by the MMAC and received by the TMM.

Table 394 — Primitives issued by MMAC to TMM

Primitive name	Source	Associated parameters	Functions
SYN_Sync.cnf	MMAC	Port DA SA CS_list CS_status	
SYN_Sync.ind	MMAC	Port DA SA CS_list CS_status	

5.8.6.1.4 Primitives exchanged between MMAC and RCTL

Table 395 shows the service primitives including their associated parameters issued by the MMAC and received by the RCTL.

Table 395 — Primitives issued by MMAC to RCTL

Primitive name	Source	Associated parameters	Functions
SYN_Delay_Req.cnf	MMAC	Port DA SA DQ_list DQ_status	
SYN_Delay_Req.ind	MMAC	Port DA SA DQ_list DQ_status	

Primitive name	Source	Associated parameters	Functions
SYN_Delay_Rsp.cnf	MMAC	Port DA SA DS_list DS_status	
SYN_Delay_Rsp.ind	MMAC	Port DA SA DS_list DS_status	
SYN_Fllw.ind	MMAC	Port DA SA FU_list FU_status	

Table 396 shows the service primitives including their associated parameters issued by the RCTL and received by the MMAC.

Table 396 — Primitives issued by RCTL to MMAC

Primitive name	Source	Associated parameters	Functions
SYN_Delay_Req.req	RCTL	Port DA SA DQ_list DQ_status	
SYN_Delay_Rsp.req	RCTL	Port DA SA DS_list DS_status	

5.8.6.1.5 Primitives exchanged between MMAC and FW

Table 397 shows the service primitives including their associated parameters issued by the MMAC and received by the FW.

Table 397 — Primitives issued by MMAC to FW

Primitive name	Source	Associated parameters	Functions
SYN_FllwFW.cnf	MMAC	Port DA SA FU_list FU_status	
SYN_FllwFW.ind	MMAC	Port DA SA FU_list FU_status	
SYN_SyncFW.cnf	MMAC	Port DA SA CS_list CS_status	
SYN_SyncFW.ind	MMAC	Port DA SA CS_list CS_status	

Table 398 shows the service primitives including their associated parameters issued by the FW and received by the MMAC.

Table 398 — Primitives issued by FW to MMAC

Primitive name	Source	Associated parameters	Functions
SYN_FllwFW.req	FW	Port DA SA FU_list FU_status	
SYN_SyncFW.req	FW	Port DA SA CS_list CS_status	

5.8.6.1.6 Parameters of DMPM primitives

A data request primitive is invoked in order to generate a M_UNITDATA.request primitive, as defined in the Internal Sublayer Service, ISO/IEC 15802-3.

The frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority and frame_check_sequence parameters are as defined for the M_UNITDATA.req primitive of the Internal Sublayer service.

The definition of the canonical_format_indicator parameter is as defined for the E-ISS data indication. The vlan_classification parameter carries the VLAN classification assigned to the frame by the Ingress rules. The rif_information parameter, if present, carries the value of any RIF information to be associated with the request. The include_tag parameter carries a Boolean value. True indicates to the service provider that the mac_service_data_unit parameter of the data request shall include a Tag Header. False indicates that a Tag Header shall not be included. The frame_type, mac_action, destination_address, source_address and frame_check_sequence parameters carry values equal to the corresponding parameters in the received data indication.

5.8.6.2 State machine description

The MMAC is responsible for mapping LMPM services to MMAC.

The other main task is the correct execution of services that are passed to and returned to LMPM via queues.

Local variables of the MMAC

Port_State

This local variable contains the state of the port.

destination_address

This local variable contains the destination MAC address.

Cy_List

This local variable contains the cyclic service send list.

Ac_List

This local variable contains the acyclic service send list.

ListOrgAdd

This local variable contains the address list.

N_List

This local variable contains non realtime service send list.

5.8.6.3 MMAC state table

The MMAC state table is shown in Table 399.

Table 399 — MMAC state table

#	Current State	Event /Condition =>Action	Next State
1	IDLE	/Port_State != Forward && !(destination_address in ListOrgAdd) && Cy_List.Num_entry <> 0 => SETUP_CREQ	IDLE
2	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Cy_List.Num_entry <> 0 => SETUP_CREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
3	IDLE	/Port_State != Forward && !(destination_address in ListOrgAdd) && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 => SETUP_AREQ	IDLE
4	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Interval = Green && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 => SETUP_AREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
5	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Interval = Yellow && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 && CalcTx(Ac_List.First_Entry.Ac_SDU.len) =< (Start_Red-TCC.cv) => SETUP_AREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
6	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Interval = Yellow && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 && CalcTx(Ac_List.First_Entry.Ac_SDU.len) > (Start_Red-TCC.cv) => ignore	IDLE
7	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Interval = Red && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 => ignore	IDLE
8	IDLE	/Port_State != Forward && !(destination_address in ListOrgAdd) && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 => SETUP_NREQ	IDLE
9	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Interval = Green && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 => SETUP_NREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
10	IDLE	/Port_State == Forward (destination_address in ListOrgAdd) && Interval = Yellow && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 && CalcTx(Ac_List.First_Entry.Ac_SDU.len) =< (Start_Red-TCC.cv) => SETUP_NREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF

#	Current State	Event /Condition =>Action	Next State
11	IDLE	/(Port_State == Forward (destination_address in ListOrgAdd)) && Interval = Yellow && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 && CalcTx(Ac_List.First_Entry.Ac_SDU.len) > (Start_Red-TCC.cv) => ignore	IDLE
12	IDLE	/(Port_State == Forward (destination_address in ListOrgAdd)) && Interval = Red && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 => ignore	IDLE
13	IDLE	MMAC_RESET.req => RESET_HL_LIST Interval := GREEN MMAC_RESET.cnf	IDLE
14	IDLE / AW-CNF	MMAC_IRT_Set_Interval.req(Port, Interv, IntervLen) /Interv = Yellow => Interval = Yellow Start_Red = TCC.cv + IntervLen	IDLE / AW-CNF
15	IDLE / AW-CNF	MMAC_IRT_Set_Interval.req(Port, Interv, IntervLen) /!(Interv = Yellow) => Interval = Interv	IDLE / AW-CNF
16	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /Port_State != Forward && !(destination_address in ListOrgAdd) =>	IDLE / AW-CNF
17	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /(Port_State == Forward (destination_address in ListOrgAdd)) && mac_service_data_unit.LT = RT && mac_service_data_unit.rt_id > 0xff && mac_service_data_unit.rt_id < 0xf000 && I_BUF_C => SETUP_IND_C	IDLE / AW-CNF
18	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /(Port_State == Forward (destination_address in ListOrgAdd)) && mac_service_data_unit.LT = RT && mac_service_data_unit.rt_id > 0xffff && I_BUF_A => SETUP_IND_A	IDLE / AW-CNF
19	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /(Port_State == Forward (destination_address in ListOrgAdd)) && ((mac_service_data_unit.LT = RT && mac_service_data_unit.rt_id < 0x100) mac_service_data_unit.LT <> RT) && I_BUF_N => SETUP_IND_N	IDLE / AW-CNF
20	AW-CNF	M_UNITDATA.cnf => SETUP_CON	CHK-NEW
21	CHK-NEW	/Port_State != Forward && !(destination_address in ListOrgAdd) && Cy_List.Num_entry <> 0 => SETUP_CREQ	IDLE

#	Current State	Event /Condition =>Action	Next State
22	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Cy_list.Num_entry <> 0 => SETUP_CREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
23	CHK-NEW	/Port_State != Forward && !(destination_address in ListOrgAdd) && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 => SETUP_AREQ	IDLE
24	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Interval = Green && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 => SETUP_AREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
25	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Interval = Yellow && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 && CalcTx(Ac_List.First_Entry.Ac_SDU.len) =< (Start_Red-TCC.cv) => SETUP_AREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
26	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Cy_list.Num_entry = 0 && Ac_List.Num_entry <> 0 && ((Interval = Yellow && CalcTx(Ac_List.First_Entry.Ac_SDU.len) > (Start_Red-TCC.cv)) (Interval = Red)) =>	IDLE
27	CHK-NEW	/Port_State != Forward && !(destination_address in ListOrgAdd) && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 => SETUP_NREQ	IDLE
28	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Interval = Green && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 => SETUP_NREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
29	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Interval = Yellow && Cy_List.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 && CalcTx(N_List.First_Entry.Ac_SDU.len) =< (Start_Red-TCC.cv) => SETUP_NREQ M_UNITDATA.req (Port, frame_type, mac_action, destination_address, source_address, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
30	CHK-NEW	/(Port_State == Forward (destination_address in ListOrgAdd)) && Cy_list.Num_entry = 0 && Ac_List.Num_entry = 0 && N_List.Num_entry <> 0 && ((Interval = Yellow && CalcTx(N_List.First_Entry.Ac_SDU.len) > (Start_Red- TCC.cv)) (Interval = Red)) =>	IDLE
31	CHK-NEW	/Cy_list.Num_entry = 0 && Ac_list.Num_entry = 0 && N_list.Num_entry = 0 =>	IDLE
32	P-ON	=> import TCC	IDLE

#	Current State	Event /Condition =>Action	Next State
33	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value != NIL && (mac_service_data_unit.Seq > TMList{SA, ProtVar}.Sync.Seq) => if(!FU) CS_list.S_SDU.Delay := mac_service_data_unit.Delay + TMList{SA, ProtVar}.Delay[Port].Value CS_list.S_SDU := (mac_service_data_unit / LT, rt_id) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.RT_ID := mac_service_data_unit.rt_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list.S_SDU.T2 :=TCC.cv CS_Status := SV TMList{SA, ProtVar}.Sync.Seq := mac_service_data_unit.Seq TMList{SA, ProtVar}.Sync.PrimRed := RedSync{SA, ProtVar}[Port] TMList{SA, ProtVar}.Sync.Prim.Port := Port TMList{SA, ProtVar}.Sync.PrimFwup := FALSE TMList{SA, ProtVar}.Sync.SecPort := NIL TMList{SA, ProtVar}.Sync.SecFwup := FALSE	IDLE / AW-CNF
34	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value == NIL && mac_service_data_unit.Seq > TMList{SA, ProtVar}.Sync.Seq => if (!FU) CS_list.S_SDU.Delay := mac_service_data_unit.Delay CS_list.S_SDU := (mac_service_data_unit / LT, rt_id) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.RT_ID := mac_service_data_unit.rt_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list..S_SDU.T2 :=TCC.cv CS_Status := NDY TMList{SA, ProtVar}.Sync.Seq := mac_service_data_unit.Seq TMList{SA, ProtVar}.Sync.PrimRed := RedSync{SA, ProtVar}[Port] TMList{SA, ProtVar}.Sync.PrimPort := Port TMList{SA, ProtVar}.Sync.PrimFwup := FALSE TMList{SA, ProtVar}.Sync.SecPort := NIL TMList{SA, ProtVar}.Sync.SecFwup := FALSE SYN_Sync.ind (Port, DA, SA, CS_list, CS	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
35	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, PortVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value == NIL && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && RedSync{SA, ProtVar}[Port] && TMList{SA, ProtVar}.Sync.PrimRed && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort == NIL => if (!FU) CS_list.S_SDU.Delay := mac_service_data_unit.Delay CS_list.S_SDU := (mac_service_data_unit / LT, rt_id) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.FRAME_ID := frame_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list..S_SDU.T2 :=TCC.cv CS_Status := NDY Port := TMList{SA, ProtVar}.Sync.PrimPort TMList{SA, ProtVar}.Sync.SecPort := Port TMList{SA, ProtVar}.Sync.SecFwup := TRUE SYN_Sync.ind (Port, DA, SA, CS_list, CS_status)	IDLE / AW-CNF
36	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value != NIL && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && RedSync{SA, ProtVar}[Port] && TMList{SA, ProtVar}.Sync.PrimRed && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort == NIL => if (!FU) CS_list.S_SDU.Delay := mac_service_data_unit.Delay + TMList{SA, ProtVar}.Delay[Port].Value CS_list.S_SDU := (mac_service_data_unit / RT, SYNC) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.RT_ID := mac_service_data_unit.rt_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list..S_SDU.T2 :=TCC.cv CS_list.S_SDU.Delay := mac_service_data_unit.Delay + TMList{SA, ProtVar}.Delay[Port].Value CS_Status := SV Port := TMList{SA, ProtVar}.Sync.PrimPort TMList{SA, ProtVar}.Sync.SecPort := Port TMList{SA, ProtVar}.Sync.SecFwup := FALSE	IDLE / AW-CNF
37	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && RedSync{SA, ProtVar}[Port] && !TMList{SA, ProtVar}.Sync.PrimRed => ignore	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
38	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && (SA in TMList) && (FramID in TMList[SA]) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && RedSync{SA, ProtVar}[Port] && TMList{SA, ProtVar}.Sync.PrimRed && TMList{SA, ProtVar}.Sync.PrimPort == Port => ignore	IDLE / AW-CNF
39	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && RedSync{SA, ProtVar}[Port] && TMList{SA, ProtVar}.Sync.PrimRed && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort != NIL => ignore	IDLE / AW-CNF
40	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && !RedSync{SA, ProtVar}[Port] => ignore	IDLE / AW-CNF
41	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq < TMList{SA, ProtVar}.Sync.Seq => ignore	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
42	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && (!{SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value != NIL && TMList.ActCntEntry < TMList.MaxCntEntry => CS_list.S_SDU := (mac_service_data_unit / LT, rt_id) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.RT_ID := mac_service_data_unit.rt_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list.S_SDU.T2 := TCC.cv CS_Status := NTM TMList{SA, ProtVar}.Sync.Seq := mac_service_data_unit.Seq TMList{SA, ProtVar}.Sync.PrimRed := RedSync{SA, ProtVar}[Port] TMList{SA, ProtVar}.Sync.PrimPort := Port TMList{SA, ProtVar}.Sync.PrimFwup := FALSE TMList{SA, ProtVar}.Sync.SecPort := NIL TMList{SA, ProtVar}.Sync.SecFwup := FALSE TMListActCntEntry := TMListActCntEntry + 1 SYN_Sync.ind (Port, DA, SA, CS_list, CS_status) SYN_SyncFW.ind (-(Port), DA, SA, CS_list, CS_status)	IDLE / AW-CNF
43	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && (!{SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value == NIL && TMList.ActCntEntry < TMList.MaxCntEntry => CS_list.S_SDU := (mac_service_data_unit / LT, rt_id) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.RT_ID := mac_service_data_unit.rt_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list.S_SDU.T2 := TCC.cv CS_Status := NTM TMList{SA, ProtVar}.Sync.Seq := mac_service_data_unit.Seq TMList{SA, ProtVar}.Sync.PrimRed := RedSync{SA, ProtVar}[Port] TMList{SA, ProtVar}.Sync.PrimPort := Port TMList{SA, ProtVar}.Sync.PrimFwup := TRUE TMList{SA, ProtVar}.Sync.SecPort := NIL TMList{SA, ProtVar}.Sync.SecFwup := FALSE TMListActCntEntry := TMListActCntEntry + 1 SYN_Sync.ind (Port, DA, SA, CS_list, CS_status)	IDLE / AW-CNF
44	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == SYNC && (!{SA, ProtVar} in TMList) && TMList.ActCnTM >= TMList.MaxCnTM => ignore	IDLE / AW-CNF
45	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && (!{SA, ProtVar} in TMList) => ignore	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
46	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq <> TMList{SA, ProtVar}.Sync.Seq => ignore	IDLE / AW-CNF
47	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort != Port => ignore	IDLE / AW-CNF
48	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort == Port && TMList{SA, ProtVar}.Sync.PrimFwup == TRUE => ignore	IDLE / AW-CNF
49	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && ({SA, ProtVar} in TMList) && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort == Port && TMList{SA, ProtVar}.Sync.SecFwup == TRUE => ignore	IDLE / AW-CNF
50	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && ({SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value != NIL && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort == Port && !TMList{SA, ProtVar}.Sync.SecFwup => if (FU) FU_list.S_SDU.Delay := mac_service_data_unit.Delay + TMList{SA, ProtVar}.Delay[Port].Value FU_list.S_SDU := (mac_service_data_unit / LT, rt_id) FU_list.Prio := user_priority FU_list.VLAN_Tag := vlan_identifier FU_list.RT_ID := mac_service_data_unit.rt_id FU_Status := SV Port := TMList{SA, ProtVar}.Sync.PrimPort TMList{SA, ProtVar}.Sync.SecFwup := TRUE SYN_Fllw.ind (Port, DA, SA, FU_list, FU_status) SYN_FllwFW.ind (Port, DA, SA, FU_list, FU_status)	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
51	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && (SA in TMList) && (FrameID in TMList) && TMList{SA, ProtVar}.Delay[Port].Value == NIL && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort != Port && TMList{SA, ProtVar}.Sync.SecPort == Port => if (FU) FU_list.S_SDU.Delay := mac_service_data_unit.Delay FU_list.S_SDU := (mac_service_data_unit / LT, rt_id) FU_list.Prio := user_priority FU_list.VLAN_Tag := vlan_identifier FU_list.RT_ID := mac_service_data_unit.rt_id FU_Status := NDY TMList{SA, ProtVar}.Sync.SecFwup := TRUE SYN_Fllw.ind (Port, DA, SA, FU_list, FU_status)	IDLE / AW-CNF
52	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && (SA in TMList) && (FrameID in TMList) && TMList{SA, ProtVar}.Delay[Port].Value == NIL && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort == Port && !TMList{SA, ProtVar}.Sync.PrimFwup => if (FU) FU_list.S_SDU.Delay := mac_service_data_unit.Delay FU_list.S_SDU := (mac_service_data_unit / LT, rt_id) FU_list.Prio := user_priority FU_list.VLAN_Tag := vlan_identifier FU_list.RT_ID := mac_service_data_unit.rt_id FU_Status := NDY TMList{SA, ProtVar}.Sync.PrimFwup := TRUE SYN_Fllw.ind (Port, DA, SA, FU_list, FU_status)	IDLE / AW-CNF
53	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == FWUP && ({SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Delay[Port].Value != NIL && mac_service_data_unit.Seq == TMList{SA, ProtVar}.Sync.Seq && TMList{SA, ProtVar}.Sync.PrimPort == Port && !TMList{SA, ProtVar}.Sync.PrimFwup => if (FU) FU_list.S_SDU.Delay := mac_service_data_unit.Delay + TMList{SA, ProtVar}.Delay[Port].Value FU_list.S_SDU := (mac_service_data_unit / LT, rt_id) FU_list.Prio := user_priority FU_list.VLAN_Tag := vlan_identifier FU_list.RT_ID := mac_service_data_unit.rt_id FU_Status := SV TMList{SA, ProtVar}.Sync.PrimFwup := TRUE SYN_FllwFW.ind (-(Port), DA, SA, FU_list, FU_status) SYN_Fllw.ind (Port, DA, SA, FU_list, FU_status)	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
54	IDLE	SYN_Sync.req (Port, DA, SA, CS_list, CS_status) /({SA, ProtVar} in TMList) && (TMList{SA, ProtVar}.Sync.Seq < CS_list.S_SDU.Seq TMList{SA, ProtVar}.Sync.Seq == NIL) => CS_status := SV TMList{SA, ProtVar}.Sync.Seq := CS_list.S_SDU.Seq TMList{SA, ProtVar}.Sync.PrimRed := FALSE TMList{SA, ProtVar}.Sync.PrimPort := 0 TMList{SA, ProtVar}.Sync.PrimFwup := FALSE TMList{SA, ProtVar}.Sync.SecPort := NIL TMList{SA, ProtVar}.Sync.SecFwup := FALSE SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) SYN_Sync.cnf (Port, DA, SA, CS_list, CS_status)	IDLE / AW-CNF
55	IDLE	SYN_Sync.req (Port, DA, SA, CS_list, CS_status) /({SA, ProtVar} in TMList) && TMList{SA, ProtVar}.Sync.Seq >= CS_list.S_SDU.Seq => CS_status := NVS SYN_Sync.cnf (Port, DA, SA, CS_list, CS_status)	IDLE / AW-CNF
56	IDLE	SYN_Sync.req (Port, DA, SA, CS_list, CS_status) /(!{SA, ProtVar} in TMList) && TMList.ActCntEntry < TMList.MaxCntEntry => CS_status := SV TMList{SA, ProtVar}.Sync.Seq := CS_list.S_SDU.Seq TMList{SA, ProtVar}.Sync.PrimRed := FALSE TMList{SA, ProtVar}.Sync.PrimPort := Port TMList{SA, ProtVar}.Sync.PrimFwup := FALSE TMList{SA, ProtVar}.Sync.SecPort := NIL TMList{SA, ProtVar}.Sync.SecFwup := FALSE TMListActCntEntry := TMListActCntEntry + 1 SYN_SyncFW.ind (Port, DA, SA, CS_list, CS_status) SYN_Sync.cnf (Port, DA, SA, CS_list, CS_status)	IDLE / AW-CNF
57	IDLE	SYN_Sync.req (Port, DA, SA, CS_list, CS_status) /(!{SA, ProtVar} in TMList) && TMList.ActCntEntry >= TMList.MaxCntEntry => CS_status := NoTMLEntry SYN_Sync.cnf (Port, DA, SA, CS_list, CS_status)	IDLE / AW-CNF
58	IDLE	SYN_SyncFW.req (Port, DA, SA, CS_list, CS_status) => if (!FU) CS_list.S_SDU.T1 := TCC.cv CS_list.S_SDU.Delay := CS_list.S_SDU.Delay + CS_list.S_SDU.T1 – CS_list.S_SDU.T2 mac_service_data_unit := (SYNC-REQ-PDU) RT, SYNC, CS_list.S_SDU DA := Sync_MC user_priority, access_priority := CS_list.user_priority calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL FW_SYNC := TRUE M_UNITDATA.req (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-FCNF

#	Current State	Event /Condition =>Action	Next State
59	AW-FCNF	M_UNITDATA.cnf /FW_SYNC => if (FU) CS_list.S_SDU.T1 := TCC.cv CS_list.S_SDU := (mac_service_data_unit / LT, rt_id) CS_list.Prio := user_priority CS_list.VLAN_Tag := vlan_identifier CS_list.RT_ID := mac_service_data_unit.rt_id CS_list.FOLLOW_UP := mac_service_data_unit.Follow_Up CS_list.S_SDU.Delay += TMList[SA]Delay[Port] CS_list.S_SDU.T2 :=TCC.cv CS_status := SV FW_SYNC := FALSE SYN_SyncFW.cnf (Port, DA, SA, CS_list, CS_status)	CHK-NEW
60	IDLE	SYN_FilwFW.req (Port, DA, SA, FU_list, FU_status) => mac_service_data_unit := (FWUP-REQ-PDU) RT, FWUP, FU_list.S_SDU DA := FollowUp_MC user_priority, access_priority := FU_list.user_priority calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL FW_FWUP := TRUE M_UNITDATA.req (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-FCNF
61	AW-FCNF	M_UNITDATA.cnf /FW_FWUP => FU_list.S_SDU := (mac_service_data_unit / LT, rt_id) FU_list.Prio := user_priority FU_list.VLAN_Tag := vlan_identifier FU_list.FRAME_ID := frame_id FU_status := SV FW_FWUP := FALSE SYN_FilwFW.cnf (Port, DA SA, FU_list, FU_status)	CHK-NEW
62	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == DYRS => DS_list.S_SDU := (mac_service_data_unit / RT, DYRS) DS_list.Prio := user_priority DS_list.VLAN_Tag := vlan_identifier DS_Status := SV SYN_Delay_Rsp.ind (Port, DA, SA, DS_list, DS_status)	IDLE / AW-CNF
63	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == DYRQ => DQ_list.S_SDU := (mac_service_data_unit / RT, DYRQ) DQ_list.Prio := user_priority DQ_list.VLAN_Tag := vlan_identifier DQ_list.S_SDU.T4 :=TCC.cv DQ_Status := SV SYN_Delay_Req.ind (Port, DA, SA, DQ_list, DQ_status)	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
64	IDLE	SYN_Delay_Req.req (Port, DA, SA, DQ_list, DQ_status) => mac_service_data_unit := (DELAY-REQ-PDU) RT, DYRQ, DQ_list SA := Local_MAC DA := DelayReq_MC user_priority, access_priority := DQList.user_priority SyncSeq := DQ_list.SyncSeq calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL DYRQ := TRUE M_UNITDATA.req (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
65	IDLE	SYN_Delay_Rsp.req (Port, DA, SA, DS_list, DS_status) => mac_service_data_unit := (DELAY-RSP-PDU) RT, DYRS, DS_list.S_SDU SA := Local_MAC DA := DelayRsp_MC user_priority, access_priority := DS_list.user_priority calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL DYRS := TRUE M_UNITDATA.req (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
66	AW-CNF	M_UNITDATA.cnf /DYRS => DS_list.S_SDU := (mac_service_data_unit / LT, rt_id) DS_list.Prio := user_priority DS_list.VLAN_Tag := vlan_identifier DS_list.FRAME_ID := frame_id DS_list.DYRQ := mac_service_data_unit.Dyrs DS_status := SV DYRS := FALSE SYN_Delay_Rsp.cnf (Port, DA, SA, DS_list, DS_status)	CHK-NEW
67	AW-CNF	M_UNITDATA.cnf /DYRQ => DQ_list.S_SDU.T3 := TCC.cv DQ_list.S_SDU := (mac_service_data_unit / LT, rt_id) DQ_list.Prio := user_priority DQ_list.VLAN_Tag := vlan_identifier DQ_list.FRAME_ID := frame_id DQ_list.DYRQ := mac_service_data_unit.Dyrq DQ_status := SV DYRQ := FALSE SYN_Delay_Req.cnf (Port, DA, SA, DQ_list, DQ_status)	CHK-NEW
68	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == ADYRQ => ADQ_list.S_SDU := (mac_service_data_unit / LT, rt_id) ADQ_list.Prio := user_priority ADQ_list.VLAN_Tag := vlan_identifier ADQ_list.S_SDU.T2 :=TCC.cv ADQ_Status := SV SYN_Adelay_Req.ind (Port, DA, SA, ADQ_list, ADQ_status)	IDLE / AW-CNF

#	Current State	Event /Condition =>Action	Next State
69	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == ADYRS => ADS_list.S_SDU := (mac_service_data_unit / LT, rt_id) ADS_list.Prio := user_priority ADS_list.VLAN_Tag := vlan_identifier ADS_list.S_SDU.T4 :=TCC.cv ADS_Status := SV SYN_Adelay_Rsp.ind (Port, DA, SA, ADS_list, ADS_status)	IDLE / AW-CNF
70	IDLE / AW-CNF	M_UNITDATA.ind (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, frame_check_sequence, canonical_format_indicator, vlan_identifier) /mac_service_data_unit.LT == RT && mac_service_data_unit.rt_id == ADYFU => AFU_list.S_SDU := (mac_service_data_unit / LT, rt_id) AFU_list.Prio := user_priority AFU_list.VLAN_Tag := vlan_identifier AFU_Status := SV SYN_Adelay_Fllw_Rsp.ind (Port, DA, SA, AFU_list, AFU_status)	IDLE / AW-CNF
71	IDLE	SYN_Adelay_Req.req (Port, DA, SA, ADQ_list, ADQ_status) => mac_service_data_unit := (ADELAY-REQ-PDU) RT, ADYRQ, ADQ_list DA := AdelayReq_MC user_priority, access_priority := ADQList.user_priority SyncSeq := ADQ_list.SyncSeq calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL ADYRQ := TRUE M_UNITDATA.req (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
72	IDLE	SYN_Adelay_Rsp.req (Port, DA, SA, ADS_list, ADS_status) => if !(FU) { ADS_list.S_SDU.T3 := TCC.cv ADS_list.S_SDU.Fwup := FALSE } else ADS_list.S_SDU.Fwup := TRUE mac_service_data_unit := (ADELAY-RSP-PDU) RT, ADYRS, ADS_list.S_SDU DA := AdelayRsp_MC user_priority, access_priority := ADS_list.user_priority calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL ADYRS := TRUE M_UNITDATA.req (Port, frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag)	AW-CNF
73	AW-CNF	M_UNITDATA.cnf /ADYRQ => ADQ_list.S_SDU.T1 := TCC.cv ADQ_list.S_SDU := (mac_service_data_unit / LT, rt_id) ADQ_list.Prio := user_priority ADQ_list.VLAN_Tag := vlan_identifier ADQ_list.FRAME_ID := frame_id ADQ_list.DYRQ := mac_service_data_unit.Dyrq ADQ_status := SV DYRQ := FALSE SYN_Adelay_Req.cnf (Port, DA, SA, ADQ_list, ADQ_status)	CHK-NEW

#	Current State	Event /Condition =>Action	Next State
74	AW-CNF	M_UNITDATA.cnf /ADYRS => ADQ_list.S_SDU.T1 := TCC.cv ADQ_list.S_SDU := (mac_service_data_unit / LT, rt_id) ADQ_list.Prio := user_priority ADQ_list.VLAN_Tag := vlan_identifier ADQ_list.FRAME_ID := frame_id ADQ_list.DYRQ := mac_service_data_unit.Dyrq ADQ_status := SV DYRQ := FALSE SYN_Adelay_Rsp.cnf (Port, DA, SA, ADQ_list, ADS_status)	CHK-NEW
75	AW-CNF	M_UNITDATA.cnf /ADYRS => if (FU) { mac_service_data_unit := (ADELAY-FU-PDU) RT, ADYFU, AFU_list.S_SDU AFU_list.S_SDU.T3 := TCC.cv DA := AdelyFillw_MC user_priority, access_priority := AFU_list.user_priority calculate frame_check_sequence canonical_format_indicator := 1 vlan_classification := NIL ADYFU:= TRUE } ADS_list.S_SDU := (mac_service_data_unit / LT, rt_id) ADS_list.Prio := user_priority ADS_list.VLAN_Tag := vlan_identifier ADS_list.FRAME_ID := frame_id ADS_list.DYRS := mac_service_data_unit.Adysr ADS_status := SV ADYRS := FALSE if (FU) M_UNITDATA.req (Port; frame_type, mac_action, DA, SA, mac_service_data_unit, user_priority, access_priority, frame_check_sequence, canonical_format_indicator, vlan_classification, include_tag) SYN_Delay_Rsp.cnf (Port, DA, SA, ADS_list, ADS_status)	CHK-NEW
76	AW-CNF	M_UNITDATA.cnf /ADYFU => ADYFU := FALSE	CHK-NEW

5.8.6.4 Functions

All functions of the MMAC are summarized in Table 400.

Table 400 — MMAC function table

Function name	Operations
SETUP_NREQ	N_List.Num_entry— mac_service_data_unit := N_List.First_Entry.LT, N_SDU source_address := TS destination_address := N_List.First_Entry.DA user_priority := N_List.First_Entry.Prio access_priority := N_List.First_Entry.Prio canonical_format_indicator := 1 vlan_classification := N_List.First_Entry.VLAN_Tag calculate frame_check_sequence Service := N N_List.Remove()

Function name	Operations
SETUP_AREQ	Ac_List.Num_entry— mac_service_data_unit := RT, Ac_List.First_Entry.CREP.rt_id, Ac_List.First_Entry.Ac_SDU source_address := TS destination_address := Ac_List.First_Entry.DA user_priority := Ac_List.First_Entry.Prio access_priority := Ac_List.First_Entry.Prio canonical_format_indicator := 1 vlan_classification := NIL calculate frame_check_sequence Service := Ac Ac_List.Remove()
SETUP_CREQ	Cy_List.Num_entry— mac_service_data_unit := RT, Cy_List.First_Entry.CREP.rt_id, Cy_List.First_Entry.C_SDU, Cy_List.First_Entry.APDU_Status source_address := TS destination_address := Cy_List.First_Entry.DA user_priority := Cy_List.First_Entry.Prio access_priority := Cy_List.First_Entry.Prio canonical_format_indicator := 1 vlan_classification := NIL calculate frame_check_sequence Service := Cy Cy_List.Remove()
RESET_ALL_LIST	while (N_List.Num_entry ≠ 0) SETUP_CON with Service := N while (Ac_List.Num_entry ≠ 0) SETUP_CON with Service := A while (Cy_List.Num_entry ≠ 0) SETUP_CON with Service := C
SETUP_CON	with Service as List-Prefix ..._List.Num_entry— C_List.Insert() C_List.Last_Entry.CREP := ..._List.First_Entry.CREP C_List.Last_Entry.APDU_Status := ..._List.First_Entry.APDU_Status C_List.Last_Entry.DA := ..._List.First_Entry.DA C_List.Last_Entry.Function := Service C_List.Num_entry++ ..._List.Remove()
SETUP_IND_N	FN_List.Ibuffer.Num_entry— I_List.Insert() I_List.Last_Entry.DA := destination address I_List.Last_Entry.SA := source address I_List.Last_Entry.Function := N I_List.Last_Entry.DLSDU := mac_service_data_unit I_List.Num_entry++ FN_List.Ibuffer.Remove() Put in NFW List(Port)
SETUP_IND_A	FA_List.Ibuffer.Num_entry— I_List.Insert() I_List.Last_Entry.DA := destination address I_List.Last_Entry.SA := source address I_List.Last_Entry.Function := Cy I_List.Last_Entry.Prio := user_priority I_List.Last_Entry.VLAN_Tag := vlan_identifier I_List.Last_Entry.DLSDU := mac_service_data_unit \ LT I_List.Num_entry++ FA_List.Ibuffer.Remove() Put in AFW List(Port)
SETUP_IND_C	FC_List.Ibuffer.Num_entry— I_List.Insert() I_List.Last_Entry.DA := destination address I_List.Last_Entry.SA := source address I_List.Last_Entry.Function := Ac I_List.Last_Entry.Prio := user_priority I_List.Last_Entry.VLAN_Tag := vlan_identifier I_List.Last_Entry.DLSDU := mac_service_data_unit \ LT I_List.Num_entry++ FC_List.Ibuffer.Remove() Put in CFW List(Port)
I_BUF_N	FN_List.Ibuffer.First_Entry ≠ NIL

Function name	Operations
I_BUF_A	FA_List. Ibuffer.First_Entry ≠ NIL
I_BUF_C	FC_List. Ibuffer.First_Entry ≠ NIL

5.9 Parameters for an IO Device

Table 401 contains limitations for AL timing parameters of an IO device.

Table 401 — Bus parameter/reaction times for an IO device

Parameter	Meaning	Value
Min_IODevice_Interval (ms)	This performance parameter is an IO device property and part of the device description.	≤ 8
Remote_Application_Timeout (s)	The parameter Remote_Application_Timeout represents the deadline of a RPC call. Each IO device implementation has to ensure that the actual time for the response reaches the smallest value that is possible.	≤ 300

Each IO device implementation has to ensure that the Min_IODevice_Interval and Remote_Application_Timeout reach the smallest possible value.

5.10 DCP protocol

5.10.1 FAL syntax description

5.10.1.1 DLPDU abstract syntax reference

Table 121 gives an outline of the abstract syntax of the DL PDU.

5.10.1.2 APDU abstract syntax

Table 402 defines the abstract syntax of the PROFINET DCP PDUs referred to as APDUs. The defined order of octets shall be used to convey the APDUs.

Table 402 — APDU syntax

APDU name	APDU structure
DLSDU	DCP-UC-PDU ^ DCP-MC-PDU ^ DCP-MC-RSP-PDU
DCP-UC-PDU	[VLAN] ^{a)} , LTField(0x8892), FrameID(=0xFEFD), DCP-Get-ReqPDU ^ DCP-Set-ReqPDU ^ DCP-Get-ResPDU ^ DCP-Set-ResPDU a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept DLSDU's with or without VLAN fields.
DCP-MC-PDU	[VLAN] ^{a)} , LTField(0x8892), FrameID(=0xFEFE), DCP-Identify-ReqPDU a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept DLSDU's with or without VLAN fields.
DCP-MC-RSP-PDU	[VLAN] ^{a) b)} , LTField(0x8892), FrameID(=0xFEFF), DCP-Identify-ResPDU a) The field VLAN shall be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept DLSDU's with or without VLAN fields. b) The DLSDU shall be coded as Response in the Service-Type. Otherwise the DLSDU shall be discarded.

APDU name	APDU structure
DCP-Identify-ReqPDU	DCP-MC-Header ^{a) b) c)} , ListOfData a) The Data of the PDU shall be considered as filter. Only those devices are expected to send a response (DCP-Identify-ResPDU) which fulfill the filter request. b) If more than one filter is given inside the Data block the different filters are to be considered using a logical AND relation. c) The most significant filter shall be given first.
ListOfData	Data*
DCP-Identify-ResPDU	DCP-UC-Header, ListOfData
DCP-Get-ReqPDU	DCP-UC-Header, ListOfSelectors
DCP-Get-ResPDU	DCP-UC-Header, (Data ^ Response)*
DCP-Set-ReqPDU	DCP-UC-Header, ListOfData
DCP-Set-ResPDU	DCP-UC-Header, Response*
NOTE 1 According to ISO/IEC 8802-3 the DLSDU's have a minimum length of 52 octets. NOTE 2 For the ISO/IEC 8802-3 frames with VLAN tag is the minimum frame size increased to 68 octets in order to guaranty the minimum frame size of 64 octets after removing the VLAN tag by a bridge. NOTE 3 For unicast PDUs a VLAN prio of 0 shall be used. NOTE 4 For multicast PDUs a VLAN prio of 0 shall be used.	

Table 403 defines structures for substitutions of elements of the APDU structures shown in Table 402.

Table 403 — Substitutions

Substitution name	Structure
DCP-MC-Header	Service-ID, Service-Type, Xid, ResponseDelay, DCPDataLength
DCP-UC-Header	Service-ID, ServiceType, Xid, Reserved (2 octets), DCPDataLength
Data	Type, DataBlockLength, Value
Response	Type ^{a)} , DataBlockLength, ResponseValue
ResponseValue	ResponseStatus ^{b)} , Result, [AdditionalUserData*]
Type	Option, Suboption
ListOfSelectors	Type*
Value	[Status ^{b)] ^{d)}, NameOfStation ^ Device-role ^ Device-ID ^ Device-options ^ MAC-address ^ IP-parameter ^ DHCP-parameter ^ LLDP-Parameter}
Device-role	Device-role-details, Reserved (0)
Device-options	Type*
IP-parameter	IP-address, Subnetmask, Default-router

Substitution name	Structure
DHCP-parameter	Option-Code ^{c)} , Option-Length Data*
a) The Type shall be set to Option "Control", suboption "Response". b) The value of Status or ResponseStatus has a different meaning depending of its use in a request or response PDU and its use in different values. c) The Option-Code shall be set to the same value as in the suboption of the selector containing the DHCP-parameter. d) The status shall be omitted in all DCP-MC-PDUs	

5.10.1.3 Coding section related to DCP-MC-PDUs

5.10.1.3.1 Coding of the DLPDU field Destination Address

This field shall be coded as data type Octetstring of length 6 and shall contain the value

01-0E-CF-00-00-00.

5.10.1.4 Coding section related to DCP-MC-Header and the DCP-UC-Header

5.10.1.4.1 Coding of the field Service-ID

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 404. They shall be set in the appropriate PDU.

Table 404 — Service-ID

Value (decimal)	Meaning
0 – 2	reserved
3	Get
4	Set
5	Identify
6 – 255	reserved

5.10.1.4.2 Coding of the field Service-Type

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 405. They shall be set in the appropriate PDU

Table 405 — Service-Type

Bit	Usage	Value (binary)	Meaning
Bit 0 – 1	Request/ Response	00	Request
		01	Response
		10	Reserved
		11	Reserved
Bit 2	Request	0	Reserved
Bit 2	Response	0	Success
		1	Request not supported
Bit 3 – 7	Request/ Response	00000	reserved

5.10.1.4.3 Coding of the field xid

This field shall be coded as data type Unsigned32 and shall contain a Transaction ID, a random number chosen by the client, used by the client and server to associate messages and responses between a client and a server.

5.10.1.4.4 Coding of the field DCPDataLength

This field shall be coded as data type Unsigned16 and shall contain the total length of the data following the DCP-UC-Header or DCP-MC-Header in octets. The maximum length of DCP Data is 1432 bytes.

5.10.1.4.5 Coding of the field ResponseDelay

This field shall be coded as data type Unsigned16 and shall contain a ResponseDelay factor. The ResponseDelay is set by the client to spread responses to multicast request. This mechanism prevents the client for too many responses at the same time.

The last two bytes of the Ethernet address are used as value for the ResponseDelay factor ($K_{DCPdelay}$). Octet 6 of the MAC address represents the low-order Byte of the word Octet 5 of the MAC address the high-order Byte. Together with a time base ($t_{DCP-timebase}$) the delay factor delivers a delay time.

The time base for the calculation of the delay time ist 10 msec. The delay factor is subject to a convolution modulo F_{DCP} (= ResponseDelay). The calculation of the delay time T_{delay} is carried out according to the following equation.

$$T_{delay} = t_{DCP-timebase} \times (K_{DCPdelay} \bmod F_{DCP}) \text{ [sec]} \quad (6)$$

The minimal delay time ($K_{DCPdelay} = 0$) is carried out according to the following equation.

$$T_{delay \min} = 10 \times 10^{-3} \times (0 \bmod F_{DCP}) = 0,0000 \text{ sec} \quad (7)$$

The maximum delay time ($K_{DCPdelay} = 2^{16-1}$) is carried out according to the following equation.

$$T_{delay \max} = 10 \times 10^{-3} \times F_{DCP} \quad (8)$$

5.10.1.5 Coding section of DCP fields

5.10.1.5.1 Coding of the field Option

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 406. They shall be set in the appropriate PDU

Table 406 — Option

Value (hexa decimal)	Meaning
0	reserved
1	IP options
2	DeviceProperties options
3	DHCP options
4	LLDP options
5	Control
6 – 127	reserved
128 – 254	manufacturer specific options
255	AllSelector

5.10.1.5.2 Coding of the field DataBlockLength

This field shall be coded as data type Unsigned16 and shall contain the length of the value of the Suboption. It shall be set in the appropriate PDU.

5.10.1.5.3 Coding of the field Value

This field shall be coded as data type OctetString and shall contain the values of the Suboptions. If the suboption has an odd length a (padding byte) fillbyte is added at the end. The (padding byte) fillbyte is not part of the DataBlockLength. The (padding byte) has the value 0 (decimal).

5.10.1.5.4 Coding of the field Status

This field shall be coded as data type Unsigned16 and shall be set to 0. If there is another meaning of this field in combination with an option or suboption it is stated there.

For a singlecast request this field shall be coded as data type Unsigned16 and shall contain the values of Table 407.

Table 407 — Status (Request)

Bit	Value (binary)	Meaning
Bit 0	0	Do not save Data-value permanent
	1	Save Data-value permanent
Bit 1 – 15		reserved

5.10.1.5.5 Coding of the field Result

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 408. They shall be set in the appropriate PDU

Table 408 — Result

Value (decimal)	Meaning
0	no error
1	option not supported
2	suboption not supported
3	suboption not set
4	resource error
5	SET not possible
6 – 255	reserved
NOTE Result 4 (resource error) is a temporary error. The application may repeat the command several times.	

5.10.1.6 Coding of the Control Options**5.10.1.6.1 Coding of the field suboption**

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 409. They shall be set in the appropriate PDU.

Table 409 — suboptions for control option

Value (decimal)	Meaning
0	reserved
1	StartTransaction
2	EndTransaction
3	Signal
4	Response
5	ResetToFactorySettings
6 – 255	reserved

5.10.1.6.1.1 Coding of the field StartTransaction

The suboption StartTransaction is coded with DataBlockLength zero. There are no additional data for this suboption.

5.10.1.6.1.2 Coding of the field EndTransaction

The suboption EndTransaction is coded with Length zero. There are no additional data for this suboption.

5.10.1.6.1.3 Coding of the field Signal

The suboption Signal is coded with length two bytes. The first byte shall be coded as data type Unsigned8 and shall contain the values as described in Table 410. They shall be set in the appropriate PDU.

Table 410 — suboption Signal

Value (decimal)	Meaning
0	reserved
1	FlashOnce

Value (decimal)	Meaning
2 – 255	reserved

FlashOnce means to flash the Ethernet LINK LED (or an alternative signalization for the Ethernet link status) with a duration of 3 sec with a frequency of 2 Hz (500 ms on, 500 ms off).

The second byte shall be coded as data type Unsigned8 and shall contain the value 0. It shall be set in the appropriate PDU.

5.10.1.6.1.4 Coding of the field Response

The suboption Response contains the result of the referenced option. The length of this option is 1 bytes but it is 3 bytes at least. It shall contain the values as described in Table 408. If there are additional user response data the DataBlockLength shall be adjusted according to the length of the data. If the length of the data leads to an odd address for the following option a “paddingbyte” with value 0 shall be added. This “paddingbyte” shall be unaccounted for the DataBlockLength.

This status field of this suboption shall be coded as data type Unsigned16 and shall contain the values as described in Table 411.

Table 411 — Status (suboption Response)

Value (decimal)	Meaning
Bit 0 – 7	Shall contain the value of the referenced suboption as described in Table 409 or Table 412 or Table 414 or Table 416
Bit 8 – 15	Shall contain the value of the referenced option as described in Table 406.

5.10.1.6.1.5 Coding of the field ResetToFactorySettings

The suboption ResetToFactorySettings is coded with length 0 bytes. The result of the suboption is to reset all device internal stored data to factory default settings (e.g. NameOfStation to Length 0).

The device may decline the request (e.g. wrong operating state). The suboption ResetToFactorySettings cannot be read back.

5.10.1.7 Coding of the IP options

5.10.1.7.1 Coding of the field suboption

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 114. They shall be set in the appropriate PDU.

Table 412 — suboptions for IP option

Value (decimal)	Meaning
0	reserved
1	MAC address
2	IP-parameter
3 – 255	reserved

5.10.1.7.1.1 Coding of the field MAC-address

This field shall be coded as data type OctetString[6]. The value of the field MAC-address shall be according to ISO/IEC8802-3 MAC address.

The least significant bit of the first Octet shall be zero. The address assignment rules of IANA applies but may be not checked by the receiver. It shall be set in the appropriate PDU.

5.10.1.7.1.2 Coding of the field IP-parameter**5.10.1.7.1.2.1 Coding of the field Status**

This field shall be coded as data type Unsigned16 and shall contain the values as described in Table 413.

Table 413 — ResponseStatus (suboption IP-parameter)

Bit	Meaning
Bit 0	IP parameter set
Bit 1	IP parameter received via DHCP
Bit 2 – 6	Reserved
Bit 7	IP parameter set (see Bit 0 or 1) but IP address currently not active because IP address conflict detected.
Bit 8 – 15	reserved
NOTE Bit 0 and bit 1 shall only be set exclusiv. Bit 7 may be set with any combination of bit 0 or bit 1.	

5.10.1.7.1.2.2 Coding of the field IP address

This field shall be coded as data type Unsigned32. The encoding of the fields shall be according to RFC791.

The value 0.0.0.0 shall be use to delete a currently set IP address.

5.10.1.7.1.2.3 Coding of the field Subnetmask

This field shall be coded as data type Unsigned32. The encoding of the fields shall be according to RFC791.

5.10.1.7.1.2.4 Coding of the field Default-router

This field shall be coded as data type Unsigned32. The encoding of the fields shall be according to RFC791.

If there is no configuration of the Default-router the value shall be set to the same value as the IP address.

5.10.1.8 Coding of the DeviceProperties options**5.10.1.8.1 Coding of the field suboption**

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 414. They shall be set in the appropriate PDU.

Table 414 — suboptions for DeviceProperties options

Value (decimal)	Meaning
0	reserved
1	Manufacturer specific
2	NameOfStation
3	Device-ID
4	Device-role
5	Device-options
6	Alias NameOfStation
7 – 255	reserved

5.10.1.8.1.1 Coding of the field NameOfStation**5.10.1.8.1.1.1 General**

The NameOfStation is an Octetstring without Null termination. With the NameOfStation the user can provide the station with a technologically marked name. In diagnosis a simple identification of the station is possible. The coding of the NameOfStation shall be conform to the DNS guidelines (see RFC 1035).

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

5.10.1.8.1.1.2 Using suboption NameOfStation in an Identify request

With the suboption NameOfStation in the Identify.Reg a filter for the station name of a PROFINET device can be accomplished. For a response to an Identify.Request with this value the following rules shall apply:

- name of the station with length != 0: 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 the station with length == 0: Only those devices shall answer, which have not received a station name yet.

The network representation of the NameOfStation in the Identify.Reg shall be according to RFC 1035 with the exception, that only lower case alphabetic characters in the range a through z are allowed. This network representation does not conflict with the DNS coding rules of name, because even if upper and lower case letters are allowed in domain names, no significance is attached to the case. That is, two names with the same spelling but different case are to be treated as if identical.

Restricting the network representation in the Identify.Reg to lower case letters allows fast and easy comparison of names in small devices (e.g. simple remote IO).

5.10.1.8.1.2 Coding of the field AliasNameOfStation

5.10.1.8.1.2.1 General

The AliasNameOfStation is coded in the same way as the NameOfStation (see 5.10.1.8.1.1).

5.10.1.8.1.2.2 Using suboption AliasNameOfStation in an Identify request

With the suboption AliasNameOfStation in the Identify.Reg a filter for additional station name of a PROFINET IO-Device can be accomplished. The request shall not be used in combination with a suboption NameOfStation in the same Identify request. For a response to an Identify.Request with this value the following rules shall apply:

- name of the station 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 AliasNameOfStation. 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 AliasNameOfStation

NOTE The AliasNameOfStation is an alternative name for a device. The alias name may be derived from topology or neighbourhood information. The alias name of station cannot be set through means of DCP.

5.10.1.8.1.3 Coding of the field Device-ID

The field shall be coded as data type array of Unsigned16 and shall contain the PROFINET vendor-ID and the PROFINET device ID in this order. It shall be set in the appropriate PDU. Details of the definition of the PROFINET device ID and vendor ID (see IEC 61158-5 and IEC 61158-6).

5.10.1.8.1.4 Coding of the field Device-role

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 415. They shall be set in the appropriate PDU.

Table 415 — suboption Device-role-details

Bit	Meaning
Bit 0	PROFINET IO-Device
Bit 1	PROFINET IO-Controller
Bit 2	PROFINET IO-Multidevice
Bit 3	PROFINET IO-Supervisor
Bit 4	Reserved

Bit	Meaning
Bit 5	Reserved
Bit 6	Reserved
Bit 7	Reserved

The bit shall be set to 1 if the device has the specific role. If none of the roles applies (e.g. a standard switch) none of the bits shall be set.

5.10.1.8.1.5 Coding of the field Device-options

This field shall contain the options and suboptions available on the server.

5.10.1.9 Coding of the DHCP option

5.10.1.9.1 Coding of the field suboption

5.10.1.9.1.1.1 General

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 416. They shall be set in the appropriate PDU.

Table 416 — suboption for DHCP

Value	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
62 – 80		reserved ^{a)}	
81	1+	FQDN, Fully Qualified Domain Name ^{d)}	
82 – 96		reserved ^{a)}	
97	Variable	UUID/GUID-based Client (=addressed Station) Identifier	
98 – 254		reserved ^{a)}	
255	1	Control DHCP for address resolution 0: Don't use DHCP 1: Don't use DHCP and reset all DHCP options 2: Use DHCP ^{b)} for address resolution	

a) The reserved options should be used in the same meaning as they are defined in the corresponding RFCs for DHCP.
b) This option should be the last in a list of DHCP options in a set request.
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.

The DHCP options dataset should only be used to set the DHCP parameters for a DHCP client.

5.10.1.9.1.2 Using the DHCP Option 61

The client Identifier is used by the device (DHCP client) to request IP parameters from the DHCP server. DHCP servers use the client identifier to find the corresponding IP parameters in their database for the requesting device. A device shall be able to use the client identifier from the data set in the following way:

- DHCP and MAC address 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:

Suboption = 61

Length = 1
Type = 1

- DHCP and 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 RFC2132):

Suboption = 61
Length = 1
Type = 0

This option enables the device to use the NameOfStation as client identifier. When the name of station of the device is changed the DHCP request uses automatically the new NameOfStation in the next DHCP request.

- DHCP with 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:

Suboption = 61
Length => 1
Type = 0
Identifier = 1 ... n Octet: ClientID

5.10.1.10 Coding of the LLDP Options

Up to now no separate addressable LLDP options are known.

All readable LLDP information is available via SNMP (see 3418) and an LLDP MIB (see IEEE P802.1AB).

5.10.1.11 Coding of the AllSelector Option

5.10.1.11.1 Coding of the field Suboption

This field shall be coded as data type Unsigned8 and shall contain the values as described in Table 417. They shall be set in the appropriate PDU.

Table 417 — Suboptions for AllSelector option

Value (decimal)	Meaning
0-254	reserved
255	AllSelector

The length of the suboption shall be set to 0.

5.10.1.12 Identify response content

All requested filters with their current values shall be part of the identify response. Additionally the identify response shall contain the options/suboptions as described in Table 418, even if they are not part of the identify request.

Table 418 — Suboptions for Identify response content

Option (decimal)	Suboption (decimal)	Meaning
1	2	IP-parameter ^{a)}
2	2	NameOfStation
2	3	Device ID
2	4	Device Role
2	5	Device options
a) If the device has no IP parameters set, the IP details have to be delivered with default values.		

Additional suboptions are allowed in the identify response (e.g. manufacturer specific), even if the suboption is not requested in the identify request.

5.10.2 FAL protocol state machines

5.10.2.1 Application relationship monitoring

5.10.2.1.1 Monitoring of the server response at the requestor

DCPUC_Timeout

This Timer, restarted when sending an unicast request on the requestor, will terminate waiting for an response.

DCPMC_Timeout

This Timer, restarted when sending an multicast request on the requestor, will terminate waiting for an response.

5.10.2.1.1.1 Intervals of the receiver

Send_MC_Response

This monitoring time specifies the smallest allowed period of time between multicast request and response.

Hold_requestor

This monitoring time specifies the smallest allowed period of time between a response and the release of the client. This is used to allow multiple services executed from a single source without disruption of other nodes.

5.10.3 Name Resolution Application Relationship Protocol Machines (ARPMs)

5.10.3.1 DCPUCS

5.10.3.1.1 Primitive definitions

5.10.3.1.1.1 Primitives exchanged between DCPUCS and NRPM

Table 419 shows the service primitives including their associated parameters issued by NRPM and received by DCPUCS.

Table 419 — Primitives issued by NRPM to DCPUCS

Primitive name	Source	Associated parameters	Functions
Activate.req	NRPM	CREP, VLAN-Prio, VLAN-ID, DCPUC_Timeout, DCPUC_Retry	
Close.req	NRPM	CREP	
DCP_Get.req()	NRPM	CREP, DA, ListOfSelectors	
DCP_Set.req()	NRPM	CREP, DA, ListOfData	

Table 420 shows the service primitives including their associated parameters issued by DCPUCS and received by the NRPM.

Table 420 — Primitives issued by DCPUCS to NRPM

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	DCPUCS	CREP	
Activate.cnf(-)	DCPUCS	CREP, ERRCLS, ERRCODE	
Close.cnf(+)	DCPUCS	CREP	
DCP_Get.cnf(+)	DCPUCS	CREP, ListOfData	
DCP_Get.cnf(-)	DCPUCS	CREP, ERRCLS, ERRCODE	

Primitive name	Source	Associated parameters	Functions
DCP_Set.cnf(+)	DCPUCS	CREP, ListOfResponse	
DCP_Set.cnf(-)	DCPUCS	CREP, ERRCLS, ERRCODE	

5.10.3.1.1.2 Primitives exchanged between DCPUCS and LMPM

Table 421 shows the service primitives including their associated parameters issued by DCPUCS and received by LMPM.

Table 421 — Primitives issued by DCPUCS to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.req	DCPUCR	CREP DA SA Prio A_SDU	
LMPM_Schedule_add.req	DCPUCS	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	DCPUCR	CREP	

Table 422 shows the service primitives including their associated parameters issued by DCPUCS and received by the LMPM.

Table 422 — Primitives issued by LMPM to DCPUCS

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	CREP LMPM_status	
LMPM_A_Data.ind	LMPM	CREP DA SA Prio A_SDU	
LMPM_Time_Event.ind	LMPM	CREP	

5.10.3.1.1.3 Parameters of DCPUCS primitives

Table 423 shows all parameters used with primitives between the FSPM and the DCPUCS.

Table 423 — Parameters used with primitives exchanged between FSPM and DCPUCS

Parameter Name	Description
A_SDU	Acyclic realtime service data unit
CREP	Communication Reference unique for that Station (can be deduced from Received Data with RT-Ethertype)
DA	Destination Address
DCPUC_Retry	
DCPUC_Timeout	
ListOfData	
LMPM_status	IV = invalid Parameter, LS = no local Resource available, OK = no error
Phase	Offset of the start of a send interval between 1 and ReductionRatio
Prio	Priority according to IEEE 802.1Q
ReductionRatio	This parameter multiplied with the SendClockTime is the send interval.
SA	Source Address
Sequence	Sequence is relative number within a phase (= 0 means no numbering)

Parameter Name	Description
VLAN-ID	
VLAN-Prio	

5.10.3.1.2 State machine description

The CLOSED state indicates that an initialisation is needed. On activate service will bring the machine in the OPEN state waiting for an service requests and Time events. After issuing the transmission of data the WACK state is used to wait for a confirmation. Receiving the confirmation will bring back the machine to the OPEN state. An error or a Close service request is needed to re-enter the CLOSED state.

Local variables of the DCPUCS

SXID

This local variable contains the Transaction ID XID used to identify services uniquely.

Tim_Act

This local variable will be used to indicate the activation of the timer after the first time event.

Pending

This local variable holds the information whether a DCP_Set.req or DCP_Get.req event was received.

Retry

This local variable contains the retry counter.

5.10.3.1.3 DCPUCS state table

Table 424 contains the complete description of the DCPUCS state table.

Table 424 — DCPUCS state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	DCPUCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPUC_Timeout, DCPUC_Retry) => SXID := Randomnumber Store VLAN-Prio, VLAN-ID, DCPUC_Timeout, DCPUC_Retry, Tim_Act := TRUE LMPM_Schedule_add.req (CREP,ReductionRatio:=DCPUC_Timeout, Phase:=NIL,Sequence:=0) DCPUCS_Activate.cnf (+)(CREP)	OPEN
2	CLOSED	DCPUCS_Close.req(CREP) => DCPUCS_Close.cnf (+) (CREP)	CLOSED
3	CLOSED	DCP_Get.req(CREP, DA, ListOfSelectors) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Get.cnf(-) (CREP, ERRCLS, ERRCODE)	CLOSED
4	CLOSED	DCP_Set.req(CREP, DA, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Set.cnf(-) (CREP, ERRCLS, ERRCODE)	CLOSED
5	CLOSED	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	CLOSED
6	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	CLOSED

#	Current State	Event /Condition =>Action	Next State
7	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ignore	CLOSED
8	OPEN	DCPUCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPUC_Timeout, DCPUC_Retry) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCS_Activate.cnf (-) (CREP, ERRCLS, ERRCODE)	OPEN
9	OPEN	DCPUCS_Close.req(CREP) => DCPUCS_Close.cnf (+) (CREP)	CLOSED
10	OPEN	DCP_Get.req(CREP, DA, ListOfSelectors) => Pending:= GET DCP_Header.xid:= SXID A_SDU := DCP_UC_PDU Retry:= DCPUC_Retry Tim_Act := FALSE LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU)	WACK
11	OPEN	DCP_Set.req(CREP, DA, ListOfData) => Pending:= SET DCP_Header.xid:= SXID A_SDU := DCP_UC_PDU Retry:= DCPUC_Retry Tim_Act := FALSE LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU)	WACK
12	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	OPEN
13	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	OPEN
14	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ignore	OPEN
15	OPEN	LMPM_Time_Event.ind(CREP) => ignore	OPEN
16	WACK	DCPUCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPUC_Timeout, DCPUC_Retry) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCS_Activate.cnf (-) (CREP, ERRCLS, ERRCODE)	WACK
17	WACK	DCPUCS_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) DCPUCS_Close.cnf (+) (CREP)	CLOSED
18	WACK	DCP_Get.req(CREP, DA, ListOfSelectors) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Get.cnf(-) (CREP, ERRCLS, ERRCODE)	WACK

#	Current State	Event /Condition =>Action	Next State
19	WACK	DCP_Set.req(CREP, DA, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Set.cnf(-) (CREP, ERRCLS, ERRCODE)	WACK
20	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Get-ResPDU && Pending == GET && DCP-Header.xid == SXID => SXID:= (SXID+1) & 0xffffffff DCP_Get.cnf(+) (CREP,ListOfData)	OPEN
21	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Get-ResPDU && (Pending != GET DCP-Header.xid != SXID) => ignore	WACK
22	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Set-ResPDU && Pending == SET && DCP-Header.xid == SXID => SXID:= (SXID+1) & 0xffffffff DCP_Set.cnf(+) (CREP, ListOfResponse)	OPEN
23	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Set-ResPDU && (Pending != SET DCP-Header.xid != SXID) => ignore	WACK
24	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	WACK
25	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK && Pending == GET => ERRCLS := PROTOCOL ERRCODE := LMPM Tim_Act := FALSE LMPM_Schedule_remove.req(CREP) DCPGet.cnf(-) (CREP, ERRCLS, ERRCODE)	CLOSED
26	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK && Pending == SET => ERRCLS := PROTOCOL ERRCODE := LMPM Tim_Act := FALSE LMPM_Schedule_remove.req(CREP) DCP_Set.cnf(-) (CREP, ERRCLS, ERRCODE)	CLOSED
27	WACK	LMPM_Time_Event.ind(CREP) /!Tim_Act => Tim_Act:=TRUE	WACK
28	WACK	LMPM_Time_Event.ind(CREP) /Tim_Act && Retry != 0 => A_SDU:=from Stored A_SDU Retry:= Retry-1 LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU)	WACK

#	Current State	Event /Condition =>Action	Next State
29	WACK	LMPM_Time_Event.ind(CREP) /Tim_Act && Retry == 0 && Pending == GET => ERRCLS := PROTOCOL ERRCODE := TIMEOUT Tim_Act := FALSE DCP_Get.cnf(-) (CREP, ERRCLS, ERRCODE)	OPEN
30	WACK	LMPM_Time_Event.ind(CREP) /Tim_Act && Retry == 0 && Pending == SET => ERRCLS := PROTOCOL ERRCODE := TIMEOUT Tim_Act := FALSE DCP_Set.cnf(-) (CREP, ERRCLS, ERRCODE)	OPEN

5.10.3.2 DCPUCR

5.10.3.2.1 Primitive definitions

5.10.3.2.1.1 Primitives exchanged between DCPUCR and RMPM

Table 425 shows the service primitives including their associated parameters issued by RMPM and received by DCPUCR.

Table 425 — Primitives issued by RMPM to DCPUCR

Primitive name	Source	Associated parameters	Functions
DCP_Get.rsp	RMPM	CREP ListOfData	
DCP_Set.rsp	RMPM	CREP ListOfResponse	

Table 426 shows the service primitives including their associated parameters issued by DCPUCR and received by the RMPM.

Table 426 — Primitives issued by DCPUCR to RMPM

Primitive name	Source	Associated parameters	Functions
DCP_Get.ind	DCPUCR	CREP SA ListOfSelectors	
DCP_Set.ind	DCPUCR	CREP SA ListOfData	
DCPUCR_Close.cnf (+)	DCPUCR	CREP	
DCPUCR_Activate.cnf (+)	DCPUCR	CREP	
DCPUCR_Activate.cnf (-)	DCPUCR	CREP ERRCLS ERRCODE	

5.10.3.2.1.2 Primitives exchanged between DCPUCR and LMPM

Table 432 shows the service primitives including their associated parameters issued by DCPUCR and received by LMPM.

Table 427 — Primitives issued by DCPUCR to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.req	DCPUCR	CREP DA SA Prio A_SDU	
LMPM_Schedule_add.req	DCPUCR	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	DCPUCR	CREP	

Table 436 shows the service primitives including their associated parameters issued by DCPUCR and received by the LMPM.

Table 428 — Primitives issued by LMPM to DCPUCR

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	CREP LMPM_status	
LMPM_A_Data.ind	LMPM	CREP DA SA Prio A_SDU	
LMPM_Time_Event.ind	LMPM	CREP	

5.10.3.2.1.3 Parameters of DCPUCR primitives

Table 429 shows all parameters used with primitives between the FSPM and the DCPUCR.

Table 429 — Parameters used with primitives exchanged between FSPM and DCPUCR

Parameter Name	Description
A_SDU	Acyclic realtime service data unit
CREP	Communication Reference unique for that Station (can be deduced from Received Data with RT-Ethertype)
DA	Destination Address
Hold_requestor	
ListOfData	
ListOfResponses	
ListOfSelectors	
LMPM_status	IV = invalid Parameter, LS = no local Resource available, OK = no error
Phase	Offset of the start of a send interval between 1 and ReductionRatio
Prio	Priority according to IEEE 802.1Q
ReductionRatio	This parameter multiplied with the SendClockTime is the send interval.
SA	Source Address
Sequence	Sequence is relative number within a phase (= 0 means no numbering)

5.10.3.2.2 State machine description

The CLOSED state indicates that an initialisation is needed. An activate service will bring the machine in the OPEN state waiting for the first LMPM_A_Data service indication. After passing the indication to the user the WACK state is entered. An response service will activate an LMPM_A_Data request and bring the machine back to the OPEN state. A timeout will cause a state transition back to OPEN (combined with an Error indication). A Close service request is needed to reenter the CLOSED state.

Local variables of the DCPUCR

SAM

This local variable contains the last valid SA conveyed with a LMPM_A_Data indication with valid data.

SXID

This local variable contains the Transaction ID XID used to identify services uniquely.

Tim_Act

This local variable will be used to indicate the activation of the timer after the first time event.

5.10.3.2.3 DCPUCR state table

Table 430 contains the complete description of the DCPUCR state table. A LMPM_A_Data.ind primitive will be accepted if Type is DCP-UC-PDU.

Table 430 — DCPUCR state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	DCPUCR_Activate.req(CREP, Hold_requestor) /Tim_Act := TRUE => LMPM_Schedule_add.req (CREP,ReductionRatio:=Hold_requestor, Phase:=NIL,Sequence:=0) DCPUCR_Activate.cnf (+)(CREP)	OPEN
2	CLOSED	DCPUCR_Close.req(CREP) => DCPUCR_Close.cnf (+)(CREP)	CLOSED
3	CLOSED	DCP_Get.rsp(CREP, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	CLOSED
4	CLOSED	DCP_Set.rsp(CREP, ListOfResponse) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	CLOSED
5	CLOSED	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCPUC_PDU => ignore	CLOSED
6	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	CLOSED
7	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	CLOSED
8	OPEN	DCPUCR_Activate.req(CREP, Hold_requestor) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCR_Activate.cnf (-) (CREP, ERRCLS, ERRCODE)	OPEN
9	OPEN	DCPUCR_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) DCPUCR_Close.cnf (+)(CREP)	CLOSED
10	OPEN	DCP_Get.rsp(CREP, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	OPEN

#	Current State	Event /Condition =>Action	Next State
11	OPEN	DCP_Set.rsp(CREP, ListOfResponse) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	OPEN
12	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Get-ReqPDU && (SAM == NIL SAM == SA) => SAM:= SA SXID := DCP_Header.xid Timer Starten DCP_Get.ind (CREP,SA,ListOfSelectors)	WACK
13	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Set-ReqPDU && (SAM == NIL SAM == SA) => SAM:= SA SXID := DCP_Header.xid Timer Starten DCP_Set.ind (CREP,SA,ListOfData)	WACK
14	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Get-ReqPDU && (SAM != NIL && SAM != SA) => ignore	OPEN
15	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Set-ReqPDU && (SAM != NIL && SAM != SA) => ignore	OPEN
16	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	OPEN
17	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	OPEN
18	OPEN	LMPM_Time_Event.ind(CREP) /!Tim_Act => Tim_Act:=TRUE	OPEN
19	OPEN	LMPM_Time_Event.ind(CREP) /Tim_Act => Tim_Act:= FALSE SAM:= NIL	OPEN
20	WACK	DCPUCR_Activate.req(CREP, Hold_requestor) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPUCR_Activate.cnf (-) (CREP, ERRCLS, ERRCODE)	WACK
21	WACK	DCPUCR_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) DCPUCR_Close.cnf (+)(CREP)	CLOSED
22	WACK	DCP_Get.rsp(CREP, ListOfData) => DCP_Header..op:= GET DCP_Header.xid:= SXID DA := SAM A_SDU := DCP_UC_PDU Tim_Act:= FALSE LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU)	OPEN

#	Current State	Event /Condition =>Action	Next State
23	WACK	DCP_Set.rsp(CREP, ListOfResponse) => DCP_Header.op:= SET DCP_Header.xid:= SXID DA := SAM A_SDU := DCP_UC_PDU Tim_Act:= FALSE LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU)	OPEN
24	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	WACK
25	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	WACK
26	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCPUCR_Error.ind (CREP, ERRCLS, ERRCODE)	WACK

5.10.3.3 DCPMCS

5.10.3.3.1 Primitive definitions

5.10.3.3.1.1 Primitives exchanged between DCPMCS and NRMC

Table 431 shows the service primitives including their associated parameters issued by NRMC and received by DCPMCS.

Table 431 — Primitives issued by NRMC to DCPMCS

Primitive name	Source	Associated parameters	Functions
Activate.req	NRPM	CREP, VLAN-ID VLAN-Prio, DCPMC_Timeout,	
Close.req	NRPM	CREP	
DCP_Identify.req	NRMC	CREP, DA, ListOfFilter, ResponseDelay	

Table 432 shows the service primitives including their associated parameters issued by DCPMCS and received by the NRMC.

Table 432 — Primitives issued by DCPMCS to NRMC

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	DCPMCS	CREP	
Activate.cnf(-)	DCPMCS	CREP ERRCLS ERRCODE	
Close.cnf(+)	DCPMCS	CREP	
DCP_Identify.cnf(+)	DCPMCS	CREP	
DCP_Identify.cnf(-)	DCPMCS	CREP, ERRCLS, ERRCODE	
DCP_IdentifyQ.ind	DCPMCS	CREP, ListOfData	

5.10.3.3.1.2 Primitives exchanged between DCPMCS and LMPM

Table 433 shows the service primitives including their associated parameters issued by DCPMCS and received by LMPM.

Table 433 — Primitives issued by DCPMCS to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.req	DCPUCR	CREP DA SA Prio A_SDU	
LMPM_Schedule_add.req	DCPUCS	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	DCPUCR	CREP	

Table 434 shows the service primitives including their associated parameters issued by DCPMCS and received by the LMPM.

Table 434 — Primitives issued by LMPM to DCPMCS

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	CREP LMPM_status	
LMPM_A_Data.ind	LMPM	CREP DA SA Prio A_SDU	
LMPM_Time_Event.ind	LMPM	CREP	

5.10.3.3.1.3 Parameters of DCPMCS primitives

Table 435 shows all parameters used with primitives between the FSPM and the DCPMCS.

Table 435 — Parameters used with primitives exchanged between FSPM and DCPMCS

Parameter Name	Description
A_SDU	Acyclic realtime service data unit
CREP	Communication Reference unique for that Station (can be deduced from Received Data with RT-Ethertype)
DA	Destination Address
DCPMC_Timeout	
ListOfData	
ListOfFilters	
LMPM_status	IV = invalid Parameter, LS = no local Resource available, OK = no error
Phase	Offset of the start of a send interval between 1 and ReductionRatio
ReductionRatio	This parameter multiplied with the SendClockTime is the send interval.
ResponseDelay	
SA	Source Address
Sequence	Sequence is relative number within a phase (= 0 means no numbering)
VLAN-ID	
VLAN-Prio	

5.10.3.3.2 State machine description

The CLOSED state indicates that an initialisation is needed. Activate service will bring the machine in the OPEN state waiting for an identify request. After issuing the transmission of

the data the WACK state is used to wait for a response. Receiving the response issues an indication with the received A-Data. Receiving the timeout will bring back the machine to the OPEN state. An error or a Close service request is needed to reenter the CLOSED state.

Local variables of the DCPMCS

Retry

This local variable is loaded with DCPMC_Retry and decremented at every retransmission of a frame. A value of zero indicates the failure in that transaction. A new Activate service is needed to re-establish the communication between the peer providers.

SXID

This local variable contains the Transaction ID XID used to identify services uniquely.

Pending

This local variable holds the information if a DCP_Identify.req event was received.

Time_Count

This local variable contains the time value.

5.10.3.3.3 DCPMCS state table

Table 436 contains the complete description of the DCPMCS state machine. A LMPM_A_Data.ind primitive will be accepted if Type is DCP-MC-RSP-PDU.

Table 436 — DCPMCS state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	DCPMCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPMC_Timeout) => SXID := RandomNumber Store VLAN-Prio, VLAN-ID LMPM_Schedule_add.req (CREP,ReductionRatio:=DCPMC_Timeout, Phase:=NIL,Sequence:=0) DCPMCS_Activate.cnf (+)(CREP)	OPEN
2	CLOSED	DCPMCS_Close.req(CREP) => DCPMCS_Close.cnf (+)(CREP)	CLOSED
3	CLOSED	DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE)	CLOSED
4	CLOSED	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	CLOSED
5	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	CLOSED
6	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ignore	CLOSED
7	OPEN	DCPMCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPMC_Timeout) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCS_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	OPEN
8	OPEN	DCPMCS_Close.req(CREP) => DCPMCS_Close.cnf (+)(CREP)	CLOSED
9	OPEN	DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay) => Pending:= IDENT DCP_Header.xid:= SXID	WACK

#	Current State	Event /Condition =>Action	Next State
		A_SDU := DCP_MC_PDU Time_Count:=Response_Delay+1 LMPM_A_SDU.req(CREP,DA, SA, VLAN-Prio,A_SDU)	
10	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	OPEN
11	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	OPEN
12	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ignore	OPEN
13	OPEN	LMPM_Time_Event.ind(CREP) => ignore	OPEN
14	WACK	DCPMCS_Activate.req(CREP, VLAN-Prio, VLAN-ID, DCPMC_Timeout) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCS_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	WACK
15	WACK	DCPMCS_Close.req(CREP) => LMPM_Schedule_remove.req(CREP) DCPMCS_Close.cnf (+)(CREP)	CLOSED
16	WACK	DCP_Identify.req(CREP, DA, ListOfFilter, ResponseDelay) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE)	WACK
17	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_UC_PDU with DCP_Identify-ResPDU && Pending == IDENT && DCP-Header.xid == SXID => ListOfData = A_SDU DCP_IdentifyQ.ind(CREP, SA, ListOfData)	WACK
18	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /!(DCP_MC_PDU with DCP_Identify-ResPDU) (Pending != IDENT DCP-Header.xid != SXID) => ignore	WACK
19	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	WACK
20	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCP_Identify.cnf(-) (CREP,ERRCLS,ERRCODE)	CLOSED
21	WACK	LMPM_Time_Event.ind(CREP) /Time_Count > 0 => Time_Count --	WACK
22	WACK	LMPM_Time_Event.ind(CREP) /Time_Count == 0 => SXID:= SXID + 1 mod 0xffffffff DCP_Identify.cnf(+) (CREP)	OPEN

5.10.3.4 DCPMCR

5.10.3.4.1 Primitive definitions

5.10.3.4.1.1 Primitives exchanged between DCPMCR and RMPM

Table 437 shows the service primitives including their associated parameters issued by RMPM and received by DCPMCR.

Table 437 — Primitives issued by RMPM to DCPMCR

Primitive name	Source	Associated parameters	Functions
Activate.req	RMPM	CREP, VLAN-ID, VLAN-Prio, ListOfData, Send_MC_Response ^a	Data contains Filters
Close.req	RMPM	CREP	
DCP_IdentifyQ.req()	RMPM	CREP, DA ListOfData	DA contains the destination address of the sender of the identify request (response is sent as singlecast)

^a The value of the timeout shall be 10ms.

Table 438 shows the service primitives including their associated parameters issued by DCPMCR and received by the RMPM.

Table 438 — Primitives issued by DCPMCR to RMPM

Primitive name	Source	Associated parameters	Functions
Activate.cnf(+)	DCPMCR	CREP	
Activate.cnf(-)	DCPMCR	CREP, ERRCLS, ERRCODE	
Close.cnf(+)	DCPMCR	CREP	
Close.cnf(-)	DCPMCR	CREP, ERRCLS, ERRCODE	
DCP_Identify.ind	DCPMCR	CREP, SA ListOfFilters ResponseDelay	
Error.ind()	DCPMCR	CREP, ERRCLS, ERRCODE	

5.10.3.4.1.2 Primitives exchanged between DCPMCR and LMPM

Table 439 shows the service primitives including their associated parameters issued by DCPMCR and received by LMPM.

Table 439 — Primitives issued by DCPMCR to LMPM

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.req	DCPMCR	CREP DA SA Prio A_SDU	
LMPM_Schedule_add.req	DCPMCR	CREP ReductionRatio Phase Sequence	
LMPM_Schedule_remove.req	DCPMCR	CREP	

Table 440 shows the service primitives including their associated parameters issued by DCPMCR and received by the LMPM.

Table 440 — Primitives issued by LMPM to DCPMCR

Primitive name	Source	Associated parameters	Functions
LMPM_A_Data.cnf	LMPM	CREP LMPM_status	
LMPM_A_Data.ind	LMPM	CREP DA SA Prio A_SDU	
LMPM_Time_Event.ind	LMPM	CREP	

5.10.3.4.1.3 Parameters of DCPMCR primitives

Table 441 shows all parameters used with primitives between the FSPM and the DCPMCR.

Table 441 — Parameters used with primitives exchanged between FSPM and DCPMCR

Parameter Name	Description
A_SDU	Acyclic realtime service data unit
CREP	Communication Reference unique for that Station (can be deduced from Received Data with RT-Ethertype)
DA	Destination Address
ListOfData	
ListOfFilters	
LMPM_status	IV = invalid Parameter, LS = no local Resource available, OK = no error
Phase	Offset of the start of a send interval between 1 and ReductionRatio
Prio	Priority according to IEEE 802.1Q
ReductionRatio	This parameter multiplied with the SendClockTime is the send interval.
ResponseDelay	
SA	Source Address
Send_MC_Response	
Sequence	Sequence is relative number within a phase (= 0 means no numbering)
VLAN-ID	
VLAN-Prio	

5.10.3.4.2 State machine description

The CLOSED state indicates that an initialisation is needed. On open service will bring the machine in the OPEN state waiting for LMPM_A_Data service indication. After passing a Data indication to the user the WACK state is used to wait for an acknowledge from the user. Receiving the acknowledge will bring back the machine to the OPEN state. An error or a Close service request is needed to reenter the CLOSED state.

Local variables of the DCPMCR

SXID

This local variable contains the Transaction ID XID used to identify services uniquely.

Count

This local variable contains the counter value.

5.10.3.4.3 DCPMCR state table

Table 442 contains the complete description of the DCPMCR state machine. A LMPM_A_Data.ind primitive will be accepted if Type is DCP-MC-PDU.

Table 442 — DCPMCR state table

#	Current State	Event /Condition =>Action	Next State
1	CLOSED	DCPMCR_Activate.req (CREP, VLAN-ID, VLAN-Prio, ListOfFilters, Send_MC_Response) => store VLAN-ID, VLAN-Prio, ListOfFilters SXID := Randomnumber LMPM_Schedule_add.req (CREP, ReductionRatio:=Send_MC_Response, Phase:=NIL, Sequence:=0) DCPMCR_Activate.cnf (+)(CREP)	OPEN
2	CLOSED	DCPMCR_Close.req(CREP) => DCPMCR_Close.cnf (+)(CREP)	CLOSED
3	CLOSED	DCP_IdentifyQ.req(CREP, DA, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	CLOSED
4	CLOSED	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	CLOSED
5	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	CLOSED
6	CLOSED	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	CLOSED
7	OPEN	DCPMCR_Activate.req (CREP, VLAN-ID, VLAN-Prio, ListOfFilters, Send_MC_Response) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCR_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	OPEN
8	OPEN	DCPMCR_Close.req(CREP) => LMPM_Schedule_remove.req (CREP) DCPMCR_Close.cnf (+)(CREP)	CLOSED
9	OPEN	DCP_IdentifyQ.req(CREP, DA, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	OPEN
10	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /DCP_MC_PDU with DCP_Identify-ReqPDU && Successful comparison with ListOfFilters => Count :=(SA[4]*256+SA[5]) mod ResponseDelay	WTIM
11	OPEN	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) /!(DCP_MC_PDU with DCP_Identify-ReqPDU && Successful comparison with ListOfFilters) => ignore	OPEN
12	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	OPEN
13	OPEN	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	OPEN

#	Current State	Event /Condition =>Action	Next State
14	WTIM	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	WTIM
15	WTIM	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	WTIM
16	WTIM	DCP_IdentifyQ.req(CREP, DA, ListOfData) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	WTIM
17	WTIM	DCPMCR_Close.req(CREP) => LMPM_Schedule_remove.req (CREP) DCPMCR_Close.cnf (+)(CREP)	CLOSED
18	WTIM	DCPMCR_Activate.req (CREP, VLAN-ID, VLAN-Prio, ListOfFilters, Send_MC_Response) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCR_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	WTIM
19	WTIM	LMPM_Time_Event.ind(CREP) /Count != 0 => Count:= Count -1	WTIM
20	WTIM	LMPM_Time_Event.ind(CREP) /Count == 0 => DCP_Identify.ind(CREP, SA, ListOfFilter, ResponseDelay)	WACK
21	WACK	DCPMCR_Activate.req (CREP, VLAN-ID, VLAN-Prio, ListOfFilters, Send_MC_Response) => ERRCLS := CTXT ERRCODE := INVALID_STATE DCPMCR_Activate.cnf (-) (CREP,ERRCLS,ERRCODE)	WACK
22	WACK	DCPMCR_Close.req(CREP) => LMPM_Schedule_remove.req (CREP) DCPMCR_Close.cnf (+)(CREP)	CLOSED
23	WACK	DCP_IdentifyQ.req(CREP, DA, ListOfData) => DCP_Header..op:= IDENT DCP_Header.xid:= SXID A_SDU := List of Data LMPM_A_Data.req (CREP, DA, SA, Prio, A_SDU)	OPEN
24	WACK	LMPM_A_Data.ind (CREP, DA, SA, Prio, A_SDU) => ignore	WACK
25	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status == OK => ignore	WACK
26	WACK	LMPM_A_Data.cnf (CREP, LMPM_status) /LMPM_status != OK => ERRCLS := PROTOCOL ERRCODE := LMPM DCPMCR_Error.ind (CREP, ERRCLS, ERRCODE)	WACK

Annex A (informative)

Device Instances

The design and construction of physical automation devices is beyond the scope of this specification and therefore vendor specific. However, PROFINET IO provides a model for physical devices to host one or more IO controller or IO devices. These particular IO controller or IO devices are referred to as instances. The physical device is characterized by its device address which is a unique IP specification address assigned to a network interface. The further addressing is provided by means of the underlying connectionless DCE RPC protocol. These means are:

- Interface UUID
- Object UUID

The model is provided in Figure A.1. It covers the IO controller and IO device. The gray boxes in Figure A.1 are beyond the scope of this service definition.

NOTE The term “Interface” belongs to the DCE RPC specification. An interface represents a collection of functions or services that can be called by a remote user.

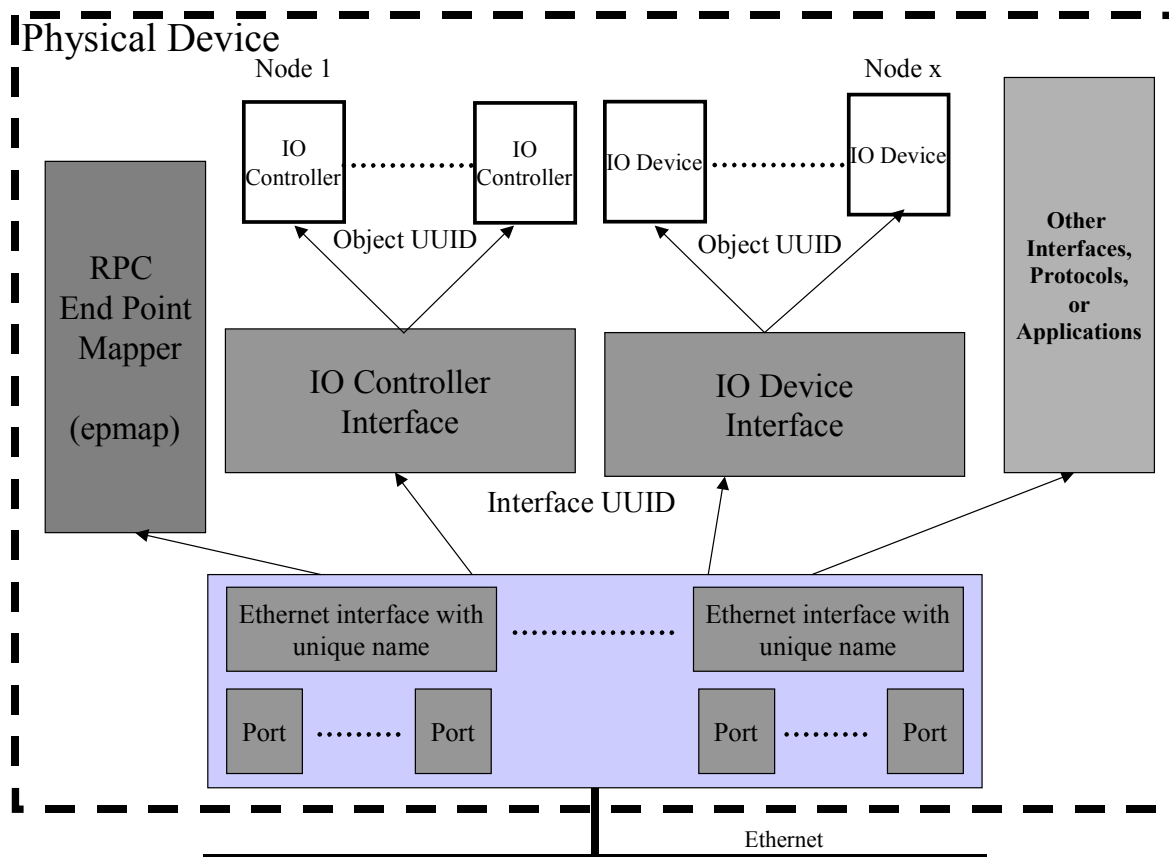


Figure A.1 — Instance model of PROFINET IO

The Interface UUID provides a means to address an IO supervisor, an IO controller, or an IO device without further configuration. The following Interface UUIDs are defined in 4.2.3.1.4.1:

- IO Device Interface UUID
- IO Controller Interface UUID
- IO Supervisor Interface UUID
- IO Parameter Server Interface UUID

Furthermore, an IO Remote Control Interface UUID is defined for each device type. The epmap interface UUID shall be supported by all PROFINET IO device classes.

The Object UUID provides a means to address the instance of the IO supervisor, IO controller, IO parameter server, or IO device. The Object UUID shall be built according the rule described in 4.2.3.1.4.1.

The connectionless DCE RPC protocol defines a local instance referred to as End Point Mapper (epmap). The End Point Mapper receives any incoming RPC request at TCP/UDP and transfers the request to the matching interface and object. Therefore, each server has to register its UUIDs before. This information includes:

- Interface with its UUID, Major Version, Minor Version
- Object UUID
- Binding Information such as protocol (UDP/TCP), server IP address, port, 64 octets annotation

NOTE The generic station description contains the information about the maximum number of instances a device supports.

BIBLIOGRAPHY

IEC 61131-1:1992, *Programmable controllers, Part 1: Programmable controllers — General information*

ISO/IEC TR P02-1:2001, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 1: Overview of local and network standards*

IEEE 8802.1AB/D11, *IEEE Draft Standards for Local and Metropolitan Networks: Station and Media Access Control Connectivity Discovery*, available at <<http://www.ieee.org>>

PROFIBUS 3.502, *Profile Guidelines: Part 1 Identification and Maintenance Functions*, available at <http://www.profibus.com>

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 1035, *Domain names – implementation and specification*; available at <<http://www.ietf.org>>

IETF RFC 1112, *Host Extensions for IP Multicasting*; available at <<http://www.ietf.org>>

IETF RFC 2737, *Entity MIB (Version 2)*, available at <<http://www.ietf.org>>

IETF RFC 3490, *Internationalizing Domain Names in Applications (IDNA)*; available at <<http://www.ietf.org>>

Open Software Foundation (OSF): C706, *CAE Specification DCE11: Remote Procedure Call* (available at <<http://www.opengroup.org/onlinepubs/9629399/toc.htm>>)



Standards Survey

The IEC would like to offer you the best quality standards possible. To make sure that we continue to meet your needs, your feedback is essential. Would you please take a minute to answer the questions overleaf and fax them to us at +41 22 919 03 00 or mail them to the address below. Thank you!

Customer Service Centre (CSC)

International Electrotechnical Commission

3, rue de Varembe
1211 Genève 20
Switzerland

or

Fax to: **IEC/CSC** at +41 22 919 03 00

Thank you for your contribution to the standards-making process.

A Prioritaire

Nicht frankieren
Ne pas affranchir



Non affrancare
No stamp required

RÉPONSE PAYÉE

SUISSE

Customer Service Centre (CSC)
International Electrotechnical Commission
3, rue de Varembe
1211 GENEVA 20
Switzerland



Q1 Please report on **ONE STANDARD** and **ONE STANDARD ONLY**. Enter the exact number of the standard: (e.g. 60601-1-1)

.....

Q2 Please tell us in what capacity(ies) you bought the standard (tick all that apply). I am the/a:

- purchasing agent ☐
 librarian ☐
 researcher ☐
 design engineer ☐
 safety engineer ☐
 testing engineer ☐
 marketing specialist ☐
 other.....

Q3 I work for/in/as a:
(tick all that apply)

- manufacturing ☐
 consultant ☐
 government ☐
 test/certification facility ☐
 public utility ☐
 education ☐
 military ☐
 other.....

Q4 This standard will be used for:
(tick all that apply)

- general reference ☐
 product research ☐
 product design/development ☐
 specifications ☐
 tenders ☐
 quality assessment ☐
 certification ☐
 technical documentation ☐
 thesis ☐
 manufacturing ☐
 other.....

Q5 This standard meets my needs:
(tick one)

- not at all ☐
 nearly ☐
 fairly well ☐
 exactly ☐

Q6 If you ticked NOT AT ALL in Question 5 the reason is: (tick all that apply)

- standard is out of date ☐
 standard is incomplete ☐
 standard is too academic ☐
 standard is too superficial ☐
 title is misleading ☐
 I made the wrong choice ☐
 other

Q7 Please assess the standard in the following categories, using the numbers:

- (1) unacceptable,
 (2) below average,
 (3) average,
 (4) above average,
 (5) exceptional,
 (6) not applicable

- timeliness.....
 quality of writing.....
 technical contents.....
 logic of arrangement of contents
 tables, charts, graphs, figures.....
 other

Q8 I read/use the: (tick one)

- French text only ☐
 English text only ☐
 both English and French texts ☐

Q9 Please share any comment on any aspect of the IEC that you would like us to know:

.....



ISBN 2-8318-8067-X



ICS 25.040.40; 35.240.50; 35.100.05
