

INTERNATIONAL STANDARD



**Industrial communication networks – Fieldbus specifications –
Part 5-10: Application layer service definition – Type 10 elements**

IECNORM.COM : Click to view the full PDF of IEC 61158-5-10:2019



THIS PUBLICATION IS COPYRIGHT PROTECTED

Copyright © 2019 IEC, Geneva, Switzerland

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

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

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

About the IEC

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

About IEC publications

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

IEC publications search - webstore.iec.ch/advsearchform

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 It also gives information on projects, replaced and withdrawn publications.

IEC Just Published - webstore.iec.ch/justpublished

Stay up to date on all new IEC publications. Just Published details all new publications released. Available online and once a month by email.

IEC Customer Service Centre - webstore.iec.ch/csc

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

Electropedia - www.electropedia.org

The world's leading online dictionary on electrotechnology, containing more than 22 000 terminological entries in English and French, with equivalent terms in 16 additional languages. Also known as the International Electrotechnical Vocabulary (IEV) online.

IEC Glossary - std.iec.ch/glossary

67 000 electrotechnical terminology entries in English and French extracted from the Terms and Definitions clause of IEC publications issued since 2002. Some entries have been collected from earlier publications of IEC TC 37, 77, 86 and CISPR.

IECNORM.COM : Click to view the full text of IEC 61858-5:2019

INTERNATIONAL STANDARD



**Industrial communication networks – Fieldbus specifications –
Part 5-10: Application layer service definition – Type 10 elements**

INTERNATIONAL
ELECTROTECHNICAL
COMMISSION

ICS 25.040.40; 35.100.70; 35.110

ISBN 978-2-8322-6741-7

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

CONTENTS

FOREWORD.....	15
INTRODUCTION.....	17
1 Scope.....	18
1.1 General.....	18
1.2 Specifications	19
1.3 Conformance	19
2 Normative references	19
3 Terms, definitions, abbreviations, symbols and conventions	22
3.1 Referenced terms and definitions.....	22
3.1.1 ISO/IEC 7498-1 terms.....	22
3.1.2 ISO/IEC 8822 terms.....	22
3.1.3 ISO/IEC 9545 terms.....	22
3.1.4 ISO/IEC 8824-1 terms.....	23
3.2 Additional Type 10 terms and definitions.....	23
3.3 Additional Type 10 terms and definitions for media redundancy	30
3.4 Abbreviations and symbols	32
3.4.1 General	32
3.4.2 Additional Type 10 abbreviations and symbols.....	33
3.4.3 Abbreviations and symbols for services.....	34
3.5 Conventions.....	34
3.5.1 Overview	34
3.5.2 General conventions.....	34
3.5.3 Conventions for class definitions	34
3.5.4 Conventions for service definitions	36
3.5.5 Conventions used in state machines.....	37
4 Concepts.....	40
5 Data type ASE.....	40
5.1 General.....	40
5.1.1 Overview	40
5.1.2 Date and time type specifics	41
5.1.3 Transfer of user data	41
5.1.4 Data type overview	41
5.2 Formal definition of data type objects.....	44
5.2.1 Data type class.....	44
5.3 FAL defined data types	45
5.3.1 Fixed length types	45
5.3.2 Variable Length types	66
5.4 Data type ASE service specification.....	69
6 Communication model for common services	69
6.1 Concepts	69
6.1.1 Overview	69
6.1.2 Middle Layer ASEs	69
6.2 ASE data types	69
6.3 Application Service Elements.....	69
6.3.1 Remote procedure call ASE.....	69
6.3.2 Domain name system ASE	78

6.3.3	Simple network management ASE	79
6.3.4	IP suite ASE	80
6.3.5	Real time cyclic ASE	84
6.3.6	Real time acyclic ASE	100
6.3.7	Discovery and basic configuration ASE	110
6.3.8	Dynamic host configuration ASE	138
6.3.9	IEEE 802.1AB ASE	138
6.3.10	Media redundancy ASE	151
6.3.11	Precision time control ASE	158
6.3.12	IEEE 802.1AS ASE	172
6.3.13	Virtual Bridges ASE	175
6.3.14	MAC Bridges ASE	178
6.3.15	Fragmentation ASE	191
6.3.16	IEEE 802.3 ASE	192
6.3.17	Common DL mapping ASE	195
7	Communication model for decentralized periphery	200
7.1	Concepts	200
7.1.1	User requirements	200
7.1.2	Features	200
7.1.3	Associations	201
7.1.4	Device types	202
7.1.5	Instance model and device addresses	210
7.1.6	Application process	210
7.1.7	Application service element	217
7.1.8	Application relationship	219
7.2	ASE data types	220
7.3	ASEs	220
7.3.1	AR ASE	220
7.3.2	Real Identification ASE	355
7.3.3	Communication Interface Management ASE	445
7.3.4	Diagnosis ASE	516
7.3.5	PE ASE	561
7.3.6	LogBook ASE	571
7.3.7	RS ASE	574
7.3.8	Time ASE	599
7.4	Application characteristics	604
7.4.1	Device ident number	604
7.4.2	Network topology	604
7.5	Summary of FAL services	606
7.5.1	IO device	606
7.5.2	IO controller	607
7.5.3	IO Supervisor	607
Annex A (informative)	Device instances	608
Annex B (informative)	Components of an Ethernet interface	610
Annex C (informative)	Scheme of MAC address assignment	614
Annex D (informative)	Measurement of the fast startup time	615
Annex E (informative)	Dynamic Frame Packing	616
Annex F (informative)	Precondition for Diagnosis	624

Bibliography.....	629
Figure 1 – Middle Layer ASEs communication architecture	69
Figure 2 – Sequence Chart for reading the EndPointMapper.....	71
Figure 3 – Media redundancy diagnosis dependencies	157
Figure 4 – PTCP applications.....	158
Figure 5 – Example of periods at a local port	177
Figure 6 – Example of communication between controlling devices and field devices.....	201
Figure 7 – Example of communication between an engineering station and several controlling and field devices.....	202
Figure 8 – Example of communication between field devices and a server station	202
Figure 9 – Example of communication between field devices	202
Figure 10 – Data Objects and Diagnosis Data Model	206
Figure 11 – Example for channel modelling.....	207
Figure 12 – Mapping to PROFINET device model	208
Figure 13 – Identification hierarchy	208
Figure 14 – Application Process with application process objects (APOs)	211
Figure 15 – Access to a remote APO	212
Figure 16 – Access to a remote APO for provider/consumer association.....	213
Figure 17 – Overview of application processes	214
Figure 18 – IO device with APs, slots and subslots	214
Figure 19 – Example 1 structural units for interfaces and ports within API 0.....	216
Figure 20 – Example 2 structural units for interfaces and ports within API 0.....	217
Figure 21 – FAL ASEs communication architecture	218
Figure 22 – Example of one AR with two AREPs.....	220
Figure 23 – Example IO application relationship (one-to-one)	223
Figure 24 – Example IO application relationship one-to-many	224
Figure 25 – Implicit application relationship	225
Figure 26 – State transition diagram DEVSM	257
Figure 27 – State transition diagram CTLSM.....	265
Figure 28 – Assignment of Communication Relationship to Application Relationship.....	270
Figure 29 – Overview Communication Relationship Class service interactions	273
Figure 30 – Example for an intersection of IO device, slot, and AR	336
Figure 31 – Substitute Value.....	364
Figure 32 – State transition diagram RSMSM.....	372
Figure 33 – Ownership handling.....	376
Figure 34 – State transition diagram OWNSM.....	379
Figure 35 – State transition diagram ASSSM	380
Figure 36 – State transition diagram PLUGSM	392
Figure 37 – State transition diagram PULLSM.....	395
Figure 38 – Basic model for isochronous applications	424
Figure 39 – General isochronous application model (example CACF == 1)	425
Figure 40 – General isochronous application model (example CACF == 2)	426

Figure 41 – ASE relations in an IO device operating in isochronous mode for a submodule	432
Figure 42 – State transition diagram of ISOM_SYNC	434
Figure 43 – State transition diagram ISOM_OUT	437
Figure 44 – State transition diagram ISOM_IN	442
Figure 45 – State transition diagram SYNC_DIAG	484
Figure 46 – MRP interconnection	506
Figure 47 – Diagnosis Base Model	516
Figure 48 – Severity classification of fault, maintenance and qualified	517
Figure 49 – Data Base Model	518
Figure 50 – State transition diagram DIAG_DIAG	547
Figure 51 – State transition diagram DIAG_MR	551
Figure 52 – State transition diagram DIAG_MD	554
Figure 53 – State transition diagram DIAG_QUALIFIED	558
Figure 54 – Architecture	562
Figure 55 – State transition diagram PESM	570
Figure 56 – Reporting System components	575
Figure 57 – AR / ARSet and Reporting System	576
Figure 58 – Max Scan Delay	584
Figure 59 – RS Incident window	587
Figure 60 – State transition diagram RSOBS	592
Figure 61 – State transition diagram RSBUFF	596
Figure 62 – State transition diagram TimeSM	602
Figure 63 – Example of network topology including slower wireless segments	605
Figure 64 – Example of media redundancy including wireless segments	605
Figure A.1 – Instance model	608
Figure B.1 – Scheme of an Ethernet interface	610
Figure B.2 – Scheme of an Ethernet interface with bridging ability	611
Figure B.3 – Scheme of an Ethernet interface with optical ports	612
Figure B.4 – Scheme of an Ethernet interface with bridging ability using radio communication	613
Figure B.5 – Scheme of an Ethernet interface with radio communication	613
Figure C.1 – Scheme of MAC address assignment	614
Figure D.1 – Measurement of the fast startup time	615
Figure E.1 – Frame Layout	616
Figure E.2 – Subframe Layout	617
Figure E.3 – End to End	618
Figure E.4 – Dynamic frame packing	618
Figure E.5 – Dynamic frame packing – Truncation of outputs	619
Figure E.6 – Dynamic frame packing – Outbound Pack	619
Figure E.7 – Dynamic frame packing – Concatenation of inputs	620
Figure E.8 – Dynamic frame packing – Inbound Pack	621
Figure E.9 – Dynamic frame packing – Distributed watchdog	623
Figure E.10 – Interrelation between IO CR and dynamically packed frame	623

Table 1 – State machine description elements	37
Table 2 – Description of state machine elements	38
Table 3 – Conventions used in state machines	38
Table 4 – Conventions for services used in state machines	39
Table 5 – Data type overview.....	41
Table 6 – V2 octets.....	46
Table 7 – L2 octets	46
Table 8 – E2 octets.....	47
Table 9 – E2 value range.....	47
Table 10 – Unipolar2.16 octets	48
Table 11 – Unipolar2.16 value range	48
Table 12 – N2 value range.....	49
Table 13 – N4 value range.....	50
Table 14 – X2 value range.....	51
Table 15 – X4 value range.....	52
Table 16 – C4 value range.....	53
Table 17 – T2 value range	54
Table 18 – T4 value range	55
Table 19 – D2 value range.....	56
Table 20 – R2 value range.....	56
Table 21 – Status value range	57
Table 22 – Status value range	58
Table 23 – F message trailer with 4 octets.....	62
Table 24 – Unsigned16_S octets.....	64
Table 25 – Unsigned16_S meaning.....	64
Table 26 – Integer16_S octets.....	65
Table 27 – Integer16_S meaning	65
Table 28 – Unsigned8_S octets	65
Table 29 – Unsigned8_S meaning	65
Table 30 – OctetString_S octets	66
Table 31 – OctetString_S status bits.....	67
Table 32 – Connect.....	72
Table 33 – Release.....	73
Table 34 – Read	74
Table 35 – Write	75
Table 36 – Control	76
Table 37 – ReadImplicit	77
Table 38 – Add Static ARP Cache Entry	83
Table 39 – Remove Static ARP Cache Entry.....	83
Table 40 – PPM Set Prov Data	89
Table 41 – PPM Set Prov Status.....	90
Table 42 – PPM Activate.....	91

Table 43 – PPM Close	93
Table 44 – PPM Start.....	94
Table 45 – PPM Error	94
Table 46 – Get Cons Data	94
Table 47 – CPM Get cons status.....	95
Table 48 – CPM Set RedRole	96
Table 49 – CPM Activate	97
Table 50 – CPM NoData	99
Table 51 – CPM Stop.....	99
Table 52 – CPM New Data Indication	100
Table 53 – APMS Activate	103
Table 54 – APMR Activate	104
Table 55 – APMS A Data	106
Table 56 – APMR A Data	107
Table 57 – APMR Ack	107
Table 58 – APMS Error	108
Table 59 – APMS Error ERRCLS/ERRCODE	108
Table 60 – APMR Error	109
Table 61 – APMR Error ERRCLS/ERRCODE	109
Table 62 – APMS_Close	109
Table 63 – APMR_Close	110
Table 64 – Get.....	118
Table 65 – Set	123
Table 66 – Local Set Command	128
Table 67 – Identify	130
Table 68 – Hello	136
Table 69 – System capabilities	144
Table 70 – Auto negotiation support and status	145
Table 71 – MDI Power Support	146
Table 72 – Link aggregation status	146
Table 73 – Remote systems data change.....	150
Table 74 – Start bridge	165
Table 75 – Start slave	166
Table 76 – Start master	167
Table 77 – Stop bridge.....	168
Table 78 – Stop slave	169
Table 79 – Stop master	170
Table 80 – Sync state change	170
Table 81 – Line Delay change.....	171
Table 82 – Local Get Time	174
Table 83 – Local time state info	174
Table 84 – Allowed values of Forwarding Mode	181
Table 85 – Allowed values of Fast Forwarding Multicast MAC Add.....	181

Table 86 – Tx Port Entry	184
Table 87 – Dependencies of RedPeriodBegin and GreenPeriodBegin	187
Table 88 – Port state change	190
Table 89 – Set port state	190
Table 90 – Flush filtering data base	191
Table 91 – MAU Type change	194
Table 92 – Set MAU Type	194
Table 93 – P Data	195
Table 94 – N Data	197
Table 95 – A Data	198
Table 96 – C Data	199
Table 97 – Requirements and features	200
Table 98 – Binding application relationship services	226
Table 99 – Device Access	229
Table 100 – Companion AR	229
Table 101 – Acknowledge Companion AR	230
Table 102 – Startup Mode	230
Table 103 – Pull Module Alarm Allowed	230
Table 104 – Input Valid on Backup AR	233
Table 105 – Mode	234
Table 106 – APStructureIdentifier with API := 0	234
Table 107 – APStructureIdentifier with API != 0	235
Table 108 – RS Alarm Transport Mode	235
Table 109 – Connect	237
Table 110 – Connect Device Access	243
Table 111 – Release	244
Table 112 – Prm Begin	246
Table 113 – Prm End	248
Table 114 – Application Ready	249
Table 115 – Abort	250
Table 116 – Local AR Abort	251
Table 117 – Local Set AR State	251
Table 118 – Local AR In Data	251
Table 119 – Data elements of Read AR Data	252
Table 120 – Data elements of Expected Fast Startup Data	254
Table 121 – Remote primitives issued or received by DEVSM	255
Table 122 – Local primitives issued or received by DEVSM	256
Table 123 – State table DEVSM	258
Table 124 – Functions, Macros, Timers and Variables by DEVSM	263
Table 125 – Remote primitives issued or received by CTLSM	264
Table 126 – Local primitives issued or received by CTLSM	264
Table 127 – State table CTLSM	266
Table 128 – Functions, Macros, Timers and Variables used by CTLSM	268

Table 129 – Binding communication relationship services	274
Table 130 – Traffic Classes versus RT Class	276
Table 131 – Local Set Input	284
Table 132 – Local Set Input IOCS	285
Table 133 – Local Get Input	286
Table 134 – Local Get Input IOCS	287
Table 135 – Local New Input	288
Table 136 – Local Set Output	289
Table 137 – Local Set Output IOCS	290
Table 138 – Local Get Output	291
Table 139 – Local Get Output IOCS	292
Table 140 – Local New Output	293
Table 141 – Local Set Provider State	294
Table 142 – Local Set Redundancy	295
Table 143 – Local Set State	295
Table 144 – Local Data State Changed	296
Table 145 – Binding expected identification services	303
Table 146 – Module State	305
Table 147 – AR Info	308
Table 148 – Ident Info	309
Table 149 – General Data definition for identification services	309
Table 150 – Data elements of Read Module Diff Block	311
Table 151 – Alarm type	315
Table 152 – Alarm types attached to diagnosis ASE	317
Table 153 – Alarm types attached to ownership	317
Table 154 – Alarm types attached to common profiles, profiles, and application	318
Table 155 – Binding Alarm services	318
Table 156 – Alarm Notification	322
Table 157 – Channel Diagnosis	324
Table 158 – Manufacturer Specific Diagnosis	324
Table 159 – Submodule Diagnosis State	325
Table 160 – AR Diagnosis State	325
Table 161 – User Structure Identifier	326
Table 162 – Semantics of Specifier	328
Table 163 – Binding Record Data services	336
Table 164 – Read	338
Table 165 – Read Services	340
Table 166 – Read Implicit	342
Table 167 – Read Query	343
Table 168 – Write	344
Table 169 – Write Services	345
Table 170 – Data elements of Write Combined Object Container	346
Table 171 – Local Write Multiple	347

Table 172 – Local New Write Multiple	350
Table 173 – Binding real identification services.....	356
Table 174 – Local Add Submodule.....	359
Table 175 – Local Remove Submodule	360
Table 176 – Local Update Submodule.....	360
Table 177 – Data elements of Read API Data	362
Table 178 – Data elements of Read Record Input Data Object Element	363
Table 179 – Data elements of Read Record Output Data Object Element	366
Table 180 – Data elements of Read Substitute Value.....	367
Table 181 – Selector for Read GSD Data.....	369
Table 182 – Data elements of Read GSD Data	369
Table 183 – Remote primitives issued or received by RSMSM	371
Table 184 – Local primitives issued or received by RSMSM.....	371
Table 185 – State table RSMSM	372
Table 186 – Functions, Macros, Timers and Variables used by RSMSM	374
Table 187 – Rules for Submodule State.Ident Info	374
Table 188 – Remote primitives issued or received by OWNSM and ASSSM.....	378
Table 189 – Local primitives issued or received by OWNSM	379
Table 190 – State table OWNSM	380
Table 191 – State table ASSSM.....	387
Table 192 – Functions, Macros, Timers and Variables used by OWNSM.....	388
Table 193 – Functions, Macros, Timers and Variables used by ASSSM	389
Table 194 – Rules for Submodule State.AR Info	389
Table 195 – Remote primitives issued or received by PLUGSM	390
Table 196 – Local primitives issued or received by PLUGSM	391
Table 197 – State table PLUGSM	393
Table 198 – Functions, Macros, Timers and Variables used by PLUGSM.....	394
Table 199 – Remote primitives issued or received by PULLSM	395
Table 200 – Local primitives issued or received by PULLSM.....	395
Table 201 – State table PULLSM	396
Table 202 – Functions, Macros, Timers and Variables used by PULLSM	397
Table 203 – Binding I&M data services	399
Table 204 – Data elements of Read I&M0 Filter Data.....	412
Table 205 – Data elements of Read I&M0 Data.....	415
Table 206 – Data elements of Write I&M1 Data.....	416
Table 207 – Data elements of Write I&M2 Data.....	417
Table 208 – Data elements of Write I&M3 Data.....	418
Table 209 – Data elements of Write I&M4 Data.....	418
Table 210 – Data elements of Read I&M5 Data.....	419
Table 211 – Data elements of Read Asset Management Data	422
Table 212 – Binding Isochronous Mode Application services.....	426
Table 213 – Data elements of Write Isochronous Mode Data	431
Table 214 – Remote primitives issued or received by ISOM_SYNC.....	433

Table 215 – Local primitives issued or received by ISOM_SYNC	433
Table 216 – State table ISOM_SYNC	434
Table 217 – Functions, Macros, Timers and Variables used by the ISOM_SYNC	435
Table 218 – Remote primitives issued or received for ISOM_OUT	435
Table 219 – Local primitives issued or received for ISOM_OUT	436
Table 220 – State table ISOM_OUT	438
Table 221 – Functions, Macros, Timers and Variables used by the ISOM_OUT	440
Table 222 – Remote primitives issued or received for ISOM_IN	441
Table 223 – Local primitives issued or received for ISOM_IN	441
Table 224 – State table ISOM_IN	443
Table 225 – Functions, Macros, Timers and Variables used by the ISOM_IN	444
Table 226 – Binding Communication Interface Management services	446
Table 227 – Subslot number for interface submodules	449
Table 228 – Subslot number for port submodules	453
Table 229 – Data elements of Read PDev Data	458
Table 230 – Data elements of Read PD Real Data	458
Table 231 – Data elements of Read PD Expected Data	460
Table 232 – Data elements of Read PD Interface Data Real	461
Table 233 – Data elements of Write PD Interface Adjust	462
Table 234 – Data elements of Write PD Interface FSU Data Adjust	463
Table 235 – Data elements of Write PD NC Data Check	464
Table 236 – Data elements of Read PD Port Data Real	465
Table 237 – Data elements of Read PD Port Data Real Extended	466
Table 238 – Data elements of Write PD Port Data Check	467
Table 239 – Data elements of Write PD Port Data Adjust	469
Table 240 – Data elements of Read Port FO Data Real	470
Table 241 – Data elements of Write PD Port FO Data Check	471
Table 242 – Data elements of Write PD Port FO Data Adjust	471
Table 243 – Data elements of Read PD Port Statistic	472
Table 244 – Binding PTCP services	475
Table 245 – Sync Properties Role	478
Table 246 – Sync Class	478
Table 247 – Data elements of Write PD Sync Data	479
Table 248 – Local Sync State Info	481
Table 249 – Local SYNCH Event	482
Table 250 – Remote primitives issued or received by SYNC_DIAG	483
Table 251 – Local primitives issued or received by SYNC_DIAG	483
Table 252 – State table SYNC_DIAG	485
Table 253 – Functions, Macros, Timers and Variables used by SYNC_DIAG	486
Table 254 – Binding MAC Bridge services	487
Table 255 – Distributed Watchdog Factor	489
Table 256 – Restart Factor For Distributed Watchdog	489
Table 257 – DFP Mode	490

Table 258 – SFIOCRProperties.DFPRedundantPathLayout.....	490
Table 259 – SFCRC16	490
Table 260 – Data elements of Write PD IR Data	491
Table 261 – Data elements of Write PD IR Subframe Data.....	493
Table 262 – Binding Media Redundancy services	494
Table 263 – Data elements of Read PD Interface MRP Data Real.....	496
Table 264 – Data elements of Write PD Interface MRP Data Check	498
Table 265 – Data elements of Write PD Interface MRP Data Adjust	499
Table 266 – Data elements of Read PD Port MRP Data Real	501
Table 267 – Data elements of Write PD Port MRP Data Adjust	501
Table 268 – Data elements of Read PD Port MRPIC Data Real	502
Table 269 – Data elements of Write PD Port MRPIC Data Check	503
Table 270 – Data elements of Write PD Port MRPIC Data Adjust.....	503
Table 271 – Binding IEEE 802.1AS services	507
Table 272 – Data elements of Write PD Time Data	509
Table 273 – List of supported MIBs.....	509
Table 274 – Cross-referencing of MIB-II objects	510
Table 275 – Cross-referencing of LLDP-MIB objects	510
Table 276 – Cross-referencing of LLDP-EXT-PNIO MIB objects.....	510
Table 277 – Cross-referencing of LLDP-EXT-DOT3 MIB objects	511
Table 278 – Binding Autoconfiguration Data services.....	511
Table 279 – Data elements of Read Autoconfiguration Data	514
Table 280 – Binding Diagnosis services.....	521
Table 281 – General Data definition for Diagnosis services.....	527
Table 282 – Local Add Diagnosis Entry.....	530
Table 283 – Local Remove Diagnosis Entry	531
Table 284 – Local Update Diagnosis Entry.....	532
Table 285 – Remote primitives issued or received by DIAG_DIAG	547
Table 286 – Local primitives issued or received by DIAG_DIAG	547
Table 287 – State table DIAG_DIAG	548
Table 288 – Functions, Macros, Timers and Variables used by DIAG_DIAG	549
Table 289 – Remote primitives issued or received by DIAG_MR	550
Table 290 – Local primitives issued or received by DIAG_MR.....	551
Table 291 – State table DIAG_MR	552
Table 292 – Functions, Macros, Timers and Variables used by DIAG_MR.....	552
Table 293 – Remote primitives issued or received by DIAG_MD	554
Table 294 – Local primitives issued or received by DIAG_MD	554
Table 295 – State table DIAG_MD	555
Table 296 – Functions, Macros, Timers and Variables used by the maintenance demanded entry.....	556
Table 297 – Remote primitives issued or received by DIAG_QUALIFIED	557
Table 298 – Local primitives issued or received by DIAG_QUALIFIED	557
Table 299 – State table DIAG_QUALIFIED	559

Table 300 – Functions, Macros, Timers and Variables used by DIAG_QUALIFIED	559
Table 301 – Binding PE services	563
Table 302 – Local Add PE Entity	564
Table 303 – Local Remove PE Entity	565
Table 304 – Local Update PE_OperationalMode	566
Table 305 – PE Entity Filter Data definition	567
Table 306 – PE Entity Filter Data definition	568
Table 307 – Remote primitives issued or received by PESM	570
Table 308 – Local primitives issued or received by PESM	570
Table 309 – State table PESM	571
Table 310 – Functions, Macros, Timers and Variables used by PESM	571
Table 311 – Binding LogBook services	572
Table 312 – Data elements of Read LogBook Data	573
Table 313 – Local Create LogBook Entry	574
Table 314 – Binding RS services	581
Table 315 – Immanent observers	583
Table 316 – Configured observers	583
Table 317 – Data elements of Local Add RS Event	589
Table 318 – Data elements of RS Get Event	590
Table 319 – Data elements of RS Ack Event	590
Table 320 – Data elements of RS Adjust Observer	591
Table 321 – Local primitives issued or received by RSOBS	592
Table 322 – State table RSOBS	594
Table 323 – Functions, Macros, Timers and Variables used by RSOBS	595
Table 324 – Local primitives issued or received by RSBUF	596
Table 325 – State table RSBUF	597
Table 326 – Functions, Macros, Timers and Variables used by RSBUF	599
Table 327 – Binding Time services	600
Table 328 – Local Get Time	601
Table 329 – Local primitives issued or received by TimeSM	602
Table 330 – State table TimeSM	603
Table 331 – Functions, Macros, Timers and Variables used by TimeSM	604
Table 332 – FAL services of the IO device	606
Table 333 – FAL services of the IO controller	607
Table F.1 – ChannelErrorType	624
Table F.2 – Preconditions ChannelErrorType 0 – 0xFF	625
Table F.3 – Preconditions for ChannelErrorType 0x0100 – 0x7FFF, 0x9000 – 0x9FFF	625
Table F.4 – Preconditions for ChannelErrorType “Data transmission impossible”	625
Table F.5 – Preconditions for ChannelErrorType “Remote mismatch”	626
Table F.6 – Preconditions for ChannelErrorType “Media redundancy mismatch – ring”	626
Table F.7 – Preconditions for ChannelErrorType “Media redundancy mismatch - interconnection”	627
Table F.8 – Preconditions for ChannelErrorType “Sync mismatch”	627

Table F.9 – Preconditions for ChannelErrorType “Isochronous mode mismatch”	627
Table F.10 – Preconditions for ChannelErrorType “Multicast CR mismatch”	627
Table F.11 – Preconditions for ChannelErrorType “Fiber optic mismatch”	628
Table F.12 – Preconditions for ChannelErrorType “Network component function mismatch”	628
Table F.13 – Preconditions for ChannelErrorType “Dynamic Frame Packing function mismatch”	628
Table F.14 – Preconditions for ChannelErrorType “Media redundancy with planned duplication mismatch”	628
Table F.15 – Preconditions for ChannelErrorType “Multiple interface mismatch”	628

IECNORM.COM : Click to view the full PDF of IEC 61158-5-10:2019

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 5-10: Application layer service definition –
Type 10 elements****FOREWORD**

- 1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
- 2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
- 3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
- 4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
- 5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
- 6) All users should ensure that they have the latest edition of this publication.
- 7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
- 8) Attention is drawn to the normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
- 9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

This International Standard IEC 61158-5-10 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

This fourth edition cancels and replaces the third edition published in 2014. This edition constitutes a technical revision.

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

- integration of system redundancy basic functionality;
- integration of dynamic reconfiguration basic functionality;
- integration of reporting system basic functionality;

- integration of asset management basic functionality;
- integration of media redundancy ring interconnection basic functionality.

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

FDIS	Report on voting
65C/947/FDIS	65C/950/RVD

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

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

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

The committee has decided that the contents of this publication will remain unchanged until the stability date indicated on the IEC web site under "<http://webstore.iec.ch>" in the data related to the specific publication. At this date, the publication will be

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

A bilingual version of this publication may be issued at a later date.

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

INTRODUCTION

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

The application service is provided by the application protocol making use of the services available from the data-link or other immediately lower layer. This document defines the application service characteristics that fieldbus applications and/or system management may exploit.

Throughout the set of fieldbus standards, the term “service” refers to the abstract capability provided by one layer of the OSI Basic Reference Model to the layer immediately above. Thus, the application layer service defined in this document is a conceptual architectural service, independent of administrative and implementation divisions.

IECNORM.COM : Click to view the full PDF of IEC 61158-5-10:2019

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 5-10: Application layer service definition – Type 10 elements

1 Scope

1.1 General

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

This part of IEC 61158 provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 10 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This International Standard defines in an abstract way the externally visible service provided by the Type 10 fieldbus application layer in terms of

- a) an abstract model for defining application resources (objects) capable of being manipulated by users via the use of the FAL service,
- b) the primitive actions and events of the service;
- c) the parameters associated with each primitive action and event, and the form which they take; and
- d) the interrelationship between these actions and events, and their valid sequences.

The purpose of this document is to define the services provided to

- a) the FAL user at the boundary between the user and the application layer of the fieldbus reference model, and
- b) Systems Management at the boundary between the application layer and Systems Management of the fieldbus reference model.

This document specifies the structure and services of the Type 10 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI application layer structure (ISO/IEC 9545).

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.

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 document to provide access to the FAL to control certain aspects of its operation.

1.2 Specifications

The principal objective of this document is to specify the characteristics of conceptual application layer services suitable for time-critical communications, and thus supplement the OSI Basic Reference Model in guiding the development of application layer protocols for time-critical communications.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols. It is this latter objective which gives rise to the diversity of services standardized as the various Types of IEC 61158, and the corresponding protocols standardized in subparts of IEC 61158-6.

This specification may be used as the basis for formal application programming interfaces. Nevertheless, it is not a formal programming interface, and any such interface will need to address implementation issues not covered by this specification, including

- a) the sizes and octet ordering of various multi-octet service parameters, and
- b) the correlation of paired request and confirm, or indication and response, primitives.

1.3 Conformance

This document does not specify individual implementations or products, nor does it constrain the implementations of application layer entities within industrial automation systems.

There is no conformance of equipment to this application layer service definition standard. Instead, conformance is achieved through implementation of conforming application layer protocols that fulfill the Type 10 application layer services as defined in this document.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

IEC 61158-1:2019, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-5-3, *Industrial communication networks – Fieldbus specifications – Part 5-3: Application layer service definition – Type 3 elements*

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

IEC 61158-6-10:2019, *Industrial communication networks – Fieldbus specifications – Part 6-10: Application layer protocol specification – Type 10 elements*

IEC 61800-7-203:2015, *Adjustable speed electrical power drive systems – Part 7-203: Generic interface and use of profiles for power drive systems – Profile type 3 specification*

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

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

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

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

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

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

ISO/IEC 9834-8, *Information technology – Procedures for the operation of object identifier registration authorities – Part 8: Generation of universally unique identifiers (UUIDs) and their use in object identifiers*

ISO/IEC 10646, *Information technology – Universal Coded Character Set (UCS)*

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

ISO/IEC 15802-1, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Common specifications – Part 1: Medium Access Control (MAC) service definition*

ISO/IEC/IEEE 60559:2011, *Information technology – Microprocessor Systems – Floating-Point arithmetic*

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

IEEE Std 802, *IEEE Standard for Local and metropolitan area networks: Overview and Architecture*, available at <<http://www.ieee.org>>[viewed 2018-09-21]

IEEE Std 802.1, *IEEE Standard for Local and metropolitan area networks: Station and Media Access Control Connectivity Discovery*, available at <<http://www.ieee.org>>[viewed 2018-09-21]

IEEE Std 802.1AB, *IEEE Standard for Local and metropolitan area networks: Station and Media Access Control Connectivity Discovery*, available at <<http://www.ieee.org>>[viewed 2018-09-21]

IEEE Std 802.1AS, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – IEEE standard for Local and metropolitan area networks – Timing and Synchronization for Time-Sensitive Applications in Bridged Local Area Networks*, available at <<http://www.ieee.org>>[viewed 2018-09-21]

IEEE Std 802.1Q, *IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridged Local Area Networks*, available at <<http://www.ieee.org>>[viewed 2018-09-21]

IEEE Std 802.3, *IEEE Standard for Ethernet*, available at <<http://www.ieee.org>>[viewed 2018-09-21]

IETF RFC 768, *User Datagram Protocol*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 791, *Internet Protocol*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 792, *Internet Control Message Protocol*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

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>>[viewed 2018-09-21]

IETF RFC 894, *A Standard for the Transmission of IP Datagrams over Ethernet Networks*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 1034, *Domain names – concepts and facilities*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 1112, *Host Extensions for IP Multicasting*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 1157, *A Simple Network Management Protocol (SNMP)*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 1213, *Management Information Base for Network Management of TCP/IP-based internets: MIB-II*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 2131, *Dynamic Host Configuration Protocol*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 2132, *DHCP Options and BOOTP Vendor Extensions*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 2365, *Administratively Scoped IP Multicast*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 2674, *Definitions of Managed Objects for Bridges with Traffic Classes, Multicast Filtering and Virtual LAN Extensions*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 2737, *Entity MIB (Version 2)*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 2863, *The Interfaces Group MIB*; available at <<http://www.ietf.org>>[2018-09-21]

IETF RFC 3418, *Management Information Base (MIB) for the Simple Network Management Protocol (SNMP)*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 3490, *Internationalizing Domain Names in Applications (IDNA)*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 3621, *Power Ethernet MIB*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 4836, *Definitions of Managed Objects for IEEE 802.3 Medium Attachment Units (MAUs)*, available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 5905, *Network Time Protocol Version 4: Protocol and Algorithms Specification*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

IETF RFC 6890, *Special-Purpose IP Address Registries*; available at <<http://www.ietf.org>>[viewed 2018-09-21]

The Open Group — Publication C706, *Technical standard DCE1.1: Remote Procedure Call*, available at <<http://www.opengroup.org/onlinepubs/9629399/toc.htm>>[viewed 2018-09-21]

3 Terms, definitions, abbreviations, symbols and conventions

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

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

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

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

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

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

3.1.2 ISO/IEC 8822 terms

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

- a) abstract syntax
- b) presentation context

3.1.3 ISO/IEC 9545 terms

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

- a) application-association
- b) application-context
- c) application context name
- d) application-entity-invocation
- e) application-entity-type
- f) application-process-invocation
- g) application-process-type

- h) application-service-element
- i) application control service element

3.1.4 ISO/IEC 8824-1 terms

For the purposes of this document, the following terms as defined in ISO/IEC 8824-1 apply:

- a) object identifier
- b) type

3.2 Additional Type 10 terms and definitions

3.2.1

alarm

activation of an event that shows a critical state

3.2.2

alarm ack

acknowledgment of an event that shows a critical state

3.2.3

alarm data object

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

3.2.4

allocate

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

3.2.5

application

function or data structure for which data is consumed or produced

3.2.6

application layer interoperability

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

3.2.7

application objects

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

3.2.8

application process

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

3.2.9

application process identifier

attribute that identifies an application process used in a device

3.2.10

application process object

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

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

3.2.11

application process object class

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

3.2.12

application relationship

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

3.2.13

application relationship application service element

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

3.2.14

application relationship endpoint

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

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

3.2.15

attribute

description of an externally visible characteristic or feature of an object

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

3.2.16

backup

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

3.2.17

behavior

indication of how an object responds to particular events

3.2.18

channel

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

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

3.2.19

channel related diagnosis

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

EXAMPLE open loop

3.2.20**class**

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

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

3.2.21**class attributes**

attribute that is shared by all objects within the same class

3.2.22**class code**

unique identifier assigned to each object class

3.2.23**class specific service**

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

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

3.2.24**clear**

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

3.2.25**client**

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

3.2.26**common profile**

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

3.2.27**communication data object**

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

3.2.28**configuration check**

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

3.2.29**configuration fault**

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

3.2.30**configuration identifier**

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

3.2.31**consume**

act of receiving data from a provider

3.2.32

consumer

node or sink receiving data from a provider

3.2.33

context management

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

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

3.2.34

conveyance path

unidirectional flow of APDUs across an application relationship

3.2.35

cyclic

repetitive in a regular manner

3.2.36

data consistency

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

3.2.37

device

physical hardware connected to the link

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

3.2.38

device ID

vendor assigned device type identification

3.2.39

device profile

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

3.2.40

diagnosis data object

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

3.2.41

diagnosis information

all data available at the server for maintenance purposes

3.2.42

dynamic reconfiguration

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

3.2.43

endpoint

one of the communicating entities involved in a connection

3.2.44**engineering**

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

3.2.45**error**

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

3.2.46**error class**

general grouping for related error definitions and corresponding error codes

3.2.47**error code**

identification of a specific type of error within an error class

3.2.48**event**

instance of a change of conditions

3.2.49**extended channel related diagnosis**

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

EXAMPLE Link Fail

3.2.50**fast startup**

accelerated procedure to setup an IO device after a power cycle

3.2.51**frame**

unit of data transfer on a link

3.2.52**identification data object**

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

3.2.53**implicit AR endpoint**

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

3.2.54**index**

address of a record data object within an application process

3.2.55**instance**

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

3.2.56**instance attributes**

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

3.2.57

instantiated

object that has been created in a device

3.2.58

invocation

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

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

3.2.59

IO controller

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

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

3.2.60

IO data object

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

3.2.61

IO device

field device which acts as server for IO operation

3.2.62

IO parameter server

server for application parameter of IO devices (client)

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

3.2.63

IO subsystem

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

3.2.64

IO supervisor

engineering device which manages commissioning and/or diagnosis of an IO system

3.2.65

IO system

system composed of all its IO subsystems

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

3.2.66

Isochronous mode

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

3.2.67

member

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

3.2.68**method**

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

3.2.69**module**

hardware or logical component of a physical device

3.2.70**network**

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

3.2.71**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 a clearly defined interface and behavior

3.2.72**object specific service**

service unique to the object class which defines it

3.2.73**operate**

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

3.2.74**peer**

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

3.2.75**physical device**

automation or other network device

3.2.76**point-to-point connection**

connection that exists between exactly two application process objects

3.2.77**primary**

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

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

3.2.78**provider**

node or source sending data to one or many consumer

3.2.79**PTCP domain**

certain number of PTCP subdomains in one IP subnet

3.2.80**PTCP subdomain**

certain amount of DTEs with synchronized clocks

3.2.81

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

3.2.82

resource

processing or information capability

3.2.83

run

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

3.2.84

server

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

3.2.85

service

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

3.2.86

slot

address of a structural unit within an IO device

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

3.2.87

stop

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

3.2.88

submodule

hardware or logical component of a module

3.2.89

subslot

address of a structural unit within a slot

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

3.2.90

vendor ID

central administrative number used as manufacturer identification

3.3 Additional Type 10 terms and definitions for media redundancy

3.3.1

failure

termination of the ability of an item to perform a required function

Note 1 to entry: After failure the item has a fault.

Note 2 to entry: "Failure" is an event, as distinguished from "fault", which is a state.

Note 3 to entry: This concept as defined does not apply to items considering of software only.

3.3.2

fault

state of an item characterized by its inability to perform a required function, excluding the inability during preventive maintenance or other planned actions, or due to lack of external resources

Note 1 to entry: A fault is often the result of a failure of the item itself, but may exist without prior failure.

3.3.3

recovery

event when the item regains the ability to perform a required function, after a fault

3.3.4

recovery time

time period required for recovery

3.3.5

redundancy

existence in an item of two or more means of performing a required function

Note 1 to entry: In this context, the existence of more than one path (consisting of links and switches) between end nodes.

3.3.6

ring

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

Note 1 to entry: Nodes are connected to one another in the logical shape of a circle.

Note 2 to entry: Frames are passed sequentially between active nodes, each able to examine or modify the frame before forwarding it.

3.3.7

ringport

port of a switch to which a ring link is attached

3.3.8

ring link

link that connects two switches of a ring

3.3.9

switch node

MAC bridge as defined in IEEE 802.1Q

Note 1 to entry: It is called "switch" in this context.

3.4 Abbreviations and symbols

3.4.1 General

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
APMR	Acyclic Protocol Machine Receiver
APMS	Acyclic Protocol Machine Sender
APO	Application Process Object
AR	Application Relationship
AREP	Application Relationship End Point
ARL	Application Relationship List
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
BMA	Best Master Algorithm
CID	Connection ID
CIM	Computer Integrated Manufacturing
CIP	Control and Information Protocol
CR	Communication Relationship
CREP	Communication Relationship End Point
CTLSM	Controller state machine
DEVSM	Device state machine
DIAG_DIAG	Diagnosis entry state machine
DIAG_MD	Maintenance demanded entry state machine
DIAG_MR	Maintenance required entry state machine
DIAG_QUALIFIED	Qualified entry state machine
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLL	Data Link Layer
DLSAP	Data Link Service Access Point
DNS	Domain Name Service
DP	Decentralized Peripherals
FAL	Fieldbus Application Layer
FIFO	First In First Out
FODIAG	Fiber Optic Diagnosis
FOMD	Fiber Optic Maintenance Demanded
FOMR	Fiber Optic Maintenance Required
HMI	Human Machine Interface
ID	Identifier
IEC	International Electrotechnical Commission
IP	Internet Protocol
ISO	International Organization for Standardization
LME	Layer Management Entity
LOC_LNK	Local Link
MAU	Medium Attachment Unit
MIB	Management Information Base
MRA	Media Redundancy Manager (auto negotiation)
MRC	Media Redundancy Client
MRM	Media Redundancy Manager
MRP	Media Redundancy Protocol
MRPD	Media Redundancy with Planned Duplication of frames
OSI	Open Systems Interconnect
OWNSM	Ownership Submodule
PDev	Physical Device
PDU	Protocol Data Unit
PL	Physical Layer
PLUGSM	Plug State Machine

PULLSM	Pull State Machine
REM_CHK	Remote check (neighborhood check)
REP	Route Endpoint
RSMSM	Real Submodule State Machine
SAP	Service Access Point
SCL	Security Level
SDU	Service Data Unit
SEM	State Event Matrix
SMIB	System Management Information Base
SMK	System Management Kernel
SNMP	Simple Network Management Protocol
STD	State Transition Diagram, used to describe object behavior
S-VFD	Simple Virtual Field Device
SYNC_DIAG	Synchronization Diagnosis
UTC	Universal Time Coordinated

3.4.2 Additional Type 10 abbreviations and symbols

ACK	Acknowledged
ALPMI	Alarm Protocol Machine Initiator
ALPMR	Alarm Protocol Machine Responder
ARP	Address Resolution Protocol
BBUU	Buffer Buffer Unconfirmed Unidirectional
BBUU-OM	Buffer Buffer Unconfirmed Unidirectional - One to Many
BMC	Best Master Clock
CL-RPC	Connectionless Remote Procedure Call
CM	Context Management
CPM	Cyclic Consumer Protocol Machine
DCE	OSF Distributed Computing Environment
DCP	Discovery and basic Configuration Protocol
DHCP	Dynamic Host Configuration Protocol
DIM	Device Interface Module
DLPDU	Data Link-Protocol Data Unit
DTE	Date Terminal Equipment
GSDML	General Station Description Markup Language
I&M	Identification and Maintenance Profile
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
IDNA	Internationalizing Domain Names in Applications
IOCR	Input Output Communication Relation
IOCS	Input Output Object Consumer Status
IOPS	Input Output Object Provider Status
IR	Isochronous Relay
IRT	Isochronous Real Time Protocol
ISO	International Organization for Standardization
Isom	Isochronous Mode
ISOM_IN	Isochronous Mode Input Data state machine
ISOM_OUT	Isochronous Mode Output Data state machine
ISOM_SYNC	Isochronous Mode Sync state machine
LED	Light Emitting Diode
LLDP	Link Layer Discovery Protocol
LME	Layer Management Entity
lsb	Least Significant Bit
LT	Length/Type
MAC	Medium Access Control
MCR	Multicast Communication Relation
msb	Most Significant Bit
NCA	Network Computing Architecture
PL	Physical Layer
PPM	Cyclic Provider Protocol Machine

PTCP	Precision Transparent Clock Protocol
QQCB-CL	Queue Queue Confirmed Bi-directional - Connectionless
QQCB-CO	Queue Queue Confirmed Bi-directional - Connection-Oriented
RPC	Remote Procedure Call
RT	Real Time
RTA	Real Time Protocol Acyclic
RTC	Real Time Protocol Cyclic
RTC3PSM	Realtime Class 3 Port State Machine
TLV	Type Length Value (coding rule)
UDP	User Datagram Protocol
UUID	Universally Unique Identifier
VLAN	Virtual Local Area Network

3.4.3 Abbreviations and symbols for services

cnf	Confirmation
ind	Indication
req	Request
rsp	Response

3.5 Conventions

3.5.1 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 attributes are accessible from instances of the class using the Object Management ASE services specified in Clause 5 of this document. The service specification defines the services that are provided by the ASE.

3.5.2 General conventions

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

3.5.3 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 document, or by a user of this document.

(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 document to identify standardized classes. They have been assigned to maintain compatibility with existing national standards. CLASS IDs between 256 and 2 048 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 document.

(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 a 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.

Attributes and services are addressed with the following syntax:

- ASE:class:attribute
- ASE::attribute (if ASE contains only one class with same name)
- ASE:class:service():parameter
- ASE::service():parameter (if ASE contains only one class with same name)

Among other things, this is used for relation definitions.

3.5.4 Conventions for service definitions

3.5.4.1 General

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

3.5.4.2 Service parameters

Service primitives are used to represent service user/service provider interactions (see 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 specifications of this document use 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. This does not imply that the support of the parameter is optional. 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.

3.5.4.3 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 services get a “local” prefix in this document.

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

Interactions between the ASEs in the same layer are not shown here.

3.5.5 Conventions used in state machines

The protocol sequences are described by means of State Machines.

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

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

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

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

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

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

Table 1 – State machine description elements

#	Current state	Event or condition => action	Next state

Table 2 – Description of state machine elements

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

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

Table 3 – Conventions used in state machines

Convention	Meaning
:=	Value of an item on the left is replaced by value of an item on the right. If an item on the right is a parameter, it comes from the primitive shown as an input event.
xxx	A parameter name. Example: Identifier := reason means value of a 'reason' parameter is assigned to a parameter called 'Identifier.'
"xxx"	Indicates fixed value. Example: Identifier := "abc" means value "abc" is assigned to a parameter named 'Identifier.'
= or ==	A logical condition to indicate an item on the left is equal to an item on the right.
<	A logical condition to indicate an item on the left is less than the item on the right.
>	A logical condition to indicate an item on the left is greater than the item on the right.
<> or !=	A logical condition to indicate an item on the left is not equal to an item on the right.
>>	A semantic condition with the meaning "newer"
<<	A semantic condition with the meaning "older"
&&	Logical "AND"
	Logical "OR"
<=	A logical condition to indicate an item on the left is equal or less than the item on the right.
>=	A logical condition to indicate an item on the left is equal or greater than the item on the right.
for (Identifier := start_value to end_value) actions endfor	This construct allows the execution of a sequence of actions in a loop within one transition. The loop is executed for all values from start_value to end_value.
If (condition) actions else actions	This construct allows the execution of alternative actions depending on some condition (which might be the value of some identifier or the outcome of a previous action) within one transition. The parts beginning with "else" can be omitted if there is no action if the condition is not fulfilled.

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

Table 4 – Conventions for services used in state machines

Convention	Meaning
xxx.req ()	request according to ISO/IEC 7498-1 invokes the service and passes any required parameter parameters are omitted and listed within the state machine description
xxx.ind ()	indication according to ISO/IEC 7498-1 advices the activation of a requested service
xxx.rsp (+), xxx.cnf (+)	response according to ISO/IEC 7498-1 may positively acknowledge or complete an action previously invoked by a request primitive confirmation according to ISO/IEC 7498-1 primitive returned to the requestor to positively acknowledge or complete an action previously invoked by a request primitive parameters are omitted and listed within the state machine description
xxx.rsp (-), xxx.cnf (-)	response according to ISO/IEC 7498-1 may negatively acknowledge or complete an action previously invoked by a request primitive confirmation according to ISO/IEC 7498-1 primitive returned to the requestor to negatively acknowledge or complete an action previously invoked by a request primitive parameters are omitted and listed within the state machine description
xxx_req (), xxx_cnf (+), xxx_cnf (-)	this construct is used for the description of local and confirmed services, which are transferred between state machines parameters are listed within the state machine description and can be delivered inside the brackets optionally
xxx_ind ()	this construct is used for the description of local services and indicates the reception of this service parameters are listed within the state machine description and can be delivered inside the brackets optionally
xxx ()	this construct is used for the description of state machine internal functions (local) parameters are listed within the state machine description and can be delivered inside the brackets optionally
(), (+), (-)	replacement of the associated parameter list, described in the primitives definition except local services and functions can deliver parameters inside the brackets if the result is not necessary for further executions, the sign + and – can be omitted

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

In addition the following description elements are used:

Wildcard in names

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

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 Concepts

NOTE Clause 9 of IEC 61158-1 describes the concepts of the application layer service descriptions and the templates used in this document.

5 Data type ASE

5.1 General

5.1.1 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 Clause 5 are provided in IEC 61158-6-10.

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 document.

The basic data classes are used to define fixed length and bitstring data types. Standard types are taken from ISO/IEC 8824-1, other standard basic data types are defined specifically for fieldbus applications.

5.1.2 Date and time type specifics

Any date and time related data type, i.e. BinaryDate, TimeOfDay with and without date indication, TimeDifference with and without date indication, NetworkTime, NetworkTimeDifference, and TimeStamp is always related to a specific timescale.

Possible timescales are:

- Local time / local counter; used e.g. for working duration, ...
- WORKING_CLOCK; used e.g. for SendClock, Tinput, Toutput, SampleTime, ...
- Global time / Wall clock; Used e.g. for a comparable point in time, a start time, a stop time, ...

Usable timescales need to be strict monotonously rising, like TAI (temps atomique international), to allow any TimeDifference to be calculated.

Any usage of these data types is only meaningful when having the information, to which timescale the value is related.

Only the data type TimeStamp offers the timescale related to and whether the time is synchronized. All other data types have no indication to which timescale they belong to, or whether the time is synchronized to other devices.

NOTE IEEE 802.1AS provides a mechanism to synchronize devices to the TAI timescale. Thus, no Universal Time Coordinated (UTC) information is available.

5.1.3 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 IEC 61158-6-10. 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 OctetString is defined by the user.

5.1.4 Data type overview

Table 5 gives an overview over all data types defined within here.

Table 5 – Data type overview

Data type numeric identifier	Data type name	Usage	Alias name	Definition
1	Boolean	Type 10, Type 3	—	see 5.3.1.1
2	Integer8	Type 10, Type 3	char, int8_t	see 5.3.1.3.3.1
3	Integer16	Type 10, Type 3	short, int16_t	see 5.3.1.3.3.2
4	Integer32	Type 10, Type 3	long, int32_t	see 5.3.1.3.3.3
5	Unsigned8	Type 10, Type 3	unsigned char, uint8_t	see 5.3.1.3.4.1
6	Unsigned16	Type 10, Type 3	unsigned short, uint16_t	see 5.3.1.3.4.2
7	Unsigned32	Type 10, Type 3	unsigned long, uint32_t	see 5.3.1.3.4.3

Data type numeric identifier	Data type name	Usage	Alias name	Definition
8	Float32	Type 10, Type 3	float, float32_t	see 5.3.1.3.1.1
9	VisibleString	Type 10, Type 3	—	see 5.3.2.2
10	OctetString	Type 10, Type 3	—	see 5.3.2.1
12	TimeOfDay with date indication	Type 10, Type 3 should not use it	—	see 5.3.1.5.3
13	TimeDifference	obsolete	—	see 5.3.1.5.5
15	Float64	Type 10, Type 3	double, float64_t	see 5.3.1.3.1.2
39	UNICODEString	Type 10, Type 3 should not use it	—	see 5.3.2.4
40	UnicodeString8	Type 10, Type 3	—	see 5.3.2.5
41	61131_STRING	Type 10	—	see 5.3.2.6
42	61131_WSTRING	Type 10	—	see 5.3.2.7
50	BinaryDate	Type 10, Type 3 should not use it	—	see 5.3.1.5.2
52	TimeOfDay without date indication	Type 10, Type 3 should not use it	Unsigned32	see 5.3.1.5.4
53	TimeDifference with date indication	Type 10, Type 3 should not use it	—	see 5.3.1.5.6
54	TimeDifference without date indication	Type 10, Type 3 should not use it	Unsigned32	see 5.3.1.5.7
55	Integer64	Type 10, Type 3	int64_t	see 5.3.1.3.3.4
56	Unsigned64	Type 10, Type 3	uint64_t	see 5.3.1.3.4.4
58	NetworkTime	Type 10, Type 3 should not use it	Unsigned64	see 5.3.1.5.8
59	NetworkTimeDifference	Type 10, Type 3 should not use it	Integer64	see 5.3.1.5.9
60	TimeStamp	Type 10, Type 3	—	see 5.3.1.4.1
61	TimeStampDifference	Type 10, Type 3	—	see 5.3.1.4.2
62	TimeStampDifferenceShort	Type 10, Type 3	Integer64	see 5.3.1.4.3
100	Float64+Unsigned8	Type 10, Type 3	—	see 5.3.1.6.5
101	Float32+Unsigned8	Type 10, Type 3	—	see 5.3.1.6.6
102	Unsigned8+Unsigned8	Type 10, Type 3	—	see 5.3.1.6.7
103	OctetString2+Unsigned8	Type 10, Type 3	—	see 5.3.1.6.4

Data type numeric identifier	Data type name	Usage	Alias name	Definition
104	Unsigned16_S	Type 10, Type 3	—	see 5.3.1.6.8
105	Integer16_S	Type 10, Type 3	—	see 5.3.1.6.9
106	Unsigned8_S	Type 10, Type 3	—	see 5.3.1.6.10
107	OctetString_S	Type 10, Type 3	—	see 5.3.2.2
110	F message trailer with 4 octets	Type 10, Type 3	—	see 5.3.1.6.2
111	F message trailer with 5 octets	Type 10, Type 3	—	see 5.3.1.6.2
113	N2	Type 10, Type 3	Integer16	see 5.3.1.3.3.5
114	N4	Type 10, Type 3	Integer32	see 5.3.1.3.3.6
115	V2	Type 10, Type 3	—	see 5.3.1.2.1
116	L2	Type 10, Type 3	—	see 5.3.1.2.2
117	R2	Type 10, Type 3	Unsigned16	see 5.3.1.3.8
118	T2	Type 10, Type 3	Unsigned16	see 5.3.1.3.5
119	T4	Type 10, Type 3	Unsigned32	see 5.3.1.3.6
120	D2	Type 10, Type 3	Unsigned16	see 5.3.1.3.7
121	E2	Type 10, Type 3	—	see 5.3.1.3.2.1
122	C4	Type 10, Type 3	Integer32	see 5.3.1.3.3.9
123	X2	Type 10, Type 3	Integer16	see 5.3.1.3.3.7
124	X4	Type 10, Type 3	Integer32	see 5.3.1.3.3.8
125	Unipolar2.16	Type 10, Type 3	—	see 5.3.1.3.2.2
128 – 255	for this document: reserved for Type 3: user specific data types	Type 3	—	—
256 – 1 024	Reserved	Type 10	—	—
1 025	UUID	Type 10	—	see 5.3.1.6.1
1 026 – 65 535	Reserved	Type 10	—	—
other	Reserved	Type 10, Type 3	—	—

5.2 Formal definition of data type objects

5.2.1 Data type class

5.2.1.1 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

5.2.1.2 Attributes

Data type Numeric Identifier

This attribute identifies the numeric identifier of the related data type. The data type numeric identifier has a data type of Unsigned32.

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

Field Data type

This conditional 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 conditional 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 conditional 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

5.3 FAL defined data types**5.3.1 Fixed length types****5.3.1.1 Boolean types****5.3.1.1.1 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 (non zero) and FALSE (zero).

5.3.1.2 Bitstring types

5.3.1.2.1 Bit sequence V2

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 115
- 2 Data type Name = V2
- 3 Format = FIXED LENGTH
- 4.1 Octet Length = 2

This data type expresses a Bitstring data type consisting of 16 single bits combined in two octets and coded according to Table 6.

Each single bit has the values TRUE (1) and FALSE (0).

Table 6 – V2 octets

Bit	8	7	6	5	4	3	2	1
Octet 1	15	14	13	12	11	10	9	8
Octet 2	7	6	5	4	3	2	1	0

5.3.1.2.2 Nibble L2

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 116
- 2 Data type Name = L2
- 3 Format = FIXED LENGTH
- 4.1 Octet Length = 2

This data type expresses a Bitstring data type consisting of four nibbles combined in two octets. A nibble consists of four associated bits. The data type is coded according to Table 7.

The definition of a single nibble is not specified.

Table 7 – L2 octets

Bit	8	7	6	5	4	3	2	1
Octet 1	Nibble 3				Nibble 2			
Octet 2	Nibble 1				Nibble 0			

The definition of a single nibble is not specified.

5.3.1.3 Numeric types

5.3.1.3.1 Floating point types

5.3.1.3.1.1 Float32

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 8
- 2 Data type Name = Float32
- 3 Format = FIXED LENGTH
- 4.1 Octet Length = 4

This type has a length of four octets. The format for float32 is that defined by IEEE 754 as single precision.

NOTE "float" is an alias name for this data type.

5.3.1.3.1.2 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 IEEE 754 as double precision.

NOTE "double" is an alias name for this data type.

5.3.1.3.2 Fixed point types

5.3.1.3.2.1 Fixed point value E2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 121
2	Data type Name	= E2
3	Format	= FIXED LENGTH
4.1	Octet Length	= 2

This data type is a signed linear fixed-point value with a length of two octets having a sign-bit, eight binary places before the decimal point and seven binary places after it. The octets are coded according to Table 8.

Table 8 – E2 octets

Bit	8	7	6	5	4	3	2	1
Octet 1	SN	2^7	2^6	2^5	2^4	2^3	2^2	2^1
	sign	integer						
Octet 2	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}
	integer	fraction						

The value range of E2 is shown in Table 9.

Table 9 – E2 value range

Range of values	Resolution
$-256+2^{-7}$ to $256-2^{-7}$	$2^{-7} = 0,0078125$

5.3.1.3.2.2 Unipolar2.16

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	125
2	Data type Name	=	Unipolar2.16
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This data type is an unsigned linear fixed-point value with two octets having two binary places before the decimal point and fourteen binary places after it. The octets are coded according to Table 10 with a value range shown in Table 11. The data type expresses a value in percent of a span.

Table 10 – Unipolar2.16 octets

Bit	8	7	6	5	4	3	2	1
Octet 1	2^1	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}
	integer		fraction					
Octet 2	2^{-7}	2^{-8}	2^{-9}	2^{-10}	2^{-11}	2^{-12}	2^{-13}	2^{-14}
	fraction							

Table 11 – Unipolar2.16 value range

Range of values	Resolution
$0 \% \leq i \leq (400 \cdot 2^{-14}) \%$	$2^{-14} = 0,0061 \%$

NOTE This type is also used in IEC 61375-2-1 and expressed in IEC 60870-5-4 as 'unsigned fixed point number'.

5.3.1.3.3 Integer types

5.3.1.3.3.1 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.

NOTE "char" is an alias name for this data type.

5.3.1.3.3.2 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.

NOTE "short" is an alias name for this data type.

5.3.1.3.3.3 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.

NOTE "long" is an alias name for this data type.

5.3.1.3.3.4 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.

5.3.1.3.3.5 Normalized value N2**CLASS:** **Data type****ATTRIBUTES:**

1	Data type Numeric Identifier	= 113
2	Data type Name	= N2
3	Format	= FIXED LENGTH
4.1	Octet Length	= 2

This integer type is the same as Integer16; however having a special interpretation.

The value provides a linear normalized value, such that the integer value 2^{14} (i.e. 16 384) equates to 100 %. The value is thus interpreted according to the following equation:

$$\text{Normalized Value N2 (in \%)} = \text{Value} / 16\,384 \quad (1)$$

where

Normalized Value N2 is the calculated linear normalized value

Value is the raw value for interpretation

The data type offers thus a resolution of 0,006 1 % (= 1 / 16 384). The N2 value range is shown in Table 12.

Table 12 – N2 value range

Value	Meaning
-32 768	-200 %
-32 767 ... -1	-199,993 896 484 375 % ... -0,006 1 %
-16 384	-100 %
0	0 %
16 384	100 %
1 ... 32 766	0,006 1 % ... 199,987 792 968 75 %
32 767	199,993 896 484 375 %

5.3.1.3.3.6 Normalized value N4

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	114
2	Data type Name	=	N4
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	4

This integer type is the same as Integer32; however having a special interpretation.

The value provides a linear normalized value, such that the integer value 2^{30} (i.e. 1 073 741 824) equates to 100 %. The value is thus interpreted according to the following equation:

$$\text{Normalized Value N4 (in \%)} = \text{Value} / 1\,073\,741\,824 \quad (2)$$

where

Normalized Value N4 is the calculated linear normalized value
Value is the raw value for interpretation

The data type offers thus a resolution of approx. 0,000 000 093 1 % (= 1 / 1 073 741 824). The N4 value range is shown in Table 13.

Table 13 – N4 value range

Value	Meaning
-2 147 483 648	-200 %
-1 073 741 824	-100 %
-2 147 483 647.. -1	approx. -199,999 999 906 9 % ... -0,000 000 093 1 %
0	0 %
1 .. 2 147 483 646	approx. 0,000 000 093 1 % ... 199,999 999 813 7 %
1 073 741 824	100 %
2 147 483 647	approx. 199,999 999 906 9 %

5.3.1.3.3.7 Variable normalized X2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	123
2	Data type Name	=	X2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Integer16; however having a special interpretation.

The value is interpreted as a linear normalized value (a value expressed as percentage). The interpretation is similar to the data type N2, however the normalization to 100 % is variable and determined by an additional parameter “X” outside the scope of this data type, such that the integer value 2^X equates to 100 %.

NOTE If the additional specified constant X equates to 14, the value range of X2 equates to that of the data type N2.

The value is interpreted according to the following equation:

$$\text{Normalized Value X2 (in \%)} = \text{Value} / 2^X \quad (3)$$

where

<i>Normalized Value X2</i>	is the calculated linear normalized value
<i>Value</i>	is the raw value for interpretation
<i>X</i>	is the exponent for calculation, outside the scope of data type definition

The data type offers thus a resolution of $1 / 2^X$ % or else 2^{-X} . Lower and upper bound of the value are computed according to the following equations:

$$\text{Lower Bound X2 (in \%)} = -2^{(15-X)} \quad (4)$$

$$\text{Upper Bound X2 (in \%)} = 2^{(15-X)} - 2^{-X} \quad (5)$$

where

<i>Lower Bound X2</i>	is the lower bound of the calculated value
<i>Upper Bound X2</i>	is the upper bound of the calculated value
<i>X</i>	is the exponent for calculation, outside the scope of data type definition

The X2 value range for selected values of “X” is shown in Table 14.

Table 14 – X2 value range

Parameter “X”	Value	Meaning
—	-32 768 ... 32 767	$-2^{(15-X)} \% \leq i \leq (2^{(15-X)} - 2^{-X}) \%$
e.g. X = 10	-32 768 ... 32 767	$-3\,200 \% \leq i \leq -3\,199,90 \%$
e.g. X = 12	-32 768 ... 32 767	$-800 \% \leq i \leq 799,98 \%$
e.g. X = 14	-32 768 ... 32 767	$-200 \% \leq i \leq 199,99 \%$

The value of this normalized value parameter refers to the additional specified constant X. The associated constant X is required in order to interpret the internal value.

5.3.1.3.3.8 Variable normalized X4

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 124
2	Data type Name	= X4
3	Format	= FIXED LENGTH
4.1	Octet Length	= 4

This integer type is the same as Integer32; however having a special interpretation.

The value is interpreted as a linear normalized value (a value expressed as percentage). The interpretation is similar to the data type N4, however the normalization to 100 % is variable and determined by an additional parameter “X” outside the scope of this data type, such that the integer value 2^X equates to 100 %.

NOTE If the additional specified constant X equates to 30, the value range of X4 equates to that of the data type N4.

The value is interpreted according to the following equation:

$$\text{Normalized Value X4 (in \%)} = \text{Value} / 2^X \quad (6)$$

where

Normalized Value X4 is the calculated linear normalized value
Value is the raw value for interpretation
X is the exponent for calculation, outside the scope of data type definition

The data type offers thus a resolution of $1 / 2^X$ % or else 2^{-X} . Lower and upper bound of the value are computed according to the following equations:

$$\text{Lower Bound X4 (in \%)} = -2^{(31-X)} \quad (7)$$

$$\text{Upper Bound X4 (in \%)} = 2^{(31-X)} - 2^{-X} \quad (8)$$

where

Lower Bound X4 is the lower bound of the calculated value
Upper Bound X4 is the upper bound of the calculated value
X is the exponent for calculation, outside the scope of data type definition

The X4 value range for selected values of “X” is shown in Table 15.

Table 15 – X4 value range

Parameter “X”	Value	Meaning
—	–2 147 483 648 ... 2 147 483 647	$-2^{(31-X)} \% \leq i \leq (2^{(31-X)} - 2^{-X}) \%$
e.g. X = 26	–2 147 483 648 ... 2 147 483 647	$-3\,200 \% \leq i \leq -3\,200 \%$
e.g. X = 28	–2 147 483 648 ... 2 147 483 647	$-800 \% \leq i \leq 800 \%$
e.g. X = 30	–2 147 483 648 ... 2 147 483 647	$-200 \% \leq i \leq 200 \%$

The value of this normalized value parameter refers to the additional specified constant X. The associated constant X is required in order to interpret the internal value.

5.3.1.3.3.9 Fixed point value C4

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 122
2	Data type Name	= C4
3	Format	= FIXED LENGTH
4.1	Octet Length	= 4

This data type is the same as Integer32; however having a special interpretation.

The value is interpreted as a linear fixed point value with four decimal places according to the following equation:

$$\text{Fixed Point Value C4} = \text{Value} / 10\,000 \quad (9)$$

where

Fixed Point Value C4 is the calculated linear fixed point value
Value is the raw value for interpretation

The data type offers thus a resolution of 0,000 1 (= 1 / 10 000). The C4 value range is shown in Table 16.

Table 16 – C4 value range

Value	Meaning
-2 147 483 648	-214 748,3648
0	0
2 147 483 647	214 748,3647

5.3.1.3.4 Unsigned types

5.3.1.3.4.1 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 octet 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.

NOTE “unsigned char” is an alias name for this data type.

5.3.1.3.4.2 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 octet 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.

NOTE “unsigned short” is an alias name for this data type.

5.3.1.3.4.3 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 octet 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.

NOTE “unsigned long” is an alias name for this data type.

5.3.1.3.4.4 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.

5.3.1.3.5 Time constant T2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 118
2	Data type Name	= T2
3	Format	= FIXED LENGTH
4.1	Octet Length	= 2

This integer type is the same as Unsigned16; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

$$\text{Time Value} = \text{Value} * T_a \quad (10)$$

where

<i>Time Value</i>	is the calculated time value
<i>Value</i>	is the raw value for interpretation
<i>T_a</i>	is the defined sampling time, outside the scope of this data type

The data type offers thus a resolution of “Ta”. The T2 value range is shown in Table 17.

Table 17 – T2 value range

Value	Meaning
0 – 32 767	Value * Ta
32 768 – 65 535	0

The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.3.6 Time constant T4

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 119
2	Data type Name	= T4
3	Format	= FIXED LENGTH
4.1	Octet Length	= 4

This integer type is the same as Unsigned32; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

$$\text{Time Value} = \text{Value} * T_a \quad (11)$$

where

<i>Time Value</i>	is the calculated time value
<i>Value</i>	is the raw value for interpretation
<i>T_a</i>	is the defined sampling time, outside the scope of this data type

The data type offers thus a resolution of “Ta”. The T4 value range is shown in Table 18. In contrary to the data type T2 the value range is not restricted.

Table 18 – T4 value range

Value	Meaning
0 – 4 294 967 295	Value * Ta

The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.3.7 Time constant D2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 120
2	Data type Name	= D2
3	Format	= FIXED LENGTH
4.1	Octet Length	= 2

This integer type is the same as Unsigned16; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

$$\text{Time value} = \text{Value} \times (T_a / 16\,384) \quad (12)$$

where

<i>Time Value</i>	is the calculated time value
<i>Value</i>	is the raw value for interpretation
<i>T_a</i>	is the defined sampling time, outside the scope of this data type

The data type offers thus a fraction of the constant sampling time with a resolution of “Ta / 16 384”. The D2 value range is shown in Table 19. Note that the value range is restricted; values outside the result provide a time value of “0”.

Table 19 – D2 value range

Value	Meaning
0 – 32 767	Value * (Ta / 16 384)
32 768 – 65 535	0

The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.3.8 Time constant R2

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	117
2	Data type Name	=	R2
3	Format	=	FIXED LENGTH
4.1	Octet Length	=	2

This integer type is the same as Unsigned16; however having a special interpretation.

The value provides a time value based on a defined sampling time “Ta” outside the scope of this data type.

The value is interpreted according to the following equation:

$$\text{Time value} = 16\,384 * (\text{Ta} / \text{Value}) \quad (13)$$

where

<i>Time Value</i>	is the calculated time value
<i>T_a</i>	is the defined sampling time, outside the scope of data type definition
<i>Value</i>	is the raw value for interpretation

The data type offers thus a reciprocal multiple of the constant sampling time with a resolution of “Ta / Value”. The R2 value range is shown in Table 20. Note that the value range is restricted; values outside the result provide a time value of “0”.

Table 20 – R2 value range

Value	Meaning
0	16 384
1 – 16 384	16 384 * (Value / Ta)
16 385 – 65 535	16 384

The value of this time parameter refers to the additional specified constant sampling time Ta. The associated sampling time Ta is required in order to interpret the internal value.

5.3.1.4 Time and date types independent of calendar and time zone

5.3.1.4.1 TimeStamp

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	60

2	Data type Name	=	TimeStamp
3	Format	=	STRUCTURE
3.1	Number of Fields	=	4
3.2.1	Field Name	=	Status
3.2.2	Field Data type	=	Unsigned16
3.2.3	Field Name	=	SecondsHigh
3.2.4	Field Data type	=	Unsigned16
3.2.5	Field Name	=	SecondsLow
3.2.6	Field Data type	=	Unsigned32
3.2.7	Field Name	=	Nanoseconds
3.2.8	Field Data type	=	Unsigned32

This data type defines a 12 octet fixed length data type.

The value is interpreted according to the following equation:

$$\text{TimeStamp.Seconds} = (\text{SecondsHigh} \ll 32) + \text{SecondsLow} \quad (14)$$

$$\text{TimeStamp.Nanoseconds} = \text{Nanoseconds}$$

where

<i>TimeStamp.Seconds</i>	is the seconds part of the calculated timestamp value (a 48 bit value)
<i>TimeStamp.Nanoseconds</i>	is the nanoseconds part of the calculated timestamp value
<i>SecondsHigh</i>	is the raw value for interpretation
<i>SecondsLow</i>	is the raw value for interpretation
<i>Nanoseconds</i>	is the raw value for interpretation

A TimeStamp is built according to Table 21 and the following definitions:

- If the device is synchronized and in sync with the global time master the TimeStamp is related to the global time,
- Otherwise if the device has means of a local time, the TimeStamp is related to the local time.
- If the time is not battery buffered while the device is in power off, the status is set to “local time (arbitrary time scale)” until the device is synchronized to the global time.
- If the synchronization for the global time is lost, the status changes to “local time” until power off.

Table 21 – Status value range

Value	Meaning
0	TimeStamp related to the global synchronized time
1	TimeStamp related to the local time
2	TimeStamp related to the local time (arbitrary time scale)
other	reserved

5.3.1.4.2 TimeStampDifference

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	61
2	Data type Name	=	TimeStampDifference
3	Format	=	STRUCTURE
3.1	Number of Fields	=	4
3.2.1	Field Name	=	Status
3.2.2	Field Data type	=	Unsigned16
3.2.3	Field Name	=	SecondsHigh
3.2.4	Field Data type	=	Unsigned16
3.2.5	Field Name	=	SecondsLow
3.2.6	Field Data type	=	Unsigned32
3.2.7	Field Name	=	Nanoseconds
3.2.8	Field Data type	=	Unsigned32

This data type defines a 12 octet fixed length data type.

A TimeStampDifference is built by subtracting two TimeStamps.

The value is interpreted according to the following equation:

$$\text{TimeStampDifference.Seconds} = (\text{SecondsHigh} \ll 32) + \text{SecondsLow} \quad (15)$$

$$\text{TimeStampDifference.Nanoseconds} = \text{Nanoseconds}$$

where

<i>TimeStamp.Seconds</i>	is the seconds part of the calculated timestamp value (a 48 bit value)
<i>TimeStamp.Nanoseconds</i>	is the nanoseconds part of the calculated timestamp value
<i>SecondsHigh</i>	is the raw value for interpretation
<i>SecondsLow</i>	is the raw value for interpretation
<i>Nanoseconds</i>	is the raw value for interpretation

A TimeStampDifference is built according to Table 22:

Table 22 – Status value range

Value	Meaning
0	TimeStampDifference is positive
1	TimeStampDifference is negative
other	reserved

5.3.1.4.3 TimeStampDifferenceShort

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	62
2	Data type Name	=	TimeStampDifferenceShort
3	Format	=	FIXED LENGTH
3.1	Octet Length	=	8

This data type is the same as Integer64. The value is interpreted as the difference between two TimeStamp values in nanoseconds.

5.3.1.5 Time and date types needing calendar and time zone knowledge

5.3.1.5.1 General

The following data types need both calendar and time zone knowledge. Therefore the limitation of 5.1.2 applies. These data types should not be used.

5.3.1.5.2 BinaryDate

CLASS:		Data type
ATTRIBUTES:		
1	Data type Numeric Identifier	= 50
2	Data type Name	= BinaryDate
3	Format	= STRUCTURE
3.1	Number of Fields	= 6
3.2.1	Field Name	= Milliseconds
3.2.2	Field Data type	= Unsigned16
3.2.3	Field Name	= Minutes
3.2.4	Field Data type	= Unsigned8
3.2.5	Field Name	= Hours
3.2.6	Field Data type	= Unsigned8
3.2.7	Field Name	= Day
3.2.8	Field Data type	= Unsigned8
3.2.9	Field Name	= Month
3.2.10	Field Data type	= Unsigned8
3.2.11	Field Name	= Year
3.2.12	Field Data type	= Unsigned8

This structure expresses calendar date (by Year, Month, and Day) and time (by Hours, Minutes, and Milliseconds).

The field Day contains both day of the week and day of month; see IEC 61158-6-10 for coding.

The values 0 ... 50 of the field Year correspond to the years 2000 to 2050; the values 51 ... 99 correspond to the years 1951 to 1999.

This data type should not be used; see 5.1.2.

5.3.1.5.3 TimeOfDay with date indication

CLASS:		Data type
ATTRIBUTES:		
1	Data type Numeric Identifier	= 12
2	Data type Name	= TimeOfDay with date indication
3	Format	= STRUCTURE
3.1	Number of Fields	= 2
3.2.1	Field Name	= Milliseconds
3.2.2	Field Data type	= Unsigned32
3.2.3	Field Name	= Days
3.2.4	Field Data type	= Unsigned16

The field Milliseconds gives the time after the midnight in milliseconds.

The field Days gives the date counting the days from 1984-01-01.

This data type should not be used; see 5.1.2.

5.3.1.5.4 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
3.1	Octet Length	=	4

This data type is the same as Unsigned32. It gives the time after the midnight in milliseconds.

This data type should not be used; see 5.1.2.

5.3.1.5.5 TimeDifference

This data type is obsolete; use the data types TimeDifference with date indication (see 5.3.1.5.6) or TimeDifference without date indication (see 5.3.1.5.7) instead.

5.3.1.5.6 TimeDifference with date indication**CLASS:** Data type**ATTRIBUTES:**

1	Data type Numeric Identifier	=	53
2	Data type Name	=	TimeDifference with date indication
3	Format	=	STRUCTURE
3.1	Number of Fields	=	2
3.2.1	Field Name	=	Milliseconds
3.2.2	Field Data type	=	Unsigned32
3.2.3	Field Name	=	Days
3.2.4	Field Data type	=	Unsigned16

This data type gives the time difference in milliseconds and number of days.

This data type should not be used; see 5.1.2.

5.3.1.5.7 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
3.1	Octet Length	=	4

This data type is the same as Unsigned32. It gives the time difference in milliseconds.

This data type should not be used; see 5.1.2.

5.3.1.5.8 NetworkTime**CLASS:** Data type**ATTRIBUTES:**

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

This data type is the same as Unsigned64. Its interpretation is based on IETF RFC 5905. For encoding see IEC 61158-6-10.

This data type should not be used; see 5.1.2.

5.3.1.5.9 NetworkTimeDifference

CLASS: Data type

ATTRIBUTES:

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

This data type is the same as Integer64. For encoding see IEC 61158-6-10.

This data type should not be used; see 5.1.2.

5.3.1.6 Further types

5.3.1.6.1 UUID

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 1 025
2	Data type Name	= UUID
3	Format	= STRUCTURE
3.1	Number of Fields	= 4
3.2.1	Field Name	= Data1
3.2.2	Field Data type	= Unsigned32
3.2.3	Field Name	= Data2
3.2.4	Field Data type	= Unsigned16
3.2.5	Field Name	= Data3
3.2.6	Field Data type	= Unsigned16
3.2.7	Field Name	= Data4
3.2.8	Field Data type	= OctetString[8]

This data type defines a 16 octet fixed length data type. The semantic is specified by The Open Group – Publication C706 and beyond the scope of this document.

Specific values for decentralized peripherals items are defined in IEC 61158-6-10.

5.3.1.6.2 F message trailer with 4 octets

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	= 110
2	Data type Name	= F message trailer with 4 octets
3	Format	= FIXED LENGTH
3.1	Octet Length	= 4

This data structure consists of the status/control octet and 3 octets CRC parameters. This data type can be associated with input or output data up to 12 octets.

This data type is structured and coded according to Table 23.

Table 23 – F message trailer with 4 octets

Bits	7	6	5	4	3	2	1	0
Octet 1	Status/Control octet							
Octet 2	1. octet CRC (most significant octet)							
Octet 3	2. octet CRC							
Octet 4	3. octet CRC (least significant octet)							

5.3.1.6.3 F message trailer with 5 octets**CLASS:** Data type**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 111
- 2 Data type Name = F message trailer with 5 octets
- 3 Format = STRUCTURE
- 3.1 Number of Fields = 2
- 3.2 List of Fields
 - 3.2.1 Field Name = Status/Control
 - 3.2.2 Field Data type = Unsigned8
 - 3.2.3 Field Name = CRC
 - 3.2.4 Field Data type = Unsigned32

This data type defines the F message trailer with 5 octets and is structured as follows.

Status/Control

This field contains the value as Unsigned8.

CRC

This field describes the CRC.

This data type can be associated with input or output data up to 122 octets.

5.3.1.6.4 OctetString2+Unsigned8**CLASS:** Data type**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 103
- 2 Data type Name = OctetString2+Unsigned8
- 3 Format = STRUCTURE
- 3.1 Number of Fields = 2
- 3.2 List of Fields
 - 3.2.1 Field Name = Value
 - 3.2.2 Field Data type = OctetString[2]
 - 3.2.3 Field Name = ProcessValueStatus
 - 3.2.4 Field Data type = Unsigned8

This data type defines the OctetString2+Unsigned8 and is structured as follows.

Value

This field contains the value as OctetString with the length of 2.

ProcessValueStatus

This field contains the validity of the value (see 7.3.1.3.6.1.6).

5.3.1.6.5 Float64+Unsigned8**CLASS:** Data type**ATTRIBUTES:**

1	Data type Numeric Identifier	=	100
2	Data type Name	=	Float64+Unsigned8
3	Format	=	STRUCTURE
3.1	Number of Fields	=	2
3.2	List of Fields		
3.2.1	Field Name	=	Value
3.2.2	Field Data type	=	Float64
3.2.3	Field Name	=	ProcessValueStatus
3.2.4	Field Data type	=	Unsigned8

This data type defines the Float32+Unsigned8 and is structured as follows.

Value

This field contains the value as Float64.

ProcessValueStatus

This field contains the validity of the value (see 7.3.1.3.6.1.6).

5.3.1.6.6 Float32+Unsigned8**CLASS:** Data type**ATTRIBUTES:**

1	Data type Numeric Identifier	=	101
2	Data type Name	=	Float32+Unsigned8
3	Format	=	STRUCTURE
3.1	Number of Fields	=	2
3.2	List of Fields		
3.2.1	Field Name	=	Value
3.2.2	Field Data type	=	Float32
3.2.3	Field Name	=	ProcessValueStatus
3.2.4	Field Data type	=	Unsigned8

This data type defines the Float32+Unsigned8 and is structured as follows.

Value

This field contains the value as Float32.

ProcessValueStatus

This field contains the validity of the value (see 7.3.1.3.6.1.6).

5.3.1.6.7 Unsigned8+Unsigned8**CLASS:** Data type**ATTRIBUTES:**

1	Data type Numeric Identifier	=	102
2	Data type Name	=	Unsigned8+Unsigned8
3	Format	=	STRUCTURE
3.1	Number of Fields	=	2
3.2	List of Fields		
3.2.1	Field Name	=	Value
3.2.2	Field Data type	=	Unsigned8
3.2.3	Field Name	=	ProcessValueStatus
3.2.4	Field Data type	=	Unsigned8

This data type defines the Unsigned8+Unsigned8 and is structured as follows.

Value

This field contains the value as Unsigned8.

ProcessValueStatus

This field contains the validity of the value (see 7.3.1.3.6.1.6).

5.3.1.6.8 Unsigned16_S

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 104
- 2 Data type Name = Unsigned16_S
- 3 Format = FIXED LENGTH
- 3.1 Octet Length = 2

This data type is structured and coded according to Table 24 and Table 25.

Table 24 – Unsigned16_S octets

Bits	7	6	5	4	3	2	1	0
Octets 1	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶
Octets 2	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	St1	St0

ST1 and ST2 contain the status of the value (see 7.3.1.3.6.1.6).

Table 25 – Unsigned16_S meaning

St1 (bit1)	St0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.1.6.9 Integer16_S

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 105
- 2 Data type Name = Integer16_S
- 3 Format = FIXED LENGTH
- 3.1 Octet Length = 2

This data type is structured and coded according to Table 26 and Table 27.

SN: sign 0 = positive, 1 = negative

Table 26 – Integer16_S octets

Bits	7	6	5	4	3	2	1	0
Octets 1	SN	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	2 ⁷	2 ⁶
Octets 2	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	St1	St0

ST1 and ST2 contain the status of the value (see 7.3.1.3.6.1.6).

Table 27 – Integer16_S meaning

St1 (bit1)	St0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.1.6.10 Unsigned8_S

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 106
- 2 Data type Name = Unsigned8_S
- 3 Format = FIXED LENGTH
- 3.1 Octet Length = 1

This data type is structured and coded according to Table 28 and Table 29.

Table 28 – Unsigned8_S octets

Bits	7	6	5	4	3	2	1	0
Octets 1	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	St1	St0

ST1 and ST2 contain the status of the value (see 7.3.1.3.6.1.6).

Table 29 – Unsigned8_S meaning

St1 (bit1)	St0 (bit0)	meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.2 Variable Length types

5.3.2.1 OctetString

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 10
- 2 Data type Name = OctetString
- 3 Format = STRING
- 3.1 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. IEC 61158-6-10 defines the order of transmission.

NOTE IEC 61158-6-3 follows the definition of IEC 61158-6-10.

5.3.2.2 OctetString_S

CLASS: Data type

ATTRIBUTES:

- 1 Data type Numeric Identifier = 107
- 2 Data type Name = OctetString_S
- 3 Format = STRING
- 3.1 Octet Length = 3 to n

This data type is structured and coded according to Table 30 and Table 31.

Table 30 – OctetString_S octets

Bits	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
Octet 1	ch(8)	ch(7)	ch(6)	ch(5)	ch(4)	ch(3)	ch(2)	ch(1)
***	***	***	***	***	***	***	***	***
Octet m	ch(m*8)	ch(m*8-1)	ch(m*8-2)	ch(m*8-3)	ch(m*8-4)	ch(m*8-5)	ch(m*8-6)	ch(m*8-7)
Octet m+1	s1(4)	s0(4)	s1(3)	s0(3)	s1(2)	s0(2)	s1(1)	s0(1)
Octet m+2	s1(8)	s0(8)	s1(7)	s0(7)	s1(6)	s0(6)	s1(5)	s0(5)
***	***	***	***	***	***	***	***	***
Octet 3*m - 1	s1(m*8-4)	s0(m*8-4)	s1(m*8-5)	s0(m*8-5)	s1(m*8-6)	s0(m*8-6)	s1(m*8-7)	s0(m*8-7)
Octet 3*m	s1(m*8)	s0(m*8)	s1(m*8-1)	s0(m*8-1)	s1(m*8-2)	s0(m*8-2)	s1(m*8-3)	s0(m*8-3)

For each octet with channel values, there are two additional status octets “at the end”.

ch(x) value for channel x

s1(x) and s2(x) status for channel x (see 7.3.1.3.6.1.6 and table below)

Table 31 – OctetString_S status bits

s1 (bit1)	s0 (bit0)	Meaning
0	0	input channel: bad (value is fail-safe value) output channel: reserved
0	1	input channel: simulation output channel: reserved
1	0	input channel: uncertain output channel: reserved
1	1	input channel: good output channel: reserved

5.3.2.3 VisibleString**CLASS:** Data type**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 9
- 2 Data type Name = VisibleString
- 3 Format = STRING
- 3.1 Octet Length = 1 to n

This type is defined according to the ISO/IEC 646 International Reference Version string type without the "del" (coding 0x7F) character.

5.3.2.4 UNICODEString**CLASS:** Data type**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 39
- 2 Data type Name = UNICODEString
- 3 Format = ARRAY
- 3.1 Number of Array Elements = 1 to n
- 3.2 Array Element Data type = Unsigned16

This type is defined as the ISO/IE1046 string type with all UTF-16 coding.

This data type should not be used.

If the actual data contains a string which uses the full length of the data type (i.e. "n"), no terminator is needed. If the actual data is a string with a smaller length, a terminator with the value 0x0000 is needed to terminate the string.

To determine the string length, search from begin for 0x0000, if not found the value is "n", otherwise the position of the terminator minus 1.

5.3.2.5 UnicodeString8**CLASS:** Data type**ATTRIBUTES:**

- 1 Data type Numeric Identifier = 40
- 2 Data type Name = UnicodeString8
- 3 Format = STRING
- 3.1 Octet Length = 1 to n

This type is defined as the ISO/IEC 10646 string type with all UTF-8 coding.

If the actual data contains a string which uses the full length of the data type (i.e. "n"), no terminator is needed. If the actual data is a string with a smaller length, a terminator with the value 0x00 terminates the string.

To determine the string length, search from begin for NULL, if not found the value is "n", otherwise the position of the terminator minus 1.

5.3.2.6 61131_STRING

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	41
2	Data type Name	=	61131_STRING
3	Format	=	STRUCTURE
3.1	Number of Fields	=	3
3.2	List of Fields		
3.2.1	Field Name	=	MaxLength
3.2.2	Field Data type	=	Unsigned8
3.2.3	Field Name	=	ActualLength
3.2.4	Field Data type	=	Unsigned8
3.2.5	Field Name	=	SBCString
3.2.6	Format	=	STRING
3.2.6.1	Octet Length	=	MaxLength

This type is defined as the IEC 61131-3 STRING type, whereas SBCString is an ISO/IEC 646 International Reference Version string type without the "del" (coding 0x7F) character.

MaxLength is the maximum count of characters in SBCString and shall be greater than zero. ActualLength is the actual count of characters in SBCString and shall be lower or equal to MaxLength.

5.3.2.7 61131_WSTRING

CLASS: Data type

ATTRIBUTES:

1	Data type Numeric Identifier	=	42
2	Data type Name	=	61131_WSTRING
3	Format	=	STRUCTURE
3.1	Number of Fields	=	3
3.2	List of Fields		
3.2.1	Field Name	=	MaxLength
3.2.2	Field Data type	=	Unsigned16
3.2.3	Field Name	=	ActualLength
3.2.4	Field Data type	=	Unsigned16
3.2.5	Field Name	=	DBCString
3.2.6.1	Format	=	ARRAY
3.2.8.4.1	Number of Array Elements	=	MaxLength
3.2.8.4.2	Array Element Data type	=	Unsigned16

This type is defined as the IEC 61131-3 WSTRING type, whereas DBCString is an ISO/IEC 10646 string type with all UCS-2 coding. Allowed are Unicode characters from the BMP (basic multilingual page) only.

MaxLength is the maximum count of characters in DBCString and shall be greater than zero. ActualLength is the actual count of characters in DBCString and shall be lower or equal to MaxLength.

5.4 Data type ASE service specification

There are no operational services defined for the type object.

6 Communication model for common services

6.1 Concepts

6.1.1 Overview

The concepts of IEC 61158-1 common elements are applied. This document defines common services, which are used for decentralized periphery. Furthermore, this document defines the usage of, refinements, or extensions to the referenced standards.

6.1.2 Middle Layer ASEs

Figure 1 shows the common services ASEs according to their OSI layer. Protocols specific for this communication system are marked in orange, others are marked in yellow.

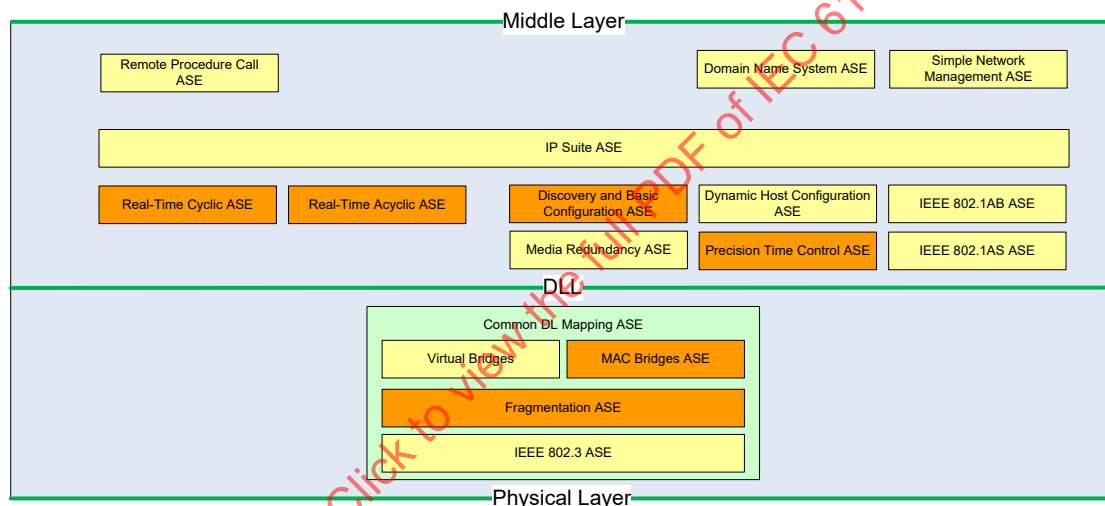


Figure 1 – Middle Layer ASEs communication architecture

6.2 ASE data types

The data types defined in Clause 5 are supported.

6.3 Application Service Elements

6.3.1 Remote procedure call ASE

6.3.1.1 Overview

The remote procedure call (RPC) concepts according to Publication C706 of the Open Group shall be applied. This ASE defines the special usage of the Publication C706 of the Open Group as a transport protocol.

6.3.1.2 RPC class specification

6.3.1.2.1 Template

The RPC object is described by the following template:

ASE: **RPC ASE**

CLASS: **RPC**

CLASS ID: **not used**

PARENT CLASS: **TOP**

ATTRIBUTES and SERVICES

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	Interface EMap
2.1	(o)	OpsService:	Insert
2.2	(o)	OpsService:	Delete
2.3	(m)	OpsService:	Lookup
2.4	(o)	OpsService:	Map
2.5	(m)	OpsService:	LookupHandleFree
2.6	(o)	OpsService:	InqObject
2.7	(o)	OpsService:	MgmtDelete
3	(m)	Attribute:	Interface IOD, IOC, IOS, IOParameterServer
3.1	(m)	OpsService:	Connect
3.2	(m)	OpsService:	Release
3.3	(m)	OpsService:	Read
3.4	(m)	OpsService:	Write
3.5	(m)	OpsService:	Control
3.6	(m)	OpsService:	ReadImplicit

6.3.1.2.2 Attributes

Implicit

The key attribute Implicit indicates that the object is implicitly addressed by the service.

Interface EMap

This attribute contains the end point mapper interface of the device. There is exactly one instance of this interface per device.

Interface IOD, IOC, IO system, IOParameterServer

This attribute contains the IO interface of the device. The type of the interface determines the addressed functionality (i.e. IO device, IO controller, IO supervisor or IO parameter server). From each type there may be several instances of each interface; for details see Annex A.

6.3.1.2.3 General service behavior

An AR class service request (i.e. connect, read...) shall be answered within less than 300 s; otherwise the client of the service may assume that the service request timed out.

At any point of time at most one outstanding service call is supported between an RPC client and the RPC server; thus succeeding service calls have to be in sequential order.

The explicit or implicit abort of a remote procedure call according to Publication C706 of the Open Group is implemented by local means.

The end point mapper is mapped to port 0x8894 (34 964) in opposite to the end point mapper as specified by Publication C706 which is mapped to port 0x0087 (135). Thus the end point mapper library functions (bound on port 135) of a standard RPC client are not usable; the client shall call the end point mapper interface (e.g. Lookup) at the port 0x8894 and evaluate its response.

The lookup response contains for a particular interface instance in its Protocol Tower information the destination port to use for the communication to this instance.

If a client makes a direct call to an interface instance, it has to respect the port forwarding of RPC end point mapper. The initial client request to the interface is directed to port 0x8894;

the response of the server is issued from its port. This source port of the response shall thus be used as destination port in further client calls to the interface instance.

6.3.1.3 RPC service specification

6.3.1.3.1 Interface EMap

6.3.1.3.1.1 Insert

For the definition of this optional service see Publication C706 of the Open Group.

6.3.1.3.1.2 Delete

For the definition of this optional service see Publication C706 of the Open Group.

6.3.1.3.1.3 Lookup

For the definition of this service see Publication C706 of the Open Group.

Figure 2 shows the correct behavior of the client and the server regarding an endpoint mapper lookup sequence.

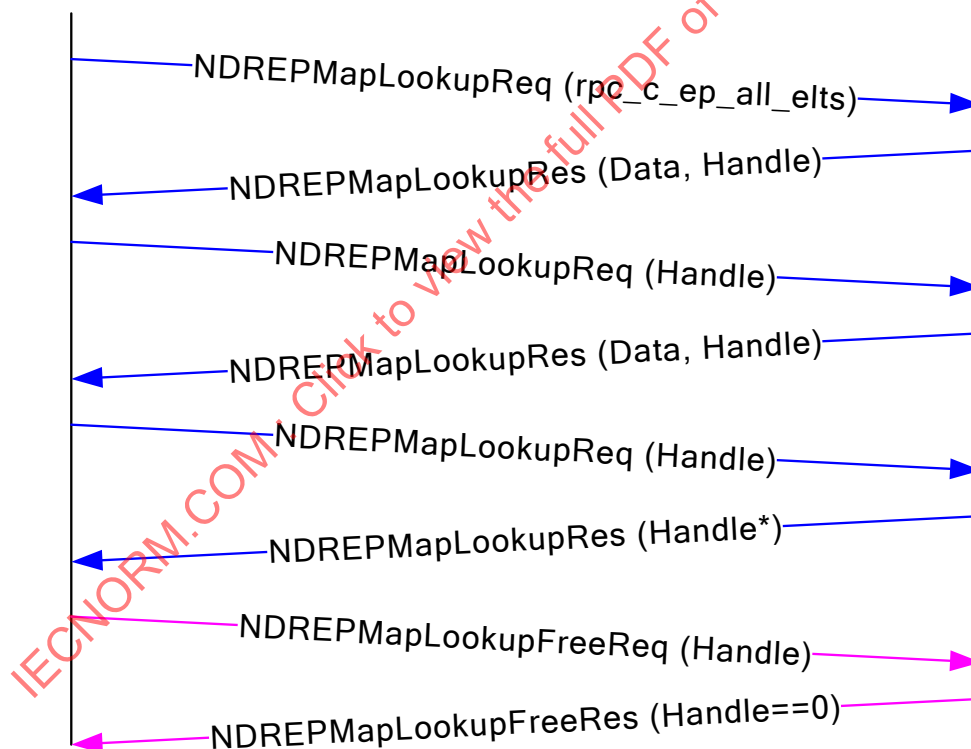


Figure 2 – Sequence Chart for reading the EndPointMapper

A server shall offer the client an auto destroy Handle feature by responding “Handle*:=0” without Data, if the client reads up to the end of the list. The client shall skip in this case the `NDREPMaPLookupFreeReq` service.

If the client reads not up to the end of list and skips the `NDREPMaPLookupFreeReq`, then the server destroys the Handle after a timeout defined in Publication C706 of the Open Group.

It is recommended to use a shorter timeout than the RPC default (e.g. 2 minutes to 5 minutes) to free resources earlier in this case (e.g. set timeout to 30 seconds).

6.3.1.3.1.4 Map

For the definition of this optional service see Publication C706 of the Open Group.

6.3.1.3.1.5 LookupHandleFree

For the definition of this service see Publication C706 of the Open Group.

6.3.1.3.1.6 InqObject

For the definition of this optional service see Publication C706 of the Open Group.

6.3.1.3.1.7 MgmtDelete

For the definition of this optional service see Publication C706 of the Open Group.

6.3.1.3.2 Interface IOD, IOC, IOS, IOPParameterServer

6.3.1.3.2.1 Connect

This confirmed service shall be used to establish a connection by means of Connectionless Remote Procedure Call (CL-RPC) Table 32 shows the parameters of the service.

Table 32 – Connect

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument

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

Args Max

This parameter shall contain the maximum buffer size available for the response.

Attribute type: Unsigned32

Args Len

This parameter shall contain the actual number of octets for the parameter Args.

Attribute type: Unsigned32

Args

This parameter shall contain the RPC user data.

Attribute type: OctetString

Result(+)

This parameter indicates that the service request succeeded.

PNIO Status

This parameter shall be set to zero.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

6.3.1.3.2.2 Release

This confirmed service shall be used to release a connection by means of CL-RPC. Table 33 shows the parameters of the service.

Table 33 – Release

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument

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

Args Max

This parameter shall contain the maximum buffer size available for the response.

Attribute type: Unsigned32

Args Len

This parameter shall contain the actual number of octets for the parameter Args.

Attribute type: Unsigned32

Args

This parameter shall contain the RPC user data.

Attribute type: OctetString

Result(+)

This parameter indicates that the service request succeeded.

PNIO Status

This parameter shall be set to zero.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

6.3.1.3.2.3 Read

This confirmed service shall be used to convey a higher layer read service by means of CL-RPC. Table 34 shows the parameters of the service.

Table 34 – Read

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument

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

Args Max

This parameter shall contain the maximum buffer size available for the response.

Attribute type: Unsigned32

Args Len

This parameter shall contain the actual number of octets for the parameter Args.

Attribute type: Unsigned32

Args

This parameter shall contain the RPC user data.

Attribute type: OctetString

Result(+)

This parameter indicates that the service request succeeded.

PNIO Status

This parameter shall be set to zero.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

6.3.1.3.2.4 Write

This confirmed service shall be used to convey a higher layer write service by means of CL–RPC. Table 35 shows the parameters of the service.

Table 35 – Write

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument

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

Args Max

This parameter shall contain the maximum buffer size available for the response.

Attribute type: Unsigned32

Args Len

This parameter shall contain the actual number of octets for the parameter Args.

Attribute type: Unsigned32

Args

This parameter shall contain the RPC user data.

Attribute type: OctetString

Result(+)

This parameter indicates that the service request succeeded.

PNIO Status

This parameter shall be set to zero.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

6.3.1.3.2.5 Control

This confirmed service shall be used to convey a higher layer control service by means of CL-RPC. Table 36 shows the parameters of the service.

Table 36 – Control

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument

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

Args Max

This parameter shall contain the maximum buffer size available for the response.

Attribute type: Unsigned32

Args Len

This parameter shall contain the actual number of octets for the parameter Args.

Attribute type: Unsigned32

Args

This parameter shall contain the RPC user data.

Attribute type: OctetString

Result(+)

This parameter indicates that the service request succeeded.

PNIO Status

This parameter shall be set to zero.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

6.3.1.3.2.6 ReadImplicit

This confirmed service shall be used to convey a higher layer read implicit service by means of CL-RPC. Table 34 shows the parameters of the service.

Table 37 – ReadImplicit

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Args Max	M	M(=)		
Args Len	M	M(=)		
Args	M	M(=)		
Result(+)			S	S(=)
PNIO Status			M	M(=)
Args Max			M	M(=)
Args			M	M(=)
Result(-)			S	S(=)
PNIO Status			M	M(=)

Argument

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

Args Max

This parameter shall contain the maximum buffer size available for the response.

Attribute type: Unsigned32

Args Len

This parameter shall contain the actual number of octets for the parameter Args.

Attribute type: Unsigned32

Args

This parameter shall contain the RPC user data.

Attribute type: OctetString

Result(+)

This parameter indicates that the service request succeeded.

PNIO Status

This parameter shall be set to zero.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

6.3.1.4 Invocation of the RPC object

There is one RPC object for each interface.

6.3.2 Domain name system ASE

6.3.2.1 Overview

The utilization of the IETF RFC 1034 (DNS) is defined for this specification. It includes the usage and the content syntax of domain name.

6.3.2.2 DNS class specification

6.3.2.2.1 General

The DNS ASE defines one DNS object type:

6.3.2.2.2 Template

A DNS object is described by the following template:

ASE: DNS ASE

CLASS: DNS

CLASS ID: not used

PARENT CLASS: top

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: DNS
- 2.1 (m) Attribute: Preferred DNS Server
- 2.2 (m) Attribute: Alternate DNS Server 1
- 2.3 (m) Attribute: Alternate DNS Server 2
- 2.4 (m) Attribute: Alternate DNS Server 3

SERVICES:

- 1 (o) OpsService: DNS Get Host By Name

6.3.2.2.3 Attributes

Implicit

The key attribute Implicit indicates that the DNS object is implicitly addressed by the service.

DNS

This attribute contains the following attributes:

Preferred DNS Server

This attribute contains the IP address of the preferred DNS server according to IETF RFC 791.

Attribute type: Unsigned32

Alternate DNS Server 1

This attribute contains the IP address of the first alternate DNS server according to IETF RFC 791.

Attribute type: Unsigned32

Alternate DNS Server 2

This attribute contains the IP address of the second alternate DNS server according to IETF RFC 791.

Attribute type: Unsigned32

Alternate DNS Server 3

This attribute contains the IP address of the third alternate DNS server according to IETF RFC 791.

Attribute type: Unsigned32

6.3.2.3 DNS service specification

The DNS services shall be derived from the IETF RFC 1034 (DNS).

6.3.2.4 Invocation of the DNS object

There is one DNS object for the device.

6.3.3 Simple network management ASE

6.3.3.1 Overview

SNMP is described in the IETF RFC 1157 (SNMP).

6.3.3.2 SNMP class specification

6.3.3.2.1 General

The SNMP ASE defines one SNMP object type.

6.3.3.2.2 Template

A SNMP object is described by the following template:

ASE:	SNMP ASE
CLASS:	SNMP
CLASS ID:	not used
PARENT CLASS:	top
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
SERVICES:	
1	(m) OpsService: SNMP Get
2	(m) OpsService: SNMP Get Next
3	(m) OpsService: SNMP Set

6.3.3.2.3 Attributes

Implicit

The key attribute Implicit indicates that the SNMP object is implicitly addressed by the service.

6.3.3.3 SNMP service specification

The SNMP services shall be derived from the IETF RFC 1157 (SNMP).

6.3.3.4 Invocation of the SNMP object

There is one SNMP object for the device.

6.3.3.5 Behavior

6.3.3.5.1 Persistency

For persistency of the SNMP object the persistency defined in the Communication Interface Management ASE is used; see 7.3.3.2.6.2 for details.

6.3.4 IP suite ASE

6.3.4.1 Overview

The utilization of the IETF RFC 768 (UDP), IETF RFC 791 (IP), IETF RFC 792 (ICMP), IETF RFC 826 (ARP), IETF RFC 1112 (IP Multicasting) is defined for this document. It includes definitions for UDP ports and IP multicast addresses and utilization of IP header fields for RT over UDP.

This ASE shall contain current volatile values of operation. Permanent values in nonvolatile memory shall be a matter of the DCP ASE defined in 6.3.1.

6.3.4.2 ARP cache behavior

Any communication relying on IP needs ARP (address resolution protocol) for the address translation between IP and MAC addresses.

ARP itself uses a so called ARP cache to minimize the amount of address translation queries going over the network.

This ARP cache is partitioned into static and dynamic entries, whereas static entries have an “unlimited” lifetime (entry type = STATIC); they are only lost with a Power Cycle. Dynamic entries (entry type = AGED) are aged out using the Cache Timeout parameter. They require thus a new address translation query on the network after each ARP cache timeout resulting in a higher ARP netload. If a dynamic entry has a wrong content (e.g. due to duplicated IP addresses on the network, or due to usage of the same IP address for multiple devices), the communication will work only again after aging out the ARP cache (this is the use case fast shutdown).

Therefore a static entry should be added to the ARP cache by IO controller and IO device whenever an application relation is established.

A static entry related to an application relation shall be removed whenever the application relation is released respectively aborted.

This has two benefits: On one hand it avoids superfluous ARPs to maintain the dynamic ARP cache entries, on the other hand it allows fast shutdown.

ARP announcements are meaningless together with static ARP cache entries and thus should not be issued by IO devices and/or IO controllers.

The minimum number of static entries should be equal or higher than either

- the number of supported application relations for an IO controller, or
- the number of concurrent application relations for an IO device.

If both are supported concurrently, the sum of entries shall be supported.

More entries, static and/or dynamic, are needed for other protocols or additional communication partners.

6.3.4.3 IP suite class specification

6.3.4.3.1 General

The IP suite ASE defines one Communication Interface Management object type:

6.3.4.3.2 Template

A IP suite object is described by the following template:

ASE:	IP suite ASE
CLASS:	IP suite
CLASS ID:	not used
PARENT CLASS:	top
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: IP
2.1	(m) Attribute: IP Address
2.2	(m) Attribute: Subnet Mask
2.3	(m) Attribute: Standard Gateway
3	(m) Attribute: ARP
3.1	(m) Attribute: Cache Size
3.2	(m) Attribute: Cache Timeout
3.3	(m) Attribute: List of Cache Entries
3.3.1	(m) Attribute: Entry Type
3.3.2	(m) Attribute: IEEE 802.3 MAC Address
3.3.3	(m) Attribute: IP Address
4	(m) Attribute: List of IP Multicast addresses
4.1	(m) Attribute: IP Multicast address
SERVICES:	
1	(o) OpsService: Add Static ARP Cache Entry
2	(o) OpsService: Remove Static ARP Cache Entry

6.3.4.3.3 Attributes

Implicit

The key attribute Implicit indicates that the IP suite object is implicitly addressed by the service.

IP

This attribute contains the following attributes:

IP Address

This attribute contains the current IP address according to IETF RFC 791.

Attribute type: Unsigned32 (usually written as 4 octets separated by a dot)

Subnet Mask

This attribute contains the current subnet mask according to IETF RFC 791.

Attribute type: Unsigned32 (usually written as 4 octets separated by a dot)

Standard Gateway

This attribute contains the current IP address of the standard gateway according to IETF RFC 791.

Attribute type: Unsigned32 (usually written as 4 octets separated by a dot)

ARP

This attribute contains the following attributes:

Cache Size

This attribute contains the size of the ARP cache and shall contain a value to cache all devices that the device is connected to.

NOTE 1 IEC 61784-2 defines a recommended value.

NOTE 2 The cache size influences the performance of the IO operation.

Attribute type: Unsigned32

Cache Timeout

This attribute contains the timeout value for the refresh of the dynamic ARP cache entry (see IETF RFC 826). This attribute influences the time to recognize changed IP address / MAC address couples in case of device replacement.

NOTE IEC 61784-2 defines a recommended value.

Attribute type: Unsigned32

List of Cache Entries

This attribute list contains the following attributes:

Entry Type

This attribute contains the lifetime of the cache entry.

Attribute type: Unsigned8

Allowed Values: AGED (entry is subject for cache refresh), STATIC (static entry)

IEEE 802.3 MAC Address

This attribute contains the value of the IEEE 802 MAC address corresponding to the IP address.

Attribute type: OctetString[6]

IP Address

This attribute contains the IP address according to IETF RFC 791.

Attribute type: Unsigned32

List of IP Multicast addresses

This attribute list contains the following attributes:

IP Multicast address

This attribute contains the IP Multicast address used for multicast communication relation. It shall be set according IETF RFC 2365 and shall follow the building rule in relation to the Frame ID shown in IEC 61158-6-10.

Attribute type: Unsigned32

Allowed values with associated MAC address and Frame ID are listed in IEC 61158-6-10.

6.3.4.4 IP suite service specification

6.3.4.4.1 Add Static ARP Cache Entry

This optional local service adds a new static ARP cache entry, which from now on is leading for the IP to MAC resolution.

Table 38 shows the parameters of the service.

Table 38 – Add Static ARP Cache Entry

Parameter name	req	cnf
Argument	M	
IEEE 802.3 MAC Address	M	
IP Address	M	
Result(+)		S(=)
Result(-)		S(=)

Argument

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

IEEE 802.3 MAC Address

This parameter contains the value for the equivalent ASE attribute.

IP Address

This parameter contains the value for the equivalent ASE attribute.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

6.3.4.4.2 Remove Static ARP Cache Entry

This optional local service removes a static ARP cache entry.

Table 39 shows the parameters of the service.

Table 39 – Remove Static ARP Cache Entry

Parameter name	req	cnf
Argument	M	
IEEE 802.3 MAC Address	M	
IP Address	M	
Result(+)		S(=)
Result(-)		S(=)

Argument

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

IEEE 802.3 MAC Address

This parameter contains the value for the equivalent ASE attribute.

IP Address

This parameter contains the value for the equivalent ASE attribute.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

6.3.4.5 Invocation of the IP suite object

There is one IP suite object for each interface.

6.3.5 Real time cyclic ASE

6.3.5.1 Overview

The real time protocol cyclic (RTC) is a concept of reading and writing cyclic data by means of buffered services. It uses IEEE 802.3 or User Datagram Protocol (UDP) services, and the virtual bridges and MAC bridges ASE (see 6.3.13 and 6.3.13.4).

6.3.5.2 RTC class specification

6.3.5.2.1 General

The RTC ASE defines for each particular CR one RTC object.

6.3.5.2.2 Template

A RTC object is described by the following template:

ASE:	RTC ASE
CLASS:	RTC
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: CREP
2	(m) Attribute: Protocol machine Parameter
2.1	(m) Attribute: DA
2.2	(m) Attribute: SA
2.3	(m) Attribute: Frame ID
2.4	(m) Attribute: Protocol Machine
2.5	(c) Constraint: Protocol Machine = PPM
2.5.1	(m) Attribute: VLAN Prio
2.5.2	(m) Attribute: VLAN ID
2.5.3	(m) Attribute: Tx Option
2.5.3.1	(m) Attribute: RT Class
2.5.3.2	(c) Constraint: RT Class = RT_CLASS_3
2.5.3.2.1	(m) Attribute: Frame Send Offset
2.5.3.2.2	(m) Attribute: Tx Port
2.5.3.2.3	(m) Attribute: Startup Mode
2.5.3.2.4	(m) Attribute: Media Redundancy
2.5.3.2.5	(c) Constraint: Media Redundancy = TRUE
2.5.3.2.5.1	(m) Attribute: Redundant Frame ID
2.5.3.2.5.2	(m) Attribute: Redundant Frame Send Offset
2.5.3.2.5.3	(m) Attribute: Redundant Tx Port
2.5.3.2.6	(m) Attribute: DFP
2.5.3.2.7	(c) Constraint: DFP = TRUE
2.5.3.2.7.3	(m) Attribute: Subframe Position
2.5.4	(m) Attribute: Reduction Ratio
2.5.5	(m) Attribute: Phase
2.5.6	(m) Attribute: Sequence

2.6	(c) Constraint:	Protocol Machine = CPM
2.6.1	(m) Attribute:	Rx Option
2.6.1.1	(m) Attribute:	RT Class
2.6.1.2	(c) Constraint:	RT Class = RT_CLASS_3
2.6.1.2.1	(m) Attribute:	Startup Mode
2.6.1.2.2	(m) Attribute:	Media Redundancy
2.6.1.2.3	(c) Constraint:	Media Redundancy = TRUE
2.6.1.2.3.1	(m) Attribute:	Redundant Frame ID
2.6.1.2.3.2	(m) Attribute:	Redundancy Watchdog
2.6.1.2.4	(m) Attribute:	DFP
2.6.1.2.5	(c) Constraint:	DFP = TRUE
2.6.1.2.5.1	(m) Attribute:	Subframe Position
2.6.2	(m) Attribute:	Data Hold Factor

SERVICES:

1	(m) OpsService:	PPM Set Prov Data
2	(m) OpsService:	PPM Set Prov Status
3	(m) OpsService:	PPM Activate
4	(m) OpsService:	PPM Close
5	(m) OpsService:	PPM Start
6	(m) OpsService:	PPM Error
7	(m) OpsService:	CPM Get Cons Data
8	(m) OpsService:	CPM Get Cons Status
9	(m) OpsService:	CPM Set RedRole
10	(m) OpsService:	CPM Activate
11	(m) OpsService:	CPM NoData
12	(m) OpsService:	CPM Stop

6.3.5.2.3 Attributes**CREP**

This key attribute contains the CR's identifier (see AR ASE, class Communication Relationship).

Attribute type: Unsigned32

Protocol machine Parameter

This attribute contains the following attributes to describe the behavior of the protocol machines:

DA

This parameter contains the destination address (i.e. the consumer's address) according to IEEE 802.3.

Attribute type: OctetString[6]

SA

This parameter contains the source address (i.e. the provider's address) according to IEEE 802.3.

Attribute type: OctetString[6]

Frame ID

This attribute contains the identifier of the frame.

Attribute type: Unsigned16

Protocol Machine

This attribute contains the particular protocol machine selected for this CR. The protocol machine is set to PPM (cyclic provider protocol machine), when the device is provider of

the particular CR. Otherwise it is set to CPM (cyclic consumer protocol machine), when the device is consumer of the particular CR.

Attribute type: Unsigned8

Allowed Value: PPM, CPM

VLAN Prio

This conditional attribute is present when the value of the protocol machine attribute is "PPM". It contains the priority of the frame according to IEEE 802.1Q.

Attribute type: Unsigned8

VLAN ID

This conditional attribute is present when the value of the protocol machine attribute is "PPM". It contains the VLAN ID according to IEEE 802.1Q.

Attribute type: Unsigned16

Allowed Value: 0

Tx Option

This conditional attribute is present when the value of the protocol machine attribute is "PPM". It specifies the provider's transmit options and consists of the following attributes:

RT Class

This attribute contains the provider's communication class.

Attribute type: Unsigned32

Allowed Values: RT_CLASS_1, RT_CLASS_2, RT_CLASS_3, RT_CLASS_UDP

NOTE 1 RT_CLASS_2 is usable in the context of legacy startup only.

Frame Send Offset

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It 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

Tx Port

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the transmit port, where the frame shall be sent.

Attribute type: Unsigned8

Allowed values: 0x01 to 0xFF

Startup Mode

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the startup mode for RT_CLASS_3.

Allowed values: Legacy, Advanced

Media Redundancy

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of Media Redundancy with Planned Duplication of frames (MRPD).

Allowed values: FALSE, TRUE

Redundant Frame ID

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the identifier of the redundant frame.

Attribute type: Unsigned16

Redundant Frame Send Offset

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It 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

Redundant Tx Port

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It indicates the port, where the redundant frame shall be sent.

Attribute type: Unsigned8

Allowed values: 0x01 to 0xFF

DFP

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of DFP.

Allowed values: FALSE, TRUE

Subframe Position

This conditional attribute is present when the value of the DFP attribute is "TRUE". It defines the position number of the subframe. The position number shall be strictly increasing within the list of Subframe Data.

Attribute type: Unsigned8

Allowed values: 0x01 to 0x3F for packetizing and unpacking of subframes during transport and end to end; 0x40 to 0x7F for packetizing and unpacking of subframes end to end; 0x00 and 0x80 to 0xFF reserved

Reduction Ratio

This conditional attribute is present when the value of the protocol machine attribute is "PPM". It contains the reduction ratio of the CR. The actual send cycle is determined by Reduction Ratio multiplied with the send clock.

Attribute type: Unsigned16

Allowed Values: see IEC 61158-6-10

Phase

This conditional attribute is present when the value of the protocol machine attribute is "PPM". It indicates in which particular send cycle the data has to be sent. This parameter depends from the parameter Reduction Ratio.

Attribute type: Unsigned16

Allowed values: 1 to selected Reduction Ratio.

Sequence

This conditional attribute is present when the value of the protocol machine attribute is "PPM". It 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

Rx Option

This conditional attribute is present when the value of the protocol machine attribute is "CPM". It specifies the consumer's receive options and consists of the following attributes:

RT Class

This attribute contains the consumer's communication class.

Attribute type: Unsigned32

Allowed Values: RT_CLASS_1, RT_CLASS_2, RT_CLASS_3, RT_CLASS_UDP

NOTE 2 RT_CLASS_2 is usable in the context of legacy startup only.

Startup Mode

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the startup mode of the related AR.

Allowed values: Legacy, Advanced

Media Redundancy

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of MRPD.

Allowed values: FALSE, TRUE

Redundant Frame ID

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the identifier of the redundant frame.

Attribute type: Unsigned16

Redundancy Watchdog Timer

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the value for the redundancy watchdog timer.

Attribute type: Unsigned16

Time Base: 1 s

Allowed values: 0x0001 to 0x0100

Default value: 0x0001

NOTE 3 The redundancy watchdog is meant to detect "static" ring problems (e.g. a cable is pulled). Thus the resolution of the redundancy watchdog timer is relaxed.

DFP

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of DFP.

Allowed values: FALSE, TRUE

Subframe Position

This conditional attribute is present when the value of the DFP attribute is "TRUE". It defines the position number of the subframe. The position number shall be strictly increasing within the list of Subframe Data.

Attribute type: Unsigned8

Allowed values: 0x01 to 0x3F for packetizing and unpacking of subframes during transport and end to end; 0x40 to 0x7F for packetizing and unpacking of subframes end to end; 0x00 and 0x80 to 0xFF reserved

Data Hold Factor

This conditional attribute is present when the value of the protocol machine attribute is "CPM". It 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.

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 be equal to or less than 1,92 s if the value of the AR attribute RT Class is 0x01, 0x02 or 0x03 and equal to or less than 49,152 s if the value of the AR attribute RT Class is 0x04.

Attribute type: Unsigned16

Allowed values: 0x0001 to 0x1E00 for RT Class 0x01, 0x02 or 0x03; 0x0001 to 0xF000 for RT Class 0x04

6.3.5.3 RTC service specification

6.3.5.3.1 PPM Set Prov Data

This local service is used to update the provider's transmit buffer with new data. Table 40 shows the parameters of the service.

Table 40 – PPM Set Prov Data

Parameter name	req	cnf
Argument	M	
CREP	M	
Data	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Data

This parameter contains the data to update the transmit buffer.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.2 PPM Set Prov Status

This local service is used to update the APDU status of the provider's transmit buffer. Table 41 shows the parameters of the service.

Table 41 – PPM Set Prov Status

Parameter name	req	cnf
Argument	M	
CREP	M	
Status	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Status

This parameter contains the APDU status to update for the transmit buffer.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.3 PPM Activate

This local service is used to activate the provider protocol machine (PPM). Table 42 shows the parameters of the service.

Table 42 – PPM Activate

Parameter name	req	cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
VLAN Prio	M	
VLAN ID	M	
Tx Option	M	
RT Class	M	
Frame Send Offset	C	
Tx Port	C	
Startup Mode	C	
Media Redundancy	C	
Redundant Frame ID	C	
Redundant Frame Offset	C	
Redundant Frame Tx Port	C	
DFP	C	
Subframe Position	C	
Reduction Ratio	M	
Phase	M	
Sequence	M	
Default Values	M	
Default Status	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

DA

This parameter contains the destination MAC address to be used for the RTC frame.

SA

This parameter contains the source MAC address to be used for the RTC frame.

Frame ID

This parameter contains the Frame ID to be used for the RTC frame.

VLAN Prio

This parameter contains the VLAN Prio to be used for the RTC frame.

VLAN ID

This parameter contains the VLAN ID to be used for the RTC frame.

Tx Option

This attribute defines the transmit option. It is composed of the following elements:

RT Class

This attribute contains the provider's communication class.

Frame Send Offset

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It contains the relative send offset to the start of the related reduction ratio and phase.

Tx Port

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the transmit port where the frame shall be sent.

Startup Mode

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the startup mode of the related AR.

Media Redundancy

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of MRPD.

Redundant Frame ID

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the identifier of the redundant frame.

Redundant Frame Send Offset

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the relative send offset to the start of the related reduction ratio and phase.

Redundant Tx Port

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It indicates the port, where the redundant frame shall be sent.

DFP

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of DFP.

Subframe Position

This conditional attribute is present when the value of the DFP attribute is "TRUE". It defines the position number of the subframe.

Reduction Ratio

This parameter contains the Reduction Ratio to be used for the RTC frame.

Phase

This parameter contains the Phase to be used for the RTC frame.

Sequence

This parameter contains the Sequence to be used for the RTC frame.

Default Value

This parameter contains the initial data for the transmit buffer.

Default Status

This parameter contains the initial APDU status for the transmit buffer.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.4 PPM Close

This local service is used to close the provider protocol machine (PPM) and thus to stop the transmission. Table 43 shows the parameters of the service.

Table 43 – PPM Close

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		M
CREP		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Result(+)

This parameter indicates that the service request succeeded.

6.3.5.3.5 PPM Start

This local service is used to start the transmission of the provider protocol machine (PPM). Table 44 shows the parameters of the service.

Table 44 – PPM Start

Parameter name	ind
Argument	M
CREP	M

Argument

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

CREP

This parameter contains the CR's local identifier.

6.3.5.3.6 PPM Error

This local service is used to indicate an error during transmission of the provider protocol machine (PPM). Table 45 shows the parameters of the service.

Table 45 – PPM Error

Parameter name	ind
Argument	M
CREP	M
No Data Send	M

Argument

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

CREP

This parameter contains the CR's local identifier.

No Data Send

This parameter indicates the failure of data transmission.

6.3.5.3.7 CPM Get Cons Data

This local service is used to read data from a consumer's receive buffer. Table 46 shows the parameters of the service.

Table 46 – Get Cons Data

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M
Data		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Result(+)

This parameter indicates that the service request succeeded.

Data

This parameter contains the data from the receive buffer.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.8 CPM Get cons status

This local service is used to read the APDU status from the consumer's receive buffer. Table 47 shows the parameters of the service.

Table 47 – CPM Get cons status

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M
Status		M
Recv Counter		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Result(+)

This parameter indicates that the service request succeeded.

Status

This parameter contains the APDU status.

Recv Counter

This parameter contains the receive counter.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.9 CPM Set RedRole

This local service is used to set the redundancy status of the CR. Table 48 shows the parameters of the service.

Table 48 – CPM Set RedRole

Parameter name	req	cnf
Argument	M	
CREP	M	
Red Role	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Red Role

This parameter determines the redundancy role (primary, backup) of the CR.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.10 CPM Activate

This local service is used to activate the consumer protocol machine (CPM). Table 49 shows the parameters of the service.

Table 49 – CPM Activate

Parameter name	req	cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
Rx Option	M	
RT Class	M	
Startup Mode	C	
Media Redundancy	C	
Redundant Frame ID	C	
Redundancy Watchdog Timer	C	
DFP	C	
Subframe Position	C	
Expected Length	M	
Start Ton	M	
Datahold Factor	M	
Default Value	M	
Default Status	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

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

CREP

This parameter contains the CR's local identifier.

DA

This parameter shall be used for the destination MAC address.

SA

This parameter shall be used for the source MAC address.

Frame ID

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

Rx Option

This conditional attribute specifies the consumer's receive options and consists of the following attributes:

RT Class

This attribute contains the consumer's communication class.

Startup Mode

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the startup mode of the related AR.

Media Redundancy

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of MRPD.

Redundant Frame ID

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the identifier of the redundant frame.

Redundancy Watchdog Timer

This conditional attribute is present when the value of the Media Redundancy attribute is set to "TRUE". It contains the value for the redundancy watchdog timer.

DFP

This conditional attribute is present when the value of the RT Class attribute is "RT_CLASS_3". It indicates the usage of DFP.

Subframe Position

This conditional attribute is present when the value of the DFP attribute is "TRUE". It defines the position number of the subframe.

Expected Length

This parameter contains the expected length of the CR.

Start Ton

This parameter shall be set to true if the timeout shall be monitored immediately. Otherwise it starts with the reception of the first frame.

Datahold Factor

This parameter contains the datahold factor.

Default Value

This parameter contains the initial value of the receive buffer's data.

Default Status

This parameter contains the initial value of the receive buffer's APDU status.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned8

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.5.3.11 CPM NoData

This local service is used to indicate that the cyclic consumer protocol machine (CPM) did not receive data within the data hold timer. Table 50 shows the parameters of the service.

Table 50 – CPM NoData

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		M
CREP		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Result(+)

This parameter indicates that the service request succeeded.

6.3.5.3.12 CPM Stop

This local service is used to stop the CPM and thus the reception. Table 51 shows the parameters of the service.

Table 51 – CPM Stop

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		M
CREP		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Result(+)

This parameter indicates that the service request succeeded.

6.3.5.3.13 CPM New Data Indication

This local service is used to indicate an update of the CPM data. Table 52 shows the parameters of the service.

Table 52 – CPM New Data Indication

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		M
CREP		M

Argument

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

CREP

This parameter contains the CR's local identifier.

Result(+)

This parameter indicates that the service request succeeded.

6.3.5.4 Invocation of the RTC object

There is one RTC object for each CR of each application relationship.

6.3.6 Real time acyclic ASE

6.3.6.1 Overview

The real time protocol acyclic (RTA) is a concept of reading and writing acyclic data by means of acknowledged services. It uses IEEE 802.3 or UDP services.

6.3.6.2 RTA class specification

6.3.6.2.1 General

The RTA ASE defines for each CR one RTA object type:

6.3.6.2.2 Template

A RTA object is described by the following template:

ASE: RTA ASE

CLASS: RTA

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

- 1 (m) Key Attribute: CREP
- 2 (m) Attribute: Protocol machine Parameter
- 2.1 (m) Attribute: DA
- 2.2 (m) Attribute: SA
- 2.3 (m) Attribute: Frame ID

2.4	(m) Attribute:	VLAN Prio
2.5	(m) Attribute:	VLAN ID
2.6	(m) Attribute:	RTA Timeout Factor
2.7	(m) Attribute:	M Retry
2.8	(m) Attribute:	Send Seq Num
2.9	(m) Attribute:	Ack Seq Num
2.10	(m) Attribute:	Dst IP
2.11	(m) Attribute:	Src IP
2.12	(m) Attribute:	Transport

SERVICES:

1	(m) OpsService:	APMS Activate
2	(m) OpsService:	APMR Activate
3	(m) OpsService:	APMS A Data
4	(m) OpsService:	APMR A Data
5	(m) OpsService:	APMR Ack
6	(m) OpsService:	APMS Error
7	(m) OpsService:	APMR Error
8	(m) OpsService:	APMS Close
9	(m) OpsService:	APMR Close

6.3.6.2.3 Attributes**CREP**

This key attribute identifies the instance of the protocol machine to which the data belong.

Attribute type: Unsigned32

Protocol machine Parameter

This attribute contains the following attributes to describe the behavior of the protocol machines:

DA

This attribute contains the physical address of the destination device according to IEEE 802

Attribute type: OctetString[6]

SA

This attribute contains the physical address of the device according to IEEE 802.3

Attribute type: OctetString[6]

Frame ID

This attribute contains the identifier of the data within the frame.

Attribute type: Unsigned16

VLAN Prio

This attribute contains the priority of the frame according to IEEE 802.1Q.

Attribute type: Unsigned8

Allowed Values: LOW, HIGH

VLAN ID

This attribute contains the VLAN ID according to IEEE 802.1Q.

Attribute type: Unsigned16

Allowed Value: 0

RTA Timeout Factor

This attribute contains the timeout factor within a sent RTA Data frame has to be Acknowledged (ACK).

Attribute type: Unsigned16

Time Base: 100 ms

Allowed Values: 1 up to 100 mandatory, 101 to $2^{16}-1$ optional

M Retry

This attribute contains the number of retries if no ACK received within RTA Timeout Factor. The value range is from 3 to 15 retries.

Attribute type: Unsigned16

Send Seq Num

This attribute contains the Sender Sequence Number.

Attribute type: Unsigned16

Allowed Values: see IEC 61158-6-10

Ack Seq Num

This attribute contains the Acknowledge Sequence Number.

Attribute type: Unsigned16

Allowed Values: see IEC 61158-6-10

Dst IP

This attribute contains the destination IP address if UDP Services shall be used.

Attribute type: Unsigned32

Src IP

This attribute contains the local IP address if UDP Services shall be used.

Attribute type: Unsigned32

Transport

This attribute contains the requested conveyance.

Attribute type: Unsigned8

Allowed Values: RTC, UDP

6.3.6.3 RTA service specification

6.3.6.3.1 APMS Activate

This local service shall be used to activate the Acyclic Protocol Machine Sender (APMS). Table 53 shows the parameters of the service.

Table 53 – APMS Activate

Parameter name	req	cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
VLAN Prio	M	
VLAN ID	M	
RTA Timeout Factor	M	
M Retry	M	
Transport	M	
Dst IP	O	
Src IP	O	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

The argument shall convey the service specific parameters of the service request. These parameters shall correspond with the ASE attributes.

CREP

This parameter identifies the instance of the protocol machine.

DA

This parameter shall be used for the physical address of the destination device according to IEEE 802.3.

SA

This parameter shall be used for the physical address of the device according to IEEE 802.3.

Frame ID

This parameter shall be used for the Frame ID.

VLAN Prio

This parameter shall contain the priority of the frame according to IEEE 802.1Q.

VLAN ID

This parameter shall contain the VLAN ID according to IEEE 802.1Q.

RTA Timeout Factor

This parameter shall contain the timeout factor.

M Retry

This parameter shall contain the number of retries.

Dst IP

This parameter shall contain the destination IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Src IP

This parameter shall contain the local IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Transport

This parameter shall contain the requested conveyance.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned16

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.6.3.2 APMR Activate

This local service shall be used to activate the Acyclic Protocol Machine Receiver (APMR). Table 54 shows the parameters of the service.

Table 54 – APMR Activate

Parameter name	req	cnf
Argument	M	
CREP	M	
DA	M	
SA	M	
Frame ID	M	
VLAN Prio	M	
VLAN ID	M	
Transport	M	
Dst IP	O	
Src IP	O	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
Error Class		M
Error code		M

Argument

The argument shall convey the service specific parameters of the service request. These parameters shall correspond with the ASE attributes.

CREP

This parameter identifies the instance of the protocol machine.

DA

This parameter shall be used for the physical address of the destination device according to IEEE 802.3.

SA

This parameter shall be used for the physical address of the device according to IEEE 802.3.

Frame ID

This parameter shall be used for the Frame ID.

VLAN Prio

This parameter shall contain the priority of the frame according to IEEE 802.1Q

VLAN ID

This parameter shall contain the VLAN ID according to IEEE 802.1Q.

Transport

This parameter shall contain the requested conveyance.

Dst IP

This parameter shall contain the destination IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Src IP

This parameter shall contain the local IP address if UDP services shall be used. Not used if IEEE 802.3 services shall be used.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Error Class

This parameter selects the error class.

Attribute type: Unsigned16

Allowed Value: CTXT

Error code

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.6.3.3 APMS A Data

This service shall be used to send RTA Data. After a RTA Data frame was sent, it has to be acknowledged within a specified time (based on RTA Timeout Factor). If no ACK arrives the data frame will be resent a specified number of times (M Retry). If no ACK arrived at all an APMS Error will be indicated (TIMEOUT). The communication uses sequence numbers to ensure proper sequence of data. Both communication endpoints have to be setup for communication to ensure proper sequence number setup. Only one service can be handled at

a time. So the service has to be finished before a new one can be issued (INVALID_STATE result shall be returned if not finished). Table 55 shows the parameters of the service.

Table 55 – APMS A Data

Parameter name	req	cnf
Argument	M	
CREP	M	
Data	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
ERRCLS		M
ERRCODE		M

Argument

The argument shall convey the service specific parameters of the service request. These parameters shall correspond with the ASE attributes.

CREP

This parameter identifies the instance of the protocol machine.

Data

This parameter shall be used for RTA-SDU data to be sent. The length is limited by ADPU size.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

ERRCLS

This parameter selects the error class.

Attribute type: Unsigned16

Allowed Value: CTXT

ERRCODE

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.6.3.4 APMR A Data

This local service may be used to receive RTA data. Table 56 shows the parameters of the service.

Table 56 – APMR A Data

Parameter name	ind
Argument	M
CREP	M
Data	M

Argument

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

CREP

This parameter identifies the instance of the protocol machine.

Data

This parameter contains the received RTA-SDU data

Result(+)

This parameter indicates that the service request succeeded.

6.3.6.3.5 APMR Ack

This service shall be used to send a RTA ACK after reception of a RTA Data. After RTA Data was received it has to be acknowledged by sending an ACK. Sending the ACK signals the sender that the Data was successfully received and further data may be send. If the ACK will not be sent, the sender retries and will timeout if no ACK arrives at all. Only one service call can be handled at a time. So the service has to be finished before a new one can be issued (INVALID_STATE result is set in not finished). Table 57 shows the parameters of the service.

Table 57 – APMR Ack

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M
Result(-)		S
CREP		M
ERRCLS		M
ERRCODE		M

Argument

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

CREP

This parameter identifies the instance of the protocol machine.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

ERRCLS

This parameter selects the error class.

Attribute type: Unsigned16

Allowed Value: CTXT

ERRCODE

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: INVALID_STATE

6.3.6.3.6 APMS error

This local service may be used to indicate an APMS error. Table 58 shows the parameters of the service.

Table 58 – APMS Error

Parameter name	ind
Argument	M
CREP	M
ERRCLS	M
ERRCODE	M

Argument

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

CREP

This parameter identifies the instance of the protocol machine.

ERRCLS

This parameter selects the error class.

Attribute type: Unsigned16

Allowed Value: see Table 59

ERRCODE

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: see Table 59

Table 59 – APMS Error ERRCLS/ERRCODE

ERRCLS	ERRCODE	Meaning
PROTOCOL	TIMEOUT	Transmission timeout occurred
PROTOCOL	SEQNUM	Sequence number mismatch
PROTOCOL	LMPM	LMPM error

6.3.6.3.7 APMR Error

This local service may be used to indicate an APMR error. Table 60 shows the parameters of the service.

Table 60 – APMR Error

Parameter name	ind
Argument	M
CREP	M
ERRCLS	M
ERRCODE	M

Argument

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

CREP

This parameter identifies the instance of the protocol machine.

ERRCLS

This parameter selects the error class.

Attribute type: Unsigned16

Allowed Value: see Table 61

ERRCODE

This parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: see Table 61

Table 61 – APMR Error ERRCLS/ERRCODE

ERRCLS	ERRCODE	Meaning
PROTOCOL	SEQNUM	Sequence number of received data mismatch
PROTOCOL	LMPM	LMPM error

6.3.6.3.8 APMS_Close

This local service shall be used to close the APMS. Table 62 shows the parameters of the service.

Table 62 – APMS_Close

Parameter name	req	cnf
Argument	M	
CREP	M	
ErrCode	U	
Result(+)		S
CREP		M

Argument

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

CREP

This parameter identifies the instance of the protocol machine.

ErrCode

This optional parameter selects the error code.

Attribute type: Unsigned16

Allowed Value: See corresponding ASE attribute

Result(+)

This parameter indicates that the service request succeeded.

6.3.6.3.9 APMR_Close

This local service shall be used to close the APMR. Table 63 shows the parameters of the service.

Table 63 – APMR_Close

Parameter name	req	cnf
Argument	M	
CREP	M	
Result(+)		S
CREP		M

Argument

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

CREP

This parameter identifies the instance of the protocol machine.

Result(+)

This parameter indicates that the service request succeeded.

6.3.6.4 Invocation of the RTC object

There is one RTA object for each application relationship.

6.3.7 Discovery and basic configuration ASE

6.3.7.1 Overview

The discovery and basic configuration is an Application Service Element for reading and writing basic network configuration parameters and discovering devices by means of different filter criteria. It uses IEEE 802.3 services.

The Discovery and basic Configuration Protocol (DCP) ASE is focused towards a particular Ethernet interface. Be aware that a particular device may have more than one Ethernet interface. The DCP ASE itself has no knowledge whether several interfaces belong to the same device; other ASE (e.g. simple network management ASE or else Communication Interface Management ASE) provide such means.

Although DCP uses the word “device”, the scope used here is the “interface”.

6.3.7.2 DCP class specification

6.3.7.2.1 Template

A DCP object is described by the following template:

ASE: DCP ASE
CLASS: DCP
CLASS ID: not used
PARENT CLASS: IEEE 802.1AB
ATTRIBUTES:

1	(m)	Key Attribute: Implicit
2	(m)	Attribute: List of Device Options
2.1	(m)	Attribute: Device Option
2.2	(m)	Attribute: Device Suboption
3	(m)	Attribute: IP Option
3.1	(m)	Attribute: MAC Address
3.2	(m)	Attribute: IP Parameter
3.2.1	(m)	Attribute: IP Address
3.2.2	(m)	Attribute: Subnet Mask
3.2.3	(m)	Attribute: Standard Gateway
3.2.4	(o)	Attribute: DNS Server Addresses
4	(m)	Attribute: Device Properties Option
4.1	(m)	Attribute: Name Of Station
4.2	(m)	Attribute: List of Ports
4.2.1	(m)	Attribute: Alias Name Of Station
4.3	(m)	Attribute: Device Vendor
4.4	(m)	Attribute: Device ID
4.5	(m)	Attribute: Device Role
4.6	(o)	Attribute: Device Instance
4.7	(o)	Attribute: OEM Device ID
5	(o)	Attribute: DHCP Option
5.1	(o)	Attribute: Host Name
5.2	(o)	Attribute: Vendor Specific Information
5.3	(o)	Attribute: Server Identifier
5.4	(o)	Attribute: Parameter Request List
5.5	(o)	Attribute: Class Identifier
5.6	(o)	Attribute: DHCP Client Identifier
5.7	(o)	Attribute: Fully Qualified Domain Name
5.8	(o)	Attribute: UUID/GUID-based Client Identifier
5.9	(m)	Attribute: Control DHCP for Address Resolution
6	(o)	Attribute: Device Initiative Option
6.1	(m)	Attribute: Device Initiative
7	(o)	Attribute: Manufacturer Specific Option
7.1	(m)	Attribute: List of Manufacturer Specific Suboptions
7.1.1	(m)	Attribute: Manufacturer Specific Suboption
7.1.2	(m)	Attribute: Manufacturer OUI
7.1.3	(m)	Attribute: Manufacturer Specific String
8	(m)	Attribute: Protocol machine Parameter
8.1	(m)	Attribute: Max Retry Limit
8.2	(m)	Attribute: Set/Get Timeout
8.3	(m)	Attribute: Identify Timeout
8.4	(m)	Attribute: TagControlInformation
8.4.1	(m)	Attribute: VLAN ID
8.4.2	(m)	Attribute: Priority
8.5	(m)	Attribute: Client Hold Time
8.6	(o)	Attribute: Initial Delay
8.5	(o)	Attribute: Number of DCP-Hello-ReqPDUs
8.5	(o)	Attribute: Hello Interval

SERVICES:

1	(m) OpsService: Get
2	(m) OpsService: Set
3	(m) OpsService: Identify
4	(o) OpsService: Hello

6.3.7.2.2 Attributes

Implicit

The key attribute Implicit indicates that the DCP object is implicitly addressed by the service.

List of Device Options

This attribute list shall contain all options and suboptions which are supported by the device. It acts as an inventory over the supported DCP options and suboptions:

Device Option

This attribute contains an option that is supported by the device.

Attribute type: Unsigned8

Allowed values: IP, DEVICE_PROPERTIES, DHCP, CONTROL, DEVICE_INITIATIVE, MANUFACTURER_SPECIFIC_128, ... MANUFACTURER_SPECIFIC_254

Device Suboption

This attribute contains a suboption related to a specific option that is supported by the device.

Attribute type: Unsigned8

Allowed values:

On Device Option = IP:

MAC_ADDRESS, IP_PARAMETER, FULL_IP_SUITE

On Device Option = DEVICE_PROPERTIES:

DEVICE_VENDOR, NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE, DEVICE_OPTIONS, ALIAS_NAME, DEVICE_INSTANCE, OEM_DEVICE_ID

On Device Option = DHCP:

DHCP_PARAMETER

On Device Option = CONTROL:

START, STOP, SIGNAL, FACTORY_RESET, RESET_TO_FACTORY

On Device Option = DEVICE_INITIATIVE:

DEVICE_INITIATIVE_STATE

On Device Option = MANUFACTURER_SPECIFIC_128, ...
MANUFACTURER_SPECIFIC_254:

MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255

IP Options

This attribute consists of the following attributes:

MAC Address

This attribute contains the physical address of the Ethernet interface according to IEEE 802.3 MAC address.

The MAC Address shall be nonvolatile. It may be read by means of the Get service or it may be used within the Identify service as a filter. It is not possible to alter the MAC address by using the Set service.

Attribute type: OctetString[6]

IP Parameter

The IP Parameter shall have the same meaning as the corresponding attributes within the IP suite ASE as defined in 6.3.15. This attribute list contains the following attributes:

IP Address

This attribute contains the IP address of the Ethernet interface according to IETF RFC 791 and IETF RFC 6890.

Attribute type: Unsigned32

Default Value: 0.0.0.0

Subnet Mask

This attribute contains the subnet mask of the Ethernet interface according to IETF RFC 791 and IETF RFC 6890.

Attribute type: Unsigned32

Default Value: 0.0.0.0

Standard Gateway

This attribute contains the IP address of the standard gateway according to IETF RFC 791 and IETF RFC 6890.

Attribute type: Unsigned32

Default Value: 0.0.0.0

DNS Server Addresses

This attribute contains up to 4 DNS server IP addresses. The value "0.0.0.0" indicates no address.

Attribute type: Unsigned32[4]

Default Value: 0.0.0.0

Device Properties Option

This attribute contains the following suboption attributes:

Name Of Station

This attribute contains the name of the interface set by local means or by the Set service. The values "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" shall not be used.

NOTE 1 Be aware that the Name Of Station is in contrast to its misleading name a Name Of Interface, since as stated above the focus of the DCP ASE is a particular Ethernet interface.

Attribute type: VisibleString[240]

Default Value: "" (empty string)

List of Ports

This attribute contains for each of the interface's ports the following attributes.

Alias Name Of Station

This attribute contains an alias name for the interface derived from the IEEE 802.1AB ASE information. For each port at most one alias name exists.

A particular alias name is derived from the peers Port ID and Chassis ID; if exactly one neighbor is detected on that port. Otherwise (i.e. no neighbor or multiple neighbors are detected on this port) no alias name is built for this port.

If the peers Port ID contains the value “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 alias name is built by concatenating the peers Port ID, a “.” and the peers Chassis ID.

If the peers Port ID consists of more than one token (separated by “.”) and the first token contains the value “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 alias name is set to the peers Port ID.

An empty string (i.e. “”) indicates that no alias name exists on this port.

The Alias Name Of Station may only be used as a filter in the Identify service. The services Get, Set and Hello do not offer access to this attribute.

NOTE 2 Example for Alias Name Of Station = “port-001.mill-1.factory3.org”.

Attribute type: VisibleString[255]

Device Role

This attribute contains a bit map to indicate the device roles on this Ethernet interface.

If it may act as IO device, indicate this role using the bits IO_DEVICE and IO_MULTIDEVICE. IO_DEVICE indicates thereby that at most one IO device instance is active, IO_MULTIDEVICE that multiple IO device instances may be active in parallel.

IO_CONTROLLER indicates that it may act as IO controller.

IO_SUPERVISOR indicates that it may act as IO supervisor.

Attribute type: Unsigned8

Allowed values: IO_DEVICE, IO_CONTROLLER, IO_MULTIDEVICE, IO_SUPERVISOR

Device Vendor

This attribute contains a vendor specific string. It may be the type of the device or an ordering number.

Attribute type: VisibleString

Device ID

This attribute contains the Device Ident Number as described in 7.4.1.

Attribute type: Unsigned32

Device Instance

This attribute contains the instance or node number of the IO device's object UUID.

Attribute type: Unsigned16

OEM Device ID

This optional attribute contains the OEM Device Ident Number.

Devices built by using separately handable OEM parts (e.g. pluggable interface submodules) provide their own Device ID as usual. The Device ID of the OEM part may additionally be offered using OEM Device ID. In this case the OEM shall also provide the I&M5 Data as described in the I&M Data class.

Attribute type: Unsigned32

DHCP Option

This optional attribute consists of the following suboption attributes.

Most of the DHCP suboptions are optional, meaning that a particular suboption may or may not be supported by a particular device. If a particular attribute is supported it may be in the state activated or deactivated. In the state activated a value is provided for the attribute by the get service indicated by having a length different from 0, otherwise the length of the returned attribute is 0. A suboption is deactivated by setting the length of the suboption attribute to 0 using the set service. Activated suboptions are contained in the DHCP protocol frames.

Host Name

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

Vendor Specific Information

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

Server Identifier

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

Parameter Request List

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

Class Identifier

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

DHCP Client Identifier

This optional attribute contains the client identifier for the usage of Dynamic Host Configuration Protocol (DHCP)

Attribute type: see IETF RFC 2132

Fully Qualified Domain Name

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

UUID/GUID-based Client Identifier

This optional attribute contains the value according to IETF RFC 2132.

Attribute type: see IETF RFC 2132

Control DHCP for Address Resolution

This mandatory attribute allows controlling the general behavior of DHCP.

Attribute type: Unsigned8

Allowed Values: DONT_USE_DHCP, DONT_USE_AND_RESET_DHCP_OPTIONS, USE_DHCP

Device Initiative Option

This optional attribute contains the following suboption attributes:

Device Initiative

This attribute contains the value to enable or disable the issuing of the Hello service.

Attribute type: Unsigned16

Allowed values: ON, OFF

Manufacturer Specific Option

This optional attribute contains the following suboption attributes:

List of Manufacturer Specific Suboptions

This attribute list contains the following attributes:

Manufacturer Specific Suboption

This attribute contains the manufacturer specific value.

Attribute type: Unsigned8

Manufacturer OUI

This attribute contains the organizational unique identifier to identify the vendor.

Attribute type: OctetString[3]

Manufacturer Specific String

This attribute contains the individual information.

Attribute type: OctetString

Protocol machine Parameter

This attribute contains the following attributes to describe the behavior of the protocol machines:

Max Retry Limit

This attribute contains the maximum limit to repeat a service in case no answer has been received by the client.

Attribute type: Unsigned8

Default Value: 2

Allowed Values: 0 to 15

Set/Get Timeout

This attribute contains the maximum value in seconds, which a client waits for a unicast response.

Attribute type: Unsigned16

Default Value: 1

Allowed Values: 1 to 30

Identify Timeout

This attribute contains the maximum value in milliseconds, which a client waits for all possible identify responses. It shall be greater than the calculated Response Delay Time using the Response Delay Factor from the request. It shall be 400 ms if the Response Delay Factor equals one and rounded up to a full second plus 1 s otherwise.

Attribute type: Unsigned16

Allowed Values: 400 to 65 000

Tag Control Information

This attribute contains the following attributes:

VLAN ID

This attribute contains the VLAN ID according to IEEE 802.1Q.

Attribute type: Unsigned16

Allowed Value: 0

Priority

This attribute contains the priority tag of the frame according to IEEE 802.1Q.

Attribute type: Unsigned8

Allowed Value: 0

Client Hold Time

This attribute contains the maximum value in seconds, which a server shall only allow Get or Set services from the last active client. Service requests from other server shall be ignored before.

Attribute type: Unsigned16

Allowed Value: 3

Initial Delay

This parameter contains the delay time for the first DCP-Hello-ReqPDU. This parameter can be modified by CIM as FSU parameter.

Time base: 1 ms

Parameter type: Unsigned32

Allowed Values: 0, 50, 100, 500, 1 000

Number of DCP-Hello-ReqPDUs

This parameter contains the number of DCP-Hello-ReqPDUs that shall be issued with the service request. This parameter can be modified by CIM as FSU parameter.

Parameter type: Unsigned32

Allowed Values: 1 to 15

Default Values: 3

Hello Interval

This parameter contains the time interval between transmitting the first and each subsequent DCP-Hello-ReqPDU. This parameter can be modified by CIM as FSU parameter.

Parameter type: Unsigned16

Allowed Values: 30_MILLISECONDS, 50_MILLISECONDS, 100_MILLISECONDS, 300_MILLISECONDS, 500_MILLISECONDS, 1_SECONDS

Default Values: 30_MILLISECONDS

6.3.7.3 DCP service specification**6.3.7.3.1 Get**

The Get service is used to read one or more ASE attributes by means of the client server communication model. The attribute group shall be addressed by means of an option. The subgroup shall be addressed by means of a suboption.

The server shall respond with the individual status “not supported” if optional options or suboptions are not available.

If the Get service requests multiple Option/Suboption and the response would result in too much data exceeding the available size of the response PDU, it shall skip starting with the first non-fitting all following Options/Suboptions silently without notice. The user of the Get service shall check which requested Options/Suboptions are responded.

Table 64 shows the parameter of the service.

Table 64 – Get

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
DA	M	M(=)		
List Of Requests	M	M(=)		
Type	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Result(+)			S	S(=)
List Of Responses			M	M(=)
Type			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
DCP Block Length			M	M(=)
Block Info			C	C(=)
Value			C	C(=)
OriginalType			C	C(=)
Option			M	M(=)
Suboption			M	M(=)
Block Error			C	C(=)
AddDataValue			C	C(=)
Result(-)			S	S(=)
ERRCLS			M	M(=)
ERRCODE			M	M(=)

Argument

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

DA

This parameter shall contain the individual IEEE 802.3 MAC address of the server.

List Of Requests

This list contains all requested options with their related suboptions. These parameters shall correspond with the ASE attributes.

Type

This parameter contains the information to be read from the device. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: IP, DEVICE_PROPERTIES, DHCP, DEVICE_INITIATIVE, MANUFACTURER_SPECIFIC_128, ... MANUFACTURER_SPECIFIC_254

Suboption

This parameter describes the type's suboption.

Allowed values:

On Option = IP:

MAC_ADDRESS, IP_PARAMETER, FULL_IP_SUITE

On Option = DEVICE_PROPERTIES:

DEVICE_VENDOR, NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE, DEVICE_OPTIONS, DEVICE_INSTANCE, OEM_DEVICE_ID

On Option = DHCP:

DHCP_PARAMETER

On Option = DEVICE_INITIATIVE:

DEVICE_INITIATIVE_STATE

On Option = MANUFACTURER_SPECIFIC_128, ...
MANUFACTURER_SPECIFIC_254:

MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255

Result(+)

This parameter indicates that the service request succeeded. It shall contain the requested values or an error code if an option/suboption is not available. Omitted elements due to the restricted size of the response are not indicated. The element order in the response is arbitrary.

List Of Responses

This parameter shall contain values of requested options/suboptions.

Type

This parameter identifies the responded information, which was requested from the device

It consists of the following attributes:

Option

This parameter contains a requested option. Additionally it may contain CONTROL, if there was an error retrieving the information.

Allowed values: CONTROL, IP, DEVICE_PROPERTIES, DHCP, DEVICE_INITIATIVE, MANUFACTURER_SPECIFIC_128, ...
MANUFACTURER_SPECIFIC_254

Suboption

This parameter contains a suboption requested from the device. Additionally it may contain RESPONSE, if there was an error retrieving the information.

Allowed values:

On Option = CONTROL:

RESPONSE

On Option = IP:

MAC_ADDRESS, IP_PARAMETER, FULL_IP_SUITE

On Option = DEVICE_PROPERTIES:

DEVICE_VENDOR, NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE,
DEVICE_OPTIONS, DEVICE_INSTANCE, OEM_DEVICE_ID

On Option = DHCP:

DHCP_PARAMETER

On Option = DEVICE_INITIATIVE:

DEVICE_INITIATIVE_STATE

On Option = MANUFACTURER_SPECIFIC:

MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255

DCP Block Length

This parameter contains the number of octets of the following data for this option/suboption.

Block Info

This conditional parameter is present only if option / suboption are unequal to CONTROL / RESPONSE.

Allowed values:

On Option = IP, Suboption = IP_PARAMETER:

NO_IP – No IP parameter set

IP_VIA_SET – IP Parameter via Set service received,

IP_VIA_SET_DUP – IP Parameter via Set service received, IP address conflict detected,

IP_VIA_DHCP – IP Parameter via DHCP service received,

IP_VIA_DHCP_DUP – IP Parameter via DHCP service received, , IP address conflict detected

All other Option & Suboption

NO_BLOCK_INFO – No additional block information

Value

This conditional parameter is present only if option / suboption are unequal to CONTROL / RESPONSE.

This parameter contains the value of the requested ASE attribute.

Allowed values:

On Option = IP, Suboption = MAC_ADDRESS:

Value is set to the value of ASE attribute IP Options / MAC Address.

On Option = IP, Suboption = IP_PARAMETER:

Value is set to parts of the value of ASE attribute IP Options / IP Parameter, restricted to IP address, Subnet Mask, and Standard Gateway.

On Option = IP, Suboption = FULL_IP_SUITE:

Value is set to the value of ASE attribute IP Options / IP Parameter, i.e. IP address, Subnet Mask, Standard Gateway, and DNS sever addresses.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_VENDOR:

Value is set to the value of ASE attribute Device Properties Options / Device Vendor.

On Option = DEVICE_PROPERTIES, Suboption = NAME_OF_STATION:

Value is set to the value of ASE attribute Device Properties Options / Name Of Station.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ID:

Value is set to the value of ASE attribute Device Properties Options / Device ID.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ROLE:

Value is set to the value of ASE attribute Device Properties Options / Device Role.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_OPTIONS:

The Value is set to a list containing the optional options / suboptions supported by the device.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_INSTANCE:

Value is set to the value of ASE attribute Device Properties Options / Device Instance.

On Option = DEVICE_PROPERTIES, Suboption = OEM_DEVICE_ID:

Value is set to the value of ASE attribute Device Properties Options / OEM Device ID.

On Option = DHCP, Suboption = DHCP_PARAMETER :

Value is set to the value of ASE attribute DHCP Option.

On Option = DEVICE_INITIATIVE, Suboption = DEVICE_INITIATIVE_STATE:

Value is set to the value of ASE attribute Device Initiative Option / Device Initiative.

On Option = MANUFACTURER_SPECIFIC, Suboption = MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255:

Value is set to the value of ASE attribute Manufacturer Specific Option / Manufacturer Specific Suboption.

OriginalType

This conditional parameter is present only if option / suboption are equal to CONTROL / RESPONSE. It contains the original requested type.

Block Error

This conditional parameter is present only if option / suboption are equal to CONTROL / RESPONSE.

This parameter contains the status of the parameter Data.

Allowed values:

OPTION_NOT_SUPPORTED – requested Option is not supported

SUBOPTION_NOT_SUPPORTED – requested suboption is not supported

RESOURCE_ERROR – temporary resource error within the server

AddDataValue

This conditional and optional parameter is present only if option / suboption are equal to CONTROL / RESPONSE.

It may contain additional specific data in a vendor specific format.

Result(-)

This parameter indicates that the service request failed in general for local reasons.

ERRCLS

The parameter ERRCLS contains the error class of the specific error.

Attribute type: Unsigned16

Allowed Values: CTXT – local context invalid, PROTOCOL – transmission error

ERRCODE

The parameter ERRCODE contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: INVALID_STATE if ERRCLS = CTX, LMPM if ERRCLS = PROTOCOL (local error during transmission), TIMEOUT if ERRCLS = PROTOCOL (remote station was not responding)

6.3.7.3.2 Set

The Set service is used to write one or more ASE attributes by means of the client server communication model. The attribute group shall be addressed by means of an option. The subgroup shall be addressed by means of a suboption.

Furthermore, special control commands control the behavior of the server. The commands Start Transaction and Stop Transaction shall be used to group a sequence of set service and the server shall be locked for a client in this case. Locking is an optional feature in case the server supports transactions. The command Signal allows with its suboption FlashOnce to identify the addressed hardware visually. The command Reset To Factory Settings allows with its block qualifier to reset in particular the application data, communication parameters, engineering parameters, all parameters, the complete device, or the firmware. Note that this command modifies the persistency defined in the Communication Interface Management ASE; see 7.3.3.2.6.2 for details.

The server shall respond with the individual status “not supported” if optional options or suboptions are not available. The element order is arbitrary. Table 65 shows the parameter of the service.

Requesting a Set service containing more “blocks, to be set” than could be acknowledged in the response is malformed. The client shall take care of that.

Table 65 – Set

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
DA	M	M(=)		
Start Transaction	C	C(=)		
Type	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Block Qualifier	M	M(=)		
List Of Requests	M	M(=)		
Type	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
DCP Block Length	M	M(=)		
Block Qualifier	M	M(=)		
Value	M	M(=)		
Stop Transaction	C	C(=)		
Type	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
Block Qualifier	M	M(=)		
Result(+)			S	S(=)
List Of Responses			M	M(=)
Option Control			M	M(=)
Suboption Response			M	M(=)
DCP Block Length			M	M(=)
Original Type			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
Block Error			M	M(=)
Add Data Value			C	C(=)
Result(-)			S	S
ERRCLS			M	M
ERRCODE			M	M

Argument

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

DA

This parameter shall contain the individual IEEE 802.3 MAC address of the server.

Start Transaction

This optional parameter shall mark the start of a transaction to bracket a set of Set services.

If the server supports transactions, it shall lock the ASE attributes for the requesting client until the Stop Transaction is indicated or the Set/Get Timeout occurs. Otherwise the parameter shall be ignored.

Type

This parameter contains the type for start transaction. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: CONTROL

Suboption

This parameter describes the type's suboption.

Allowed values: START

Block Qualifier

This parameter contains further information for the data to be set to the server.

Allowed values: 0

List Of Requests

This list contains all requested options with their related suboptions. These parameters shall correspond with the ASE attributes.

The list shall contain one request only, if option is set to CONTROL and suboption is set to FACTORY_RESET or RESET_FACTORY_SETTING.

Type

This parameter contains the information to be read from the device. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: CONTROL, IP, DEVICE_PROPERTIES, DHCP, MANUFACTURER_SPECIFIC_128, ... MANUFACTURER_SPECIFIC_254

Suboption

This parameter describes the type's suboption.

Allowed values:

On Option = IP:

IP_PARAMETER, FULL_IP_SUITE

On Option = DEVICE_PROPERTIES:

NAME_OF_STATION

On Option = DHCP:

DHCP_PARAMETER

On Option = MANUFACTURER_SPECIFIC_128, ...
MANUFACTURER_SPECIFIC_254:

MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255

On Option = CONTROL:

SIGNAL, FACTORY_RESET, RESET_FACTORY_SETTINGS

DCP Block Length

This parameter contains the number of octets of the following data for this option/suboption.

Block Qualifier

This parameter contains further information for the data to be set to the server.

Allowed Values for Option IP, DEVICE_PROPERTIES, DHCP, and MANUFACTURER_SPECIFIC:

USE_TEMPORARY – Store the value of the option/suboption and its attribute “temporary” in the DCP ASE and use the value subsequently. The value shall be reset to its default value after a power off / power on cycle.

SAVE_PERMANENT – Store the value of the option/suboption and its attribute “permanent” in the DCP ASE and use the value subsequently. The value shall still be available after a power off / power on cycle.

Allowed Values for Option CONTROL, Suboption RESET_FACTORY_SETTINGS:

RESET_APPLICATION_DATA

Reset data which has been stored permanent in submodules and modules to factory values.

Manufacturer specific Record-Data shall be set to factory values.

All I&M Data shall be set to factory values, excluding I&M0, I&M4, I&M5. Furthermore the iParameters including the access password should not be reset.

The support of this suboption is recommended.

RESET_COMMUNICATION_PARAMETER

All parameters active for the interface or the ports and the ARs shall be set to the default values and if permanently stored reset.

This mode is intended to set the addressed communication interface of a device into a state which almost similar to the “out of the box” state.

In particular these are:

NameOfStation shall be set to “” (empty string).

IP suite parameter shall be set to 0.0.0.0

DHCP parameters (if available) shall be reset to factory values.

All PDev parameters (PD IR Data, PD Port Data Adjust, PD Interface MRP Data Adjust...) shall be set to factory values.

The support of this suboption is mandatory.

RESET_ENGINEERING_PARAMETER

Reset engineering parameters which have been stored permanent in the IOC or IOD to factory values.

If a node is able to switch its functionality from IOC to IOD and vice versa via engineering system, the factory functionality shall be activated.

An application program loaded by an engineering system shall be reset (or removed).

A configuration loaded by an engineering system shall be reset (or removed).

The support of this suboption is recommended.

If an IOC or IOD has not such kind of parameters it shall respond with success.

RESET_ALL_STORED_DATA

Resets all stored data in the IOD or IOC to its factory values.

This covers all data related to RESET_APPLICATION_DATA, RESET_COMMUNICATION_PARAMETER and RESET_ENGINEERING_PARAMETER (if the particular service is implemented).

If a node is able to switch its functionality from IOC to IOD and vice versa via engineering system, the factory functionality shall be activated.

The support of this suboption is recommended.

RESET_DEVICE

Resets all stored data in the IOD or IOC to its factory values.

The service has the same behavior as RESET_ALL_STORED_DATA applied to all interfaces.

This service shall reset the communication parameters of all interfaces of the device and should reset all stored data of the device.

This mode is intended to set the device into a state which is similar to the “out of the box” state.

The support of this suboption is recommended.

RESET_AND_RESTORE_DATA

Reset installed software revisions to factory images.

The support of this suboption is optional.

Allowed Values for all other Options and Suboptions:

Only the value “0” is allowed.

Value

This parameter contains the value of the ASE attribute to set.

Allowed values:

On Option = IP, Suboption = IP_PARAMETER:

Contains value to set ASE attribute IP Options / IP Parameter to, restricted to IP address, Subnet Mask, and Standard Gateway.

On Option = IP, Suboption = FULL_IP_SUITE:

Contains value to set ASE attribute IP Options / IP Parameter to.

On Option = DEVICE_PROPERTIES, Suboption = NAME_OF_STATION:

Contains value to set ASE attribute Device Properties Options / Name Of Station to.

On Option = DHCP, Suboption = DHCP_PARAMETER :

Contains value to set ASE attribute DHCP Option to.

On Option = MANUFACTURER_SPECIFIC, Suboption = MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255:

Contains value is set ASE attribute Manufacturer Specific Option / Manufacturer Specific Suboption to.

On Option = CONTROL, Suboption = FLASH_ONCE:

Contains value to initiate flashing an LED (e.g. the Ethernet LINK LED) or an alternative signalization with a duration of 3 s.

Allowed value: 0x0100

Stop Transaction

This optional parameter shall mark the end of a sequence of different Set services.

If the server supports transactions, the server shall unlock the ASE attributes. Otherwise the parameter shall be ignored.

Type

This parameter contains the type for stop transaction. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: CONTROL

Suboption

This parameter describes the type's suboption.

Allowed values: STOP

Block Qualifier

This parameter contains further information for the data to be set to the server.

Allowed values: 0

Result(+)

This parameter indicates that the service request succeeded.

List Of Responses

This parameter contains the return values of the write operation of the requested option/suboption.

Option

This parameter is set to CONTROL.

Suboption

This parameter is set to RESPONSE.

DCP Block Length

This parameter contains the number of octets of the subsequent data.

Original Type

This parameter contains the original type of the service request.

Option

This parameter contains the requested original option.

Suboption

This parameter contains the requested original suboption.

Block Error

This parameter contains the result of the set service for the requested option/suboption.

Allowed values:

NO_ERROR – requested data has been set,
 OPTION_NOT_SUPPORTED – requested optional Option is not supported,
 SUBOPTION_NOT_SUPPORTED – requested optional suboption is not supported,
 SET_NOT_POSSIBLE – set not possible for local reasons,
 RESOURCE_ERROR – temporary resource error within server,
 SET_NOT_POSSIBLE_LOCAL – set not possible for local reasons,
 IN_OPERATION – set not possible because of operating application

AddDataValue

This optional parameter contains additional specific data in a vendor specific format.

Result(-)

This parameter indicates that the service request failed in general for local reasons.

ERRCLS

The parameter ERRCLS contains the error class of the specific error.

Attribute type: Unsigned16

Allowed Values: CTXT – local context invalid, PROTOCOL – transmission error

ERRCODE

The parameter ERRCODE contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: INVALID_STATE if ERRCLS = CTX, LMPM if ERRCLS = PROTOCOL (local error during transmission), TIMEOUT if ERRCLS = PROTOCOL (remote station was not responding)

6.3.7.3.3 Local Set Command

The local Set Command service is used to indicate a Set service to the FAL.

The service passes the arguments of the Set service; thus for further description regarding the arguments see 6.3.7.3.2.

Table 66 shows the parameter of the service.

Table 66 – Local Set Command

Parameter name	req	cnf
Argument	M	
DA	M	
Start Transaction	C	
Type	M	
Option	M	
Suboption	M	
Block Qualifier	M	
List Of Requests	M	
Type	M	
Option	M	

Parameter name	req	cnf
Suboption	M	
DCP Block Length	M	
Block Qualifier	M	
Value	M	
Stop Transaction	C	
Type	M	
Option	M	
Suboption	M	
Block Qualifier	M	
Result(+)		S
List Of Responses		M
Option Control		M
Suboption Response		M
DCP Block Length		M
Original Type		M
Option		M
Suboption		M
Block Error		M
Add Data Value		C
Result(-)		S
ERRCLS		M
ERRCODE		M

6.3.7.3.4 Identify

The Identify service can be used to discover devices on the network.

The service request shall be sent to a well-known multicast IEEE 802.3 MAC address that is defined by the DCP protocol (see IEC 61158-6-10). The Identify confirmation shall contain all responses received within the Identify Timeout interval. The responding devices are identified by their IEEE 802.3 source MAC address. The confirmation contains an empty list, if no device responded within the time interval.

The service allows specifying either “all”, meaning all devices supporting the Identify service shall respond, or a list of filters, meaning only those devices that fulfill all requested filters shall respond.

The response of a particular server shall contain the following ASE attributes:

- All options and suboptions which have been requested as a filter,
- Device Properties Option / Name Of Station,
- IP option / IP Parameter,
- Device Properties Option / Device ID,
- Device Properties Option / Device Vendor,
- Device Properties Option / Device Options,
- Device Properties Option / Device Role,

- Device Properties Option / Device Instance (optional),
- Device Properties Option / OEM Device ID (optional),
- Device Initiative Option / Device Initiative (optional).

The sequence of data blocks may be arbitrary. Table 67 shows the parameter of the service.

Table 67 – Identify

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
Response Delay Factor	M	M(=)		
DA	M	M(=)		
List Of Filter	S	S(=)		
Type	M	M(=)		
Option	M	M(=)		
Suboption	M	M(=)		
DCP Block Length	M	M(=)		
Filter	M	M(=)		
Result(+)			S	S(=)
List of Devices			C	C(=)
SA				M
List of Responses			M	M(=)
Type			M	M(=)
Option			M	M(=)
Suboption			M	M(=)
DCP Block Length			M	M(=)
Block Info			M	M(=)
Value			C	C(=)
Result(-)			S	S(=)
ERRCLS			M	M(=)
ERRCODE			M	M(=)

Argument

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

DA

This parameter shall contain the well-known multicast IEEE 802.3 MAC address.

Response Delay Factor

This parameter contains the factor for the server to calculate the spread value to delay the response if the List Of Filter criteria are fulfilled. The time base is 10 ms. The calculated delay of the server shall be between 10 ms and 64 s.

Parameter type: Unsigned16

Allowed Values: 1 to 6 400

Default Values: 1 (IO controller)

List Of Filter

This conditional parameter indicates that options/suboptions are used to decide whether the device shall respond. The list shall be used to filter the request on the server side. More than one filter shall be handled with an AND operation at the server side.

NOTE AND operation means that a server only responds to an identify request within the response delay time if **all** filter criteria match with currently used attribute values.

Type

This parameter contains the type of the filter. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: ALL, IP, DEVICE_PROPERTIES, DHCP, DEVICE_INITIATIVE, MANUFACTURER_SPECIFIC_128, ... MANUFACTURER_SPECIFIC_254

Suboption

This parameter describes the type's suboption.

Allowed values:

On Option = ALL:

ALL

If ALL / ALL is specified, there shall be only one filter.

On Option = IP:

MAC_ADDRESS, IP_PARAMETER

On Option = DEVICE_PROPERTIES:

DEVICE_VENDOR, NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE, DEVICE_OPTIONS, ALIAS_NAME, OEM_DEVICE_ID

On Option = DHCP:

DHCP_PARAMETER

On Option = MANUFACTURER_SPECIFIC_128 ..
MANUFACTURER_SPECIFIC_254:

MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255

DCP Block Length

This parameter contains the length of the filter.

Filter

This parameter contains the filter for the content of the appropriate ASE attribute identified by Type.

Allowed values:

On Option = ALL, Suboption = ALL:

Filter is not provided, since all devices shall reply.

On Option = IP, Suboption = MAC_ADDRESS:

Contains filter for content of ASE attribute IP Options / MAC address.

On Option = IP, Suboption = IP_PARAMETER:

Contains filter for content of ASE attribute IP Options / IP Parameter, restricted to IP address, Subnet Mask, and Standard Gateway.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_VENDOR:

Contains filter for content of ASE attribute Device Properties Options / Device Vendor.

On Option = DEVICE_PROPERTIES, Suboption = NAME_OF_STATION:

Contains filter for content of ASE attribute Device Properties Options / Name Of Station.

If the filter DEVICE_PROPERTIES / NAME_OF_STATION is used, it shall be the first filter in the List of Filters. In this case the filter DEVICE_PROPERTIES / ALIAS_NAME shall not be used.

On Option = DEVICE_PROPERTIES, Suboption = ALIAS_NAME:

Contains filter for content of ASE attribute Device Properties Options / Alias Name Of Station.

If the filter DEVICE_PROPERTIES / ALIAS_NAME is used, it shall be the first filter in the List of Filters. In this case the filter DEVICE_PROPERTIES / NAME_OF_STATION shall not be used.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ID:

Contains filter for content of ASE attribute Device Properties Options / Device ID.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ROLE:

Contains filter for content of ASE attribute Device Properties Options / Device Role.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_OPTIONS:

Contains filter for content of ASE attribute Device Options.

On Option = DEVICE_PROPERTIES, Suboption = OEM_DEVICE_ID:

Contains filter for content of ASE attribute Device Properties Options / OEM Device ID.

On Option = DHCP, Suboption = DHCP_PARAMETER :

Contains filter for content of ASE attribute DHCP Option.

On Option = MANUFACTURER_SPECIFIC, Suboption = MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255:

Contains filter for content of ASE attribute Manufacturer Specific Option / Manufacturer Specific Suboption.

Result(+)

This parameter indicates that the service request succeeded. It shall contain the requested values.

List of Devices

This parameter contains the list of devices returned from the identification.

SA

This parameter contains the individual IEEE 802.3 MAC address of a device, which has responded to the Identify service.

List of Responses

This parameter contains the responded data of the device specified by SA.

It consists of the following elements:

Type

This parameter contains the type of the filter. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: IP, DEVICE_PROPERTIES, DHCP, DEVICE_INITIATIVE, MANUFACTURER_SPECIFIC_128, ... MANUFACTURER_SPECIFIC_254

Suboption

This parameter describes the type's suboption.

Allowed values:

On Option = IP:

MAC_ADDRESS, IP_PARAMETER

On Option = DEVICE_PROPERTIES:

DEVICE_VENDOR, NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE, DEVICE_OPTIONS, ALIAS_NAME, OEM_DEVICE_ID

On Option = DHCP:

DHCP_PARAMETER

On Option = MANUFACTURER_SPECIFIC_128 .. MANUFACTURER_SPECIFIC_254:

MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255

DCP Block Length

This parameter contains the length of the returned data.

Block Info

This parameter contains additional information to the returned data.

Allowed values:

On Option = IP, Suboption = IP_PARAMETER:

NO_IP – No IP parameter set

IP_VIA_SET – IP Parameter via Set service received,

IP_VIA_SET_DUP – IP Parameter via Set service received, IP address conflict detected,

IP_VIA_DHCP – IP Parameter via DHCP service received,

IP_VIA_DHCP_DUP – IP Parameter via DHCP service received, IP address conflict detected

All other Option & Suboption

NO_BLOCK_INFO – No additional block information

Value

This parameter contains the value.

Allowed values:

On Option = IP, Suboption = MAC_ADDRESS:

Value is set to the value of ASE attribute IP Options / MAC Address.

Included in the list of responses only, if requested as a filter.

On Option = IP, Suboption = IP_PARAMETER:

Value is set to parts of the value of ASE attribute IP Options / IP Parameter, restricted to IP address, Subnet Mask, and Standard Gateway.

Always included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_VENDOR:

Value is set to the value of ASE attribute Device Properties Options / Device Vendor.

Always included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = NAME_OF_STATION:

Value is set to the value of ASE attribute Device Properties Options / Name Of Station.

Always included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ID:

Value is set to the value of ASE attribute Device Properties Options / Device ID.

Always included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ROLE:

Value is set to the value of ASE attribute Device Properties Options / Device Role.

Always included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_OPTIONS:

Value is set to a list containing the mandatory options / suboptions plus those optional options / suboptions supported by the device.

Always included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = ALIAS_NAME:

Value is set to the value of the ASE attribute Device Properties Options / Alias Name. Only the requested alias name is responded; further alias names are not responded.

Included in the list of responses only, if requested as a filter.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_INSTANCE:

Value is set to the value of ASE attribute Device Properties Options / Device Instance.

Optionally included in the list of responses.

On Option = DEVICE_PROPERTIES, Suboption = OEM_DEVICE_ID:

Value is set to the value of ASE attribute Device Properties Options / OEM Device ID.

Optionally included in the list of responses.

On Option = DHCP, Suboption = DHCP_PARAMETER :

Value is set to the value of ASE attribute DHCP Option.

Included in the list of responses only, if requested as a filter.

On Option = DEVICE_INITIATIVE, Suboption = DEVICE_INITIATIVE_STATE:

Value is set to the value of ASE attribute Device Initiative Option / Device Initiative.

Optionally included in the list of responses.

On Option = MANUFACTURER_SPECIFIC, Suboption = MANUFACTURER_SPECIFIC_0, ... MANUFACTURER_SPECIFIC_255:

Value is set to the value of ASE attribute Manufacturer Specific Option / Manufacturer Specific Suboption.

Included in the list of responses only, if requested as a filter.

Result(-)

This parameter indicates that the service request failed in general for local reasons.

ERRCLS

The parameter ERRCLS contains the error class of the specific error.

Attribute type: Unsigned16

Allowed Values: CTXT – local context invalid, PROTOCOL – transmission error

ERRCODE

The parameter ERRCODE contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: INVALID_STATE if ERRCLS = CTX, LMPM if ERRCLS = PROTOCOL (local error during transmission), TIMEOUT if ERRCLS = PROTOCOL (remote station was not responding)

6.3.7.3.5 Hello

The Hello service is used to announce the presence of a device in a network. The service shall be sent to a well-known multicast IEEE 802.3 MAC address that is defined by the DCP protocol (see IEC 61158-6-10).

The service is initiated by the device after the device was powered up and is “communication ready”; delayed by the time as specified by the ASE attribute Initial Delay. The service is initiated multiple times, as specified by the ASE attribute Number of DCP-Hello-ReqPDUs, whereas the ASE attribute Hello Interval defines the time period between each of the hello requests.

The hello request of a particular server shall contain the following ASE attributes:

- Device Properties Option / Name Of Station,
- IP option / IP Parameter,

- Device Properties Option / Device ID,
- Device Properties Option / Device Role,
- Device Initiative Option / Device Initiative,
- Device Properties Option / Device Options (optional),
- Device Properties Option / Device Instance (optional).

Table 68 shows the parameter of the service.

Table 68 – Hello

Parameter name	req	ind
Argument	M	M(=)
List of Data	M	M(=)
Type	M	M(=)
Option	M	M(=)
Suboption	M	M(=)
DCP Block Length	M	M(=)
Block Info	M	M(=)
Value	C	C(=)

Argument

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

List Of Data

This parameter list consists of the following parameters.

Type

This parameter contains the information to be read from the device. It consists of the following attributes:

Option

This parameter describes the type's option.

Allowed values: IP, DEVICE_PROPERTIES, DEVICE_INITIATIVE

Suboption

This parameter describes the type's suboption.

Allowed values:

On Option = IP:

IP_PARAMETER

On Option = DEVICE_PROPERTIES:

NAME_OF_STATION, DEVICE_ID, DEVICE_ROLE, DEVICE_OPTIONS,
DEVICE_INSTANCE

On Option = DEVICE_INITIATIVE:

DEVICE_INITIATIVE_STATE

DCP Block Length

This parameter contains the number of octets of the following data for this option/suboption.

Block Info

This parameter contains additional information for the data.

Allowed values:

On Option = IP, Suboption = IP_PARAMETER:

NO_IP – No IP parameter set

IP_VIA_SET – IP Parameter via Set service received,

IP_VIA_SET_DUP – IP Parameter via Set service received, IP address conflict detected,

IP_VIA_DHCP – IP Parameter via DHCP service received,

IP_VIA_DHCP_DUP – IP Parameter via DHCP service received, IP address conflict detected

All other Option & Suboption

NO_BLOCK_INFO – No additional block information

Value

This parameter contains the value of the ASE attribute.

Allowed values:

On Option = IP, Suboption = IP_PARAMETER:

Value is set to parts of the value of ASE attribute IP Options / IP Parameter, restricted to IP address, Subnet Mask, and Standard Gateway.

On Option = DEVICE_PROPERTIES, Suboption = NAME_OF_STATION:

Value is set to the value of ASE attribute Device Properties Options / Name Of Station.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ID:

Value is set to the value of ASE attribute Device Properties Options / Device ID.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_ROLE:

Value is set to the value of ASE attribute Device Properties Options / Device Role.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_OPTIONS:

The Value is set to a list containing the mandatory options / suboptions plus those optional options / suboptions supported by the device.

On Option = DEVICE_PROPERTIES, Suboption = DEVICE_INSTANCE:

Value is set to the value of ASE attribute Device Properties Options / Device Instance.

On Option = DEVICE_INITIATIVE, Suboption = DEVICE_INITIATIVE_STATE:

Value is set to the value of ASE attribute Device Initiative Option / Device Initiative.

6.3.7.4 Invocation of the DCP object

There is one DCP object for each interface.

6.3.7.5 Behavior

6.3.7.5.1 Persistency

For persistency of the DCP object the persistency defined in the Communication Interface Management ASE is used; see 7.3.3.2.6.2 for details.

6.3.8 Dynamic host configuration ASE

6.3.8.1 Overview

The utilization of the IETF RFC 2131 (DHCP) is defined for this specification. It includes the usage and the content syntax of Client ID.

6.3.8.2 DHCP class specification

6.3.8.2.1 General

The DHCP ASE defines one DHCP object type:

6.3.8.2.2 Template

A DHCP object is described by the following template:

ASE: DHCP ASE
CLASS: DHCP
CLASS ID: not used
PARENT CLASS: top
ATTRIBUTES:
 1 (m) Key Attribute: Implicit
SERVICES:
 1 (o) OpsService: DHCP Get IP

6.3.8.2.3 Attributes

Implicit

The key attribute Implicit indicates that the DHCP object is implicitly addressed by the service.

6.3.8.3 DHCP service specification

The DHCP services shall be derived from the IETF RFC 2131 (DHCP).

6.3.8.4 Invocation of the DHCP object

There is one DHCP object for each interface.

6.3.9 IEEE 802.1AB ASE

6.3.9.1 Overview

The concepts according to IEEE 802.1AB shall be applied. This document defines the special usage of Chassis ID, Port ID, Chassis ID Subtype, Port ID Subtype, and Time to live. Furthermore, extensions for the OUI regarding frames and MIB are defined.

Furthermore, the IETF RFC 3490 (IDNA) shall be applied for the content syntax of Chassis ID and Port ID.

If Simple Network Management Protocol (SNMP) is supported, the system shall conform to the LLDP management specifications and shall implement the sections of the basic LLDP MIB for the operating mode being implemented. If SNMP is not supported, the system shall

provide storage and retrieval capability equivalent to the functionality for the operating mode being implemented.

6.3.9.2 IEEE 802.1AB class specification

6.3.9.2.1 Template

The IEEE 802.1AB object is described by the following template:

ASE: Link Layer Discovery ASE

CLASS: IEEE 802.1AB

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m) Key Attribute:	Implicit
2	(m) Attribute:	Multiple Interface Mode
2.1	(m) Attribute:	Name Of Device
3	(m) Attribute:	Chassis ID
4	(m) Attribute:	Port ID
5	(m) Attribute:	LLDP Time To Live
6	(o) Attribute:	Port Description
7	(o) Attribute:	System Name
8	(o) Attribute:	System Description
9	(o) Attribute:	System Capabilities
10	(m) Attribute:	Management Address
11	(o) Attribute:	Object Identifier
12	(o) Attribute:	List of Organizationally-specific extensions
12.1	(m) Attribute:	Organizationally Unique Identifier
12.2	(c) Constraint:	Organizationally Unique Identifier = 00-80-C2
12.2.1	(m) Attribute:	List of IEEE 802.1 Subtypes
12.2.1.1	(m) Attribute:	Subtype
12.2.1.2	(c) Constraint:	Subtype = PORT_VLAN_ID
12.2.1.2.1	(m) Attribute:	Port VLAN Identifier
12.2.1.3	(c) Constraint:	Subtype = PORT_AND_PROTOCOL_VLAN_ID
12.2.1.3.1	(m) Attribute:	Flags
12.2.1.3.2	(m) Attribute:	PPVID Reference Number
12.2.1.4	(c) Constraint:	Subtype = VLAN_NAME
12.2.1.4.1	(m) Attribute:	VLAN ID
12.2.1.4.2	(m) Attribute:	VLAN Name
12.2.1.5	(c) Constraint:	Subtype = PROTOCOL_IDENTITY
12.2.1.5.1	(m) Attribute:	List of Protocol Identities
12.2.1.5.1.1	(m) Attribute:	Protocol Identity
12.3	(c) Constraint:	Organizationally Unique Identifier = 00-12-0F
12.3.1	(m) Attribute:	List of IEEE 802.3 Subtypes
12.3.1.1	(m) Attribute:	Subtype
12.3.1.2	(c) Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
12.3.1.2.1	(m) Attribute:	Auto Negotiation Support And Status
12.3.1.2.2	(m) Attribute:	Advertised Capability
12.3.1.2.3	(m) Attribute:	Operational MAU type
12.3.1.3	(c) Constraint:	Subtype = POWER_VIA_MDI
12.3.1.3.1	(m) Attribute:	MDI Power Support
12.3.1.3.2	(m) Attribute:	Power Pair
12.3.1.3.3	(m) Attribute:	Power Class
12.3.1.4	(c) Constraint:	Subtype = LINK_AGGREGATION

12.3.1.4.1	(m) Attribute:	Link Aggregation Status
12.3.1.4.2	(m) Attribute:	Aggregated Port ID
12.3.1.5	(c) Constraint:	Subtype = MAXIMUM_FRAME_SIZE
12.3.1.5.1	(m) Attribute:	Maximum Frame Size
12.4	(c) Constraint:	Organizationally Unique Identifier = 00-0E-CF
12.4.1	(m) Attribute:	List of PNIO Subtypes
12.4.1.1	(m) Attribute:	Subtype
12.4.1.2	(c) Constraint:	Subtype = LLDP_PNIO_DELAY
12.4.1.2.1	(m) Attribute:	PnioDelay
12.4.1.2.1.1	(m) Attribute:	PTCP_PortRxDelayLocal
12.4.1.2.1.2	(m) Attribute:	PTCP_PortRxDelayRemote
12.4.1.2.1.3	(m) Attribute:	PTCP_PortTxDelayLocal
12.4.1.2.1.4	(m) Attribute:	PTCP_PortTxDelayRemote
12.4.1.2.1.5	(m) Attribute:	CableDelayLocal
12.4.1.3	(c) Constraint:	Subtype = LLDP_PNIO_PORTSTATUS
12.4.1.3.1	(m) Attribute:	PnioPortStatus
12.4.1.3.1.1	(m) Attribute:	RTClass2_PortStatus
12.4.1.3.1.1.1	(m) Attribute:	Status
12.4.1.3.1.2	(m) Attribute:	RTClass3_PortStatus
12.4.1.3.1.2.1	(m) Attribute:	Status
12.4.1.3.1.2.2	(m) Attribute:	Fragmentation
12.4.1.3.1.2.3	(m) Attribute:	PreambleLength
12.4.1.3.1.2.4	(m) Attribute:	Optimized
12.4.1.4	(c) Constraint:	Subtype = LLDP_PNIO_ALIAS
12.4.1.4.1	(m) Attribute:	AliasNameValue
12.4.1.5	(c) Constraint:	Subtype = LLDP_PNIO_MRPPORTSTATUS
12.4.1.5.1	(m) Attribute:	PnioMRPPortStatus
12.4.1.5.1.1	(m) Attribute:	MRP_DomainUUID
12.4.1.5.1.2	(m) Attribute:	MRRT_PortStatus
12.4.1.6	(c) Constraint:	Subtype = LLDP_PNIO_CHASSIS_MAC
12.4.1.6.1	(m) Attribute:	ChassisMacAddress
12.4.1.7	(c) Constraint:	Subtype = LLDP_PNIO_PTCPSTATUS
12.4.1.7.1	(m) Attribute:	PnioPTCPStatus
12.4.1.7.1.1	(m) Attribute:	PTCP_MasterSourceAddress
12.4.1.7.1.2	(m) Attribute:	PTCP_SubdomainUUID
12.4.1.7.1.3	(m) Attribute:	IRDataUUID
12.4.1.7.1.4	(m) Attribute:	LLDP_LengthOfPeriod
12.4.1.7.1.4.1	(m) Attribute:	Length
12.4.1.7.1.4.2	(m) Attribute:	Validity
12.4.1.7.1.5	(m) Attribute:	LLDP_RedPeriodBegin
12.4.1.7.1.6	(m) Attribute:	LLDP_GreenPeriodBegin
12.4.1.8	(c) Constraint:	Subtype = LLDP_PNIO_MAUTYPEEXTENSION
12.4.1.8.1	(m) Attribute:	PnioMAUTypeExtension
12.4.1.8.1.1	(m) Attribute:	MAUTypeExtension
13	(m) Attribute:	List of Remote Systems Data
13.1	(m) Attribute:	Port ID
13.2	(m) Attribute:	Chassis ID
13.3	(m) Attribute:	MAC Address
13.4	(m) Attribute:	LLDP Time To Live
13.5	(o) Attribute:	Port Description
13.6	(o) Attribute:	System Name
13.7	(o) Attribute:	System Description
13.8	(o) Attribute:	System Capabilities

13.9	(o) Attribute:	Management Address
13.10	(o) Attribute:	Object Identifier
13.11	(m) Attribute:	Line Delay
13.12	(m) Attribute:	List of Organizationally-specific extensions
13.12.1	(m) Attribute:	Organizationally Unique Identifier
13.12.2	(c) Constraint:	Organizationally Unique Identifier = 00-80-C2
13.12.2.1	(m) Attribute:	List of IEEE 802.1 Subtypes
13.12.2.1.1	(m) Attribute:	Subtype
13.12.2.1.2	(c) Constraint:	Subtype = PORT_VLAN_ID
13.12.2.1.2.1	(m) Attribute:	Port VLAN Identifier
13.12.2.1.3	(c) Constraint:	Subtype = PORT_AND_PROTOCOL_VLAN_ID
13.12.2.1.3.1	(m) Attribute:	Flags
13.12.2.1.3.2	(m) Attribute:	PPVID Reference Number
13.12.2.1.4	(c) Constraint:	Subtype = VLAN_NAME
13.12.2.1.4.1	(m) Attribute:	VLAN ID
13.12.2.1.4.2	(m) Attribute:	VLAN Name
13.12.2.1.5	(c) Constraint:	Subtype = PROTOCOL_IDENTITY
13.12.2.1.5.1	(m) Attribute:	List of Protocol Identities
13.12.2.1.5.1.1	(m) Attribute:	Protocol Identity
13.12.3	(c) Constraint:	Organizationally Unique Identifier = 00-12-0F
13.12.3.1	(m) Attribute:	List of IEEE 802.3 Subtypes
13.12.3.1.1	(m) Attribute:	Subtype
13.12.3.1.2	(c) Constraint:	Subtype = MAC_PHY_CONFIGURATION_STATUS
13.12.3.1.2.1	(m) Attribute:	Auto Negotiation Support And Status
13.12.3.1.2.2	(m) Attribute:	Advertised Capability
13.12.3.1.2.3	(m) Attribute:	Operational MAU type
13.12.3.1.3	(c) Constraint:	Subtype = POWER_VIA_MDI
13.12.3.1.3.1	(m) Attribute:	MDI Power Support
13.12.3.1.3.2	(m) Attribute:	Power Pair
13.12.3.1.3.3	(m) Attribute:	Power Class
13.12.3.1.4	(c) Constraint:	Subtype = LINK_AGGREGATION
13.12.3.1.4.1	(m) Attribute:	Link Aggregation Status
13.12.3.1.4.2	(m) Attribute:	Aggregated Port ID
13.12.3.1.5	(c) Constraint:	Subtype = MAXIMUM_FRAME_SIZE
13.12.3.1.5.1	(m) Attribute:	Maximum Frame Size
13.12.4	(c) Constraint:	Organizationally Unique Identifier = 00-0E-CF
13.12.4.1	(m) Attribute:	List of PNIO Subtypes
13.12.4.1.1	(m) Attribute:	Subtype
13.12.4.1.2	(c) Constraint:	Subtype = LLDP_PNIO_DELAY
13.12.4.1.2.1	(m) Attribute:	PnioDelay
13.12.4.1.2.1.1	(m) Attribute:	PTCP_PortRxDelayLocal
13.12.4.1.2.1.2	(m) Attribute:	PTCP_PortRxDelayRemote
13.12.4.1.2.1.3	(m) Attribute:	PTCP_PortTxDelayLocal
13.12.4.1.2.1.4	(m) Attribute:	PTCP_PortTxDelayRemote
13.12.4.1.2.1.5	(m) Attribute:	CableDelayLocal
13.12.4.1.3	(c) Constraint:	Subtype = LLDP_PNIO_PORTSTATUS
13.12.4.1.3.1	(m) Attribute:	PnioPortStatus
13.12.4.1.3.1.1	(m) Attribute:	RTClass2_PortStatus
13.12.4.1.3.1.2	(m) Attribute:	RTClass3_PortStatus
13.12.4.1.3.1.2.1	(m) Attribute:	Status
13.12.4.1.3.1.2.2	(m) Attribute:	Fragmentation
13.12.4.1.3.1.2.3	(m) Attribute:	PreambleLength
13.12.4.1.3.1.2.4	(m) Attribute:	Optimized

13.12.4.1.4	(c) Constraint:	Subtype = LLDP_PNIO_ALIAS
13.12.4.1.4.1	(m) Attribute:	AliasNameValue
13.12.4.1.5	(c) Constraint:	Subtype = LLDP_PNIO_MRPPORTSTATUS
13.12.4.1.5.1	(m) Attribute:	PnioMRPPortStatus
13.12.4.1.5.1.1	(m) Attribute:	MRP_DomainUUID
13.12.4.1.5.1.2	(m) Attribute:	MRRT_PortStatus
13.12.4.1.6	(c) Constraint:	Subtype = LLDP_PNIO_MRPICPORTSTATUS
13.12.4.1.6.1	(m) Attribute:	PnioMRPICPortStatus
13.12.4.1.6.1.1	(m) Attribute:	MRPIC_DomainID
13.12.4.1.6.1.2	(m) Attribute:	MRPIC_Role
13.12.4.1.7	(c) Constraint:	Subtype = LLDP_PNIO_CHASSIS_MAC
13.12.4.1.7.1	(m) Attribute:	ChassisMacAddress
13.12.4.1.8	(c) Constraint:	Subtype = LLDP_PNIO_PTCPSTATUS
13.12.4.1.8.1	(m) Attribute:	PnioPTCPStatus
13.12.4.1.8.1.1	(m) Attribute:	PTCP_MasterSourceAddress
13.12.4.1.8.1.2	(m) Attribute:	PTCP_SubdomainUUID
13.12.4.1.8.1.3	(m) Attribute:	IRDataUUID
13.12.4.1.8.1.4	(m) Attribute:	LLDP_LengthOfPeriod
13.12.4.1.8.1.4.1	(m) Attribute:	Length
13.12.4.1.8.1.4.2	(m) Attribute:	Validity
13.12.4.1.8.1.5	(m) Attribute:	LLDP_RedPeriodBegin
13.12.4.1.8.1.6	(m) Attribute:	LLDP_GreenPeriodBegin
13.12.4.1.9	(c) Constraint:	Subtype = LLDP_PNIO_MAUTypeExtension
13.12.4.1.9.1	(m) Attribute:	PnioMAUTypeExtension
13.12.4.1.9.1.1	(m) Attribute:	MAUTypeExtension

SERVICES:

- 1 (m) OpsService: Remote Systems Data Change

6.3.9.2.2 Attributes

Implicit

The key attribute Implicit indicates that the object is implicitly addressed by the service.

Multiple Interface Mode

This attribute consists of the following attributes:

Name Of Device

This attribute controls the behavior of the IEEE 802.1AB ASE regarding Chassis and Port ID.

The value FALSE indicates that the attribute Chassis ID contains the Name Of Station and the attribute Port ID contains the Name of Port.

The value TRUE indicates that the attribute Chassis ID is defined by local means and the attribute Port ID contains the Name Of Port + "." + the Name Of Station.

Attribute type: Boolean

Default value: FALSE

Chassis ID

This non-volatile attribute contains the Chassis ID which shall be used as Chassis ID subtype locally assigned compliant with IEEE 802.1AB. This field shall contain a VisibleString indicating the specific identifier for the particular chassis in this system.

The value of the attribute is dependent on the attribute Multiple Interface Mode.

If the attribute Name Of Device is set to FALSE this attribute contains the Name Of Station as defined in the Discovery and basic configuration ASE.

If the attribute Name Of Device is set to TRUE the content of this attribute is defined by local means.

Attribute type: VisibleString[240]

Port ID

This attribute contains the name of the local port which shall be used as Port ID subtype locally assigned compliant with IEEE 802.1AB.

The value of the attribute is dependent on the attribute Multiple Interface Mode.

If the attribute Name Of Device is set to FALSE, the attribute contains the Name of Port in the form “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 00 000 up to 65 535. The values “port-001” or “port-001-00000” shall be used for the first port submodule for an interface within slot 0. The values x, y, z shall be used to identify the port number. The values r, s, t, u, v shall be used to identify the slot.

If the attribute Name Of Device is set to TRUE, the attribute contains the Name of Port (“port-xyz” or “port-xyz-rstuv”) + “.” + Name Of Station as defined in the Discovery and basic configuration ASE.

Attribute type: VisibleString[240]

NOTE 1 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 802.1AB.

Attribute type: Unsigned16

Allowed Values: 20

Port Description

This attribute contains an alpha-numeric string that indicates the port description. It shall be compliant with IEEE 802.1AB.

If IETF RFC 2863 is implemented, the use of the ifDescr object is recommended for this field.

System Name

This attribute contains an alpha-numeric string that indicates the system’s administratively-assigned name. It shall be compliant with IEEE 802.1AB.

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 contains an alpha-numeric string that is the textual description of the network entity. It shall be compliant with IEEE 802.1AB.

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 contains 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 69. A binary one in the associated bit indicates the existence of that capability. It shall be compliant with IEEE 802.1AB.

Individual systems may indicate more than one implemented functional capability (for example, both a bridge and router capability).

Table 69 – 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 contains an OctetString indicating the particular management address. It shall be compliant with IEEE 802.1AB.

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 identifies the type of hardware component or protocol entity associated with the indicated management address. If no object identifier is available, this field shall not be provided. It shall be compliant with IEEE 802.1AB.

NOTE 2 The interface number and object identifier are included to assist NMS discovery by indicating Enterprise-specific or other starting points for the search, such as the Interface (see IETF RFC 2863) or Entity MIB (see IETF RFC 2737).

List of Organizationally-specific extensions

This attribute list contains the following attributes:

Organizationally Unique Identifier

This attribute contains an OUI that is provided to allow different organizations, such as IEEE 802, IETF, as well as individual software and equipment vendors, to define Type Length Values (TLV) that advertise information to remote entities.

List of IEEE 802.1 Subtypes

This attribute list contains the following attributes:

Subtype

This attribute shall be compliant with IEEE 802.1AB.

Allowed values: PORT_VLAN_ID, PORT_AND_PROTOCOL_VLAN_ID, VLAN_NAME, PROTOCOL_IDENTITY

Port VLAN Identifier

This attribute contains 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 port VLAN identifier or does not support port-based VLAN operation.

Flags

This attribute contains a bit map indicating the port and protocol VLAN capability and status.

Allowed Values: PPVID_SUPPORTED, PPVID_ENABLED

PPVID Reference Number

This attribute contains the port and protocol VALN ID reference number for this 802 LAN station.

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 contains the VID number associated with the VLAN name.

VLAN Name

This attribute contains the VLAN's name.

Attribute type: OctetString[32]

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 contains 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 3 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 IEEE 802.3 Subtypes

This attribute list contains the following attributes:

Subtype

This attribute shall be compliant with IEEE 802.1AB.

Allowed values: MAC_PHY_CONFIGURATION_STATUS, POWER_VIA_MDI, LINK_AGGREGATION, MAXIMUM_FRAME_SIZE

Auto Negotiation Support And Status

This attribute contains a bit map that identifies the auto-negotiation support and current status of the local IEEE 802.3 LAN station as defined in Table 70. If auto-negotiation support bit (bit-0) is one and the auto-negotiation status bit (bit-1) is zero, the IEEE 802.3 physical media dependent sublayer operating mode will be determined the default configuration field value rather than by auto-negotiation.

Table 70 – 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 4 This attribute can be written via Write PD Port Data Adjust.

Advertised Capability

This attribute contains an integer value as defined by the ifMauAutoNegCapAdvertisedBits object in IETF RFC 4836.

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 4836 (or subsequent revisions) and is equal to the last number in the respective dot3MauType. For MAU types not listed in IETF RFC 4836 (or subsequent revisions), the value of this field shall be set to zero.

MDI Power Support

This attribute contains a bit-map of the MDI power capabilities and status as defined in Table 71.

Table 71 – 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 cannot be controlled, 1 - pair selection can be controlled)

Power Pair

This attribute contains an integer value as defined by the pethPsePortPowerPairs object in IETF RFC 3621.

Power Class

This attribute contains an integer value as defined by the pethPsePortPowerClassification object in IETF RFC 3621.

Link Aggregation Status

This attribute contains a bit map of the link aggregation capabilities and the current aggregation status of the link as defined in Table 72.

Table 72 – 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 contains the ISO/IEC 15802-1 aggregated port identifier. An AggPortID is derived from the ifNumber in the ifIndex for the interface.

Maximum Frame Size

This attribute contains an integer value indicating the maximum supported frame size in octets shall be set to 1 522.

List of PNIO Subtypes

This attribute list contains the following attributes:

Subtype

This attribute contains the following values.

Allowed values: LLDP_PNIO_DELAY, LLDP_PNIO_PORTSTATUS, LLDP_PNIO_ALIAS, LLDP_PNIO_MRP_PORT_STATUS, LLDP_PNIO_INTERFACE_MAC_ADDRESS, LLDP_PNIO_PTCP_STATUS, LLDP_PNIO_MAUTypeEx

PnioDelay

This attribute consists of the following attributes:

PTCP_PortRxDelayLocal

This attribute contains the local receiving delay of the port in nanoseconds.

Attribute type: Unsigned32

Allowed values: UNKNOWN (0x00000000), 0x00000001 to 0x00000FFF

PTCP_PortRxDelayRemote

This attribute contains the local receiving delay of the peer port in nanoseconds.

Attribute type: Unsigned32

Allowed values: UNKNOWN (0x00000000), 0x00000001 to 0x00000FFF

PTCP_PortTxDelayLocal

This attribute contains the local transmission delay of the port in nanoseconds.

Attribute type: Unsigned32

Allowed values: UNKNOWN (0x00000000), 0x00000001 to 0x00000FFF

PTCP_PortTxDelayRemote

This attribute contains the local transmission delay of the peer port in nanoseconds.

Attribute type: Unsigned32

Allowed values: UNKNOWN (0x00000000), 0x00000001 to 0x00000FFF

CableDelayLocal

This attribute contains the local cable delay in nanoseconds.

Attribute type: Unsigned32

Allowed values: UNKNOWN (0x00000000), 0x00000001 to 0x000FFFFF

PnioPortStatus

This attribute consists of the following attributes:

RTClass2_PortStatus

This attribute contains the following attributes.

Status

This attribute contains the port status for RT_CLASS_2.

Allowed values: OFF, RTCLASS2_RUN

RTClass3_PortStatus

This attribute contains the following attributes.

Status

This attribute contains the port status for RT_CLASS_3.

Allowed values: OFF, RTCLASS3_UP, RTCLASS3_RUN

Fragmentation

This attribute contains the fragmentation mode of this port.

Allowed values: OFF, ON

PreambleLength

This attribute contains the preamble length setting of this port.

Allowed values: SEVEN, ONE

Optimized

This attribute contains the port mode for RT_CLASS_3.

Allowed values: OFF, ON

AliasNameValue

This attribute contains the following VisibleString:

"Value of attribute Port ID" + "." + "Value of Chassis ID"

Attribute type: VisibleString

PnioMRPPortStatus

This attribute shall consist of the following attributes.

MRP_DomainUUID

This attribute contains the MRP domain UUID.

Attribute type: UUID

MRRT_PortStatus

This attribute contains the MRRT port status (will never be implemented).

Attribute type: Unsigned8

Allowed Values: OFF

PnioMRPICPortStatus

This attribute shall consist of the following attributes.

MRPIC_DomainID

This attribute contains the MRPIC domain ID.

Attribute type: Unsigned16

Allowed Values: 0x0000 (no MRP interconnection domain assigned), 0x0001 to 0xFFFF

MRPIC_Role

This attribute contains the MRP interconnection role.

Attribute type: Unsigned16

Allowed Values: NO_ROLE_ASSIGNED, MIM (MRP interconnection manager), MIC (MRP interconnection client)

ChassisMacAddress

This attribute contains the IEEE 802 MAC address.

Attribute type: OctetString[6]

PnioPTCPStatus

This attribute shall consist of the following attributes.

PTCP_MasterSourceAddress

This attribute contains the IEEE 802 MAC address of the PTCP master.

Attribute type: OctetString[6]

PTCP_SubdomainUUID

This attribute contains the UUID of the PTCP subdomain.

Attribute type: UUID

IRDataUUID

This attribute contains the UUID of the Isochronous Relay (IR) data.

Attribute type: UUID

LLDP_LengthOfPeriod

This attribute shall consist of the following attributes.

Length

This attribute contains the value for the duration of the cycle in nanoseconds. The value shall be a multiply of 31 250 nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00007A12 to 0x003D0900 if valid, 0 if invalid

Validity

This attribute contains the validity of the length.

Allowed values: INVALID, VALID

LLDP_RedPeriodBegin

This attribute shall consist of the following attributes.

Offset

This attribute contains the value for the offset related to the start of the cycle in nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00000000 to 0x003D08FF if valid, 0 if invalid

Validity

This attribute contains the validity of the offset.

Allowed values: INVALID, VALID

LLDP_GreenPeriodBegin

This attribute shall consist of the following attributes.

Offset

This attribute contains the value for the offset related to the start of the cycle in nanoseconds.

Attribute type: Unsigned32

Allowed values: 0x00000000 to 0x003D08FF if valid, 0 if invalid

Validity

This attribute contains the validity of the offset.

Allowed values: INVALID, VALID

PnioMAUTypeExtension

This optional attribute shall consist of the following attributes.

MAUTypeExtension

This document defines, in extension to the IEEE 802.3, additional MAUTypes. This MAUTypeExtension indicates an extension of the attribute OperationalMAUType.

The value 0x0 indicates that the OperationalMAUType is not extended.

Attribute type: Unsigned16

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 contains the devices physical address according to IEEE 802.3 MAC address.

Attribute type: OctetString[6]

Line Delay

This attribute contains the measured line delay. The time base shall be 1 ns.

Attribute type: Unsigned32

6.3.9.2.3 Invocation of the IEEE 802.1AB object

For the invocation of the IEEE 802.1AB object the following rules apply:

- IEEE 802.1AB objects shall exist for each interface.

6.3.9.3 IEEE 802.1AB service specification

The protocol according to IEEE 802.1AB shall be used to send and receive the attribute values of the ASE. IEEE 802.1AB uses the services of the LPM to transmit and receive information to and from other LLDP agents.

6.3.9.3.1 Remote systems data change

This local service shall be used to indicate a change of certain Remote Systems Data. Table 73 shows the parameters of the service.

Table 73 – Remote systems data change

Parameter name	ind
Argument	M
Local Port ID	M
List of Neighbors	U
Remote Chassis ID	U
Remote Port ID	U
Remote Cable Delay	U
Remote RT_CLASS_3 Port Status	U
Remote PTCP Status	U
Remote Line Delay	U
Auto Negotiation Support And Status	U
Advertised Capability	U
Operational MAU type	U

Argument

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

Local Port ID

This parameter is the local identifier.

List of Neighbors

This parameter list consists of the following parameters:

Remote Chassis ID

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

Remote Port ID

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

Remote Cable Delay

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

Remote RT_CLASS_3 Port Status

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

Remote PTCP Status

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

Remote Line Delay

This parameter contains the value of the sum of the following attribute of the ASE:

CableDelayLocal

PortRxDelayLocal

PortTxDelayRemote

Auto Negotiation Support And Status

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

Advertised Capability

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

Operational MAU type

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

6.3.9.3.2 Placeholder

This is an empty subclause to be in line with IEC directive part 2 and the structure of service specifications.

6.3.10 Media redundancy ASE**6.3.10.1 Overview**

For media redundancy, IEC 62439-2 applies. The support of the MRP Management Information Base (MIB) is optional.

The MRP consists of a service and a protocol entity. Both are described by IEC 62439-2.

6.3.10.2 Media redundancy class specification**6.3.10.2.1 General**

The Media redundancy ASE defines one object type.

6.3.10.2.2 Template

A Media redundancy object is described by the following template:

ASE:	Media redundancy ASE
CLASS:	Media redundancy
CLASS ID:	not used
PARENT CLASS:	IEEE 802.3, MAC Bridges ASE, IEEE 802.1 AB
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of MRP Instances
2.1	(m) Attribute: MRP Instance
2.2	(m) Attribute: MRP Domain UUID
2.3	(m) Attribute: MRP Domain Name
2.4	(m) Attribute: MRP Version
2.5	(m) Attribute: Administrative MRP Role
2.6	(m) Attribute: Operational MRP Role
2.7	(c) Constraint: Operational MRP Role = MANAGER or MANAGER_AUTO
2.7.1	(m) Attribute: MRP Prio
2.7.2	(m) Attribute: MRP TOPchgT
2.7.3	(m) Attribute: MRP TOPNRmax
2.7.4	(m) Attribute: MRP TSTshortT
2.7.5	(m) Attribute: MRP TSTdefaultT
2.7.6	(m) Attribute: MRP TSTNRmax
2.7.7	(m) Attribute: MRP Ring State
2.7	(c) Constraint: Operational MRP Role = CLIENT
2.7.1	(m) Attribute: MRP LNKdownT
2.7.2	(m) Attribute: MRP LNKupT
2.7.3	(m) Attribute: MRP LNKNRmax
2.8	(m) Attribute: MRP Data Check
2.8.1	(m) Attribute: MRP Domain UUID
2.8.2	(m) Attribute: MRP Check
2.8.2.1	(m) Attribute: Media Redundancy Manager Check
2.8.2.2	(m) Attribute: Domain UUID Check
2.9	(m) Attribute: Ringport 1 ID
2.10	(m) Attribute: Ringport 2 ID
3	(m) Attribute: List of MRPIC Ports
3.1	(m) Attribute: MRPIC Port ID
3.2	(m) Attribute: MRPIC Domain ID
3.3	(m) Attribute: MRPIC Role
3.4	(m) Attribute: MRPIC Domain Name
3.5	(c) Attribute: MRPIC MIM Params
3.5.1	(m) Attribute: MRPIC TOPchgT
3.5.2	(m) Attribute: MRPIC TOPNRmax
3.5.3	(m) Attribute: MRPIC LinkStatusChangeT
3.5.4	(m) Attribute: MRPIC LinkStatusNRmax
3.5.5	(m) Attribute: MRPIC LinkStatusStartDelay
3.6	(c) Attribute: MRPIC State Data
3.7	(c) Attribute: MRPIC MIC Params
3.7.1	(m) Attribute: MRPIC LNKdownT
3.7.2	(m) Attribute: MRPIC LNKupT
3.7.3	(m) Attribute: MRPIC LNKNRmax
3.7.4	(m) Attribute: MRPIC LinkStatusStartDelay

SERVICES:

none

See also the Media Redundancy ASE in IEC 62439-2.

6.3.10.2.3 Attributes**List of MRP Instances**

This attribute contains the MRP instances of this node. A list element consists of the following attributes:

MRP Instance

This attribute defines the internal reference number of the MRP instance.

Attribute type: Unsigned8

Allowed values: 0 to 126

MRP Domain UUID

This attribute defines the redundancy domain representing the ring the Media redundancy object belongs to. It is set to default domain ID or provided as unique ID by engineering.

Attribute type: UUID

MRP Domain Name

This attribute defines the redundancy domain representing the ring the Media redundancy object belongs to. It is set to default domain name or provided as unique ID by engineering.

Attribute type: VisibleString[240]

MRP Version

This attribute specifies the version of the Media redundancy object in the redundancy domain referenced by the value of the attribute Domain ID.

Attribute type: Unsigned16

Allowed Values: 1

Administrative MRP Role

This attribute specifies the administrative role of the Media redundancy object in the redundancy domain referenced by the value of the attribute Domain ID.

Attribute type: Unsigned16

Allowed Values: DISABLED, MANAGER, CLIENT, MANAGER_AUTO

Operational MRP Role

This attribute specifies the operational role of the Media redundancy object in the redundancy domain referenced by the value of the attribute Domain ID.

Attribute type: Unsigned16

Allowed Values: DISABLED, MANAGER, CLIENT, MANAGER_AUTO

MRP Prio

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the manager's priority. A lower value indicates a higher priority.

Attribute type: Unsigned16

Allowed values: see IEC 62439-2

MRP TOPchgT

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the interval for sending topology change frames.

Attribute type: Unsigned16

MRP TOPNRmax

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the interval count which controls repeated transmission of topology change frames.

Attribute type: Unsigned16

MRP TSTshortT

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the short interval for sending test frames on ringports after link changes in the ring.

Attribute type: Unsigned16

MRP TSTdefaultT

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the default interval for sending test frames on ringports.

Attribute type: Unsigned16

MRP TSTNRmax

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the interval count for monitoring the reception of test frames.

Attribute type: Unsigned16

MRP Ring State Data

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the real ring state of the Media redundancy object in the redundancy domain. Note that only the node acting as MRM knows the current state of the ring (open or closed).

Attribute type: Unsigned16

Allowed values:

OPEN (Ring is open due to link or device failure in ring),

CLOSED (Ring is closed (normal operation, no error)),

UNDEFINED (node is in operational role CLIENT or DISABLED)

MRP LNKdownT

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the interval for sending link down Link Change frames on ringports.

Attribute type: Unsigned16

MRP LNKupT

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the interval for sending link up Link Change frames on ringports.

Attribute type: Unsigned16

MRP LKNRmax

This conditional attribute is present when the value of the operational MRP role attribute is "CLIENT". It contains the Link Change frame count which controls repeated transmission of Link up or link down frames.

Attribute type: Unsigned16

MRP Data Check

This parameter contains the checking rules. It consists of the following attributes:

MRP Domain UUID

This parameter contains the MRP Domain UUID to check against, if the Domain UUID Check attribute is set to TRUE.

Attribute type: UUID

MRP Check

This attribute consists of the following elements:

Media Redundancy Manager Check

This attribute contains the value TRUE to enable diagnosis with channel error type MEDIA_REDUNDANCY_MISMATCH to be added to the diagnosis ASE. Otherwise it shall be set to FALSE.

Attribute type: Boolean

Domain UUID Check

This attribute contains the value TRUE to enable the continued evaluation of MRP Domain UUID with each Remote System Data Change indication as specified within the Neighborhood Check state machine. Otherwise it shall be set to FALSE.

Attribute type: Boolean

Ringport 1 ID

This attribute specifies one port of a bridge which is assigned as ringport 1 in the redundancy domain referenced by the value of the attribute Domain ID. One port shall only belong to a single redundancy domain.

Attribute type: Unsigned16

Ringport 2 ID

This attribute specifies another port of a bridge different from Ringport 1 ID which is assigned as ringport 2 in the redundancy domain referenced by the value of the attribute Domain ID.

Attribute type: Unsigned16

List of MRPIC Ports

This attribute contains the ports of this node used for MRP interconnection. A list element consists of the following attributes:

MRPIC Port ID

This attribute specifies one port of a bridge which is assigned as MRP interconnection port in the interconnection domain referenced by the value of the attribute Domain ID. One port shall only belong to a single interconnection domain.

Attribute type: Unsigned16

MRPIC Domain ID

This attribute contains the MRPIC domain ID.

Attribute type: Unsigned16

Allowed Values: 0x0000 (no MRP interconnection domain assigned), 0x0001 to 0xFFFF

MRPIC Role

This attribute contains the MRP interconnection role.

Attribute type: Unsigned16

Allowed Values: NO_ROLE_ASSIGNED, MIM (MRP interconnection manager), MIC (MRP interconnection client)

MRP Domain Name

This attribute defines the domain name. It is set to default domain name or provided as unique ID by engineering.

Attribute type: VisibleString[240]

MRPIC MIM Params

This conditional attribute specifies the parameters for the MRP interconnection manager. It is present only when the value of MRPIC role is "MIM".

MRPIC TOPchgT

This attribute contains the interval for sending repetition of interconnection topology change frames for the MIM.

Attribute type: Unsigned16

Time base: 10 ms

MRP TOPNRmax

This attribute contains the interval count which controls repeated transmission of interconnection topology change frames for the MIM.

Attribute type: Unsigned16

MRP LinkStatusChangeT

This attribute contains the interconnection Link Status poll time interval for the MIM.

Attribute type: Unsigned16

Time base: 1 ms

MRPIC LinkStatusNRmax

This attribute contains the maximum outstanding response to Link Status poll frame count for the MIM. A higher value causes an interconnection failure.

Attribute type: Unsigned16

MRPIC LinkStatusStartDelay

This attribute contains the startup delay for the MIM.

Attribute type: Unsigned16

Time base: 100 ms

MRPIC State Data

This conditional attribute contains the state of the MRP interconnection. It is present only when the value of MRPIC role is "MIM".

Attribute type: Unsigned16

Allowed Values:

OPEN (MRP interconnection is open due to link or device failure in interconnection),

CLOSED (MRP interconnection is closed (normal operation, no error))

MRPIC MIC Params

This conditional attribute specifies the parameters for the MRP interconnection client. It is present only when the value of MRPIC role is "MIC".

MRPIC LNKdownT

This attribute contains the interval for sending link down Interconnection Link Change frames on ring ports and interconnection ports for the MIC.

Attribute type: Unsigned16

Time base: 1 ms

MRPIC LNKupT

This attribute contains the interval for sending link up Interconnection Link Change frames on ring ports and interconnection ports for the MIC.

Attribute type: Unsigned16

Time base: 1 ms

MRPIC LNKNRmax

This attribute contains the Interconnection Link Change frame count which controls repeated transmission of Interconnection Link up or Link down frames for the MIC.

Attribute type: Unsigned16

MRPIC LinkStatusStartDelay

This attribute contains the startup delay for the MIC.

Attribute type: Unsigned16

Time base: 100 ms

6.3.10.3 Media redundancy service specification

The media redundancy service specification is defined in IEC 62439-2.

6.3.10.4 Media redundancy behavior**6.3.10.4.1 General**

Media redundancy events shall cause diagnosis events and alarm notifications if an application relationship is established and the attribute Check Media Redundancy has the value TRUE.

6.3.10.4.2 Media redundancy diagnosis dependencies

Figure 3 shows the dependencies of media redundancy diagnosis:

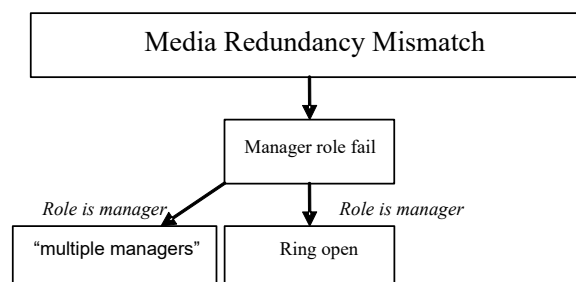


Figure 3 – Media redundancy diagnosis dependencies

If a node in a media redundancy domain is in operational role **MANAGER** or **MANAGER_AUTO** and another node with operational role **MANAGER** or **MANAGER_AUTO** is detected, the “multiple managers” diagnosis shall be added to the diagnosis ASE.

If a node in a media redundancy domain is in operational role **MANAGER** or **MANAGER_AUTO** and an open ring is detected, the “ring open” diagnosis shall be added to the diagnosis ASE.

These events are indicated by means of the State Change service.

In addition, media redundancy handles the following event for both ringports of a redundancy domain by means of the Neighborhood Change service:

If the attribute **Check Ringport Neighborhood** is **TRUE** and the attribute **Ringport Domain State** is **BAD** or **UNDEFINED**, the “Peer Domain ID mismatch” event shall be signaled.

6.3.10.5 Invocation of the MRP object

There is one MRP object for each interface supporting MRP.

6.3.11 Precision time control ASE

6.3.11.1 Concepts

The “Precision Transparent Clock Protocol” (PTCP) represents a uniform method of distributing a time. In particular, this means that a PTCP master synchronizes an amount of PTCP slaves in the sub-microsecond-range. It supports a PTCP best master model for one domain. The transport protocol PTCP is data link layer based.

A PTCP domain shall be used for **WORKING_CLOCK**, which is a local time to synchronize the network access (in particular the cyclic communication) for all devices participating in the particular PTCP domain.

Additionally PTCP supports the measurement of the line delay between two corresponding ports. Figure 4 shows the context of the precision time control ASE.

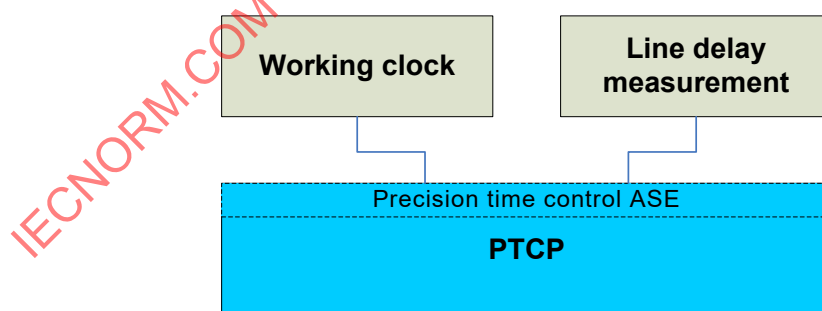


Figure 4 – PTCP applications

6.3.11.2 PTCP class specification

6.3.11.2.1 Template

ASE: PTCP ASE

CLASS: PTCP

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m)	Key Attribute:	Sync ID
2	(m)	Attribute:	Domain
2.1	(m)	Attribute:	Domain UUID
2.2	(m)	Attribute:	Sequence ID
2.3	(m)	Attribute:	Sync Send Factor
2.4	(m)	Attribute:	PTCP Takeover Timeout
2.5	(m)	Attribute:	PTCP Timeout
2.6	(m)	Attribute:	Role (MASTER, SLAVE)
2.7	(c)	Constraint:	Role = MASTER
2.7.1	(m)	Attribute:	Priority
2.7.2	(m)	Attribute:	Accuracy
2.7.3	(m)	Attribute:	Variance
2.7.4	(m)	Attribute:	PTCP Master Startup Time
2.8	(c)	Constraint:	Sync ID = WORKING CLOCK
2.8.1	(m)	Attribute:	PTCP Time
2.8.1.1	(m)	Attribute:	Epoch Number
2.8.1.2	(m)	Attribute:	Seconds
2.8.1.3	(m)	Attribute:	Nano Seconds
2.8.1.4	(m)	Attribute:	Current UTC Offset
2.8.2	(m)	Attribute:	Local Time
2.9	(m)	Attribute:	PLL Window
2.10	(m)	Attribute:	Sync State Info
2.10.1	(m)	Attribute:	Jitter out of Boundary
2.10.2	(m)	Attribute:	No Sync Message Received
2.10.3	(m)	Attribute:	Wrong PTCP Domain UUID
2.11	(m)	Attribute:	Rate Compensation Factor
2.12	(m)	Attribute:	List of Ports
2.12.1	(m)	Attribute:	PortID
2.12.2	(m)	Attribute:	Borderline
2.12.2.1	(m)	Attribute:	Ingress
2.12.2.2	(m)	Attribute:	Egress
2.12.3	(m)	Attribute:	Line Delay
2.12.4	(m)	Attribute:	Rate Compensation Factor Peer
2.13	(o)	Attribute:	List of OUIs
2.13.1	(m)	Attribute:	OUI
2.13.2	(s)	Attribute:	Subtype
2.13.3	(s)	Attribute:	Data Block

SERVICES:

1	(m)	OpsService:	Start Bridge
2	(m)	OpsService:	Start Slave
3	(o)	OpsService:	Start Master
4	(m)	OpsService:	Stop Bridge
5	(m)	OpsService:	Stop Slave
6	(o)	OpsService:	Stop Master
7	(m)	OpsService:	Sync State Change

6.3.11.2.2 Attributes

Sync ID

This key attribute identifies the instance of the protocol machine to which the data belong. The value WORKING_CLOCK shall be used for cycle synchronization.

Attribute type: Unsigned8

Allowed values: WORKING_CLOCK

Domain

This attribute consists of the following elements:

Domain UUID

This attribute contains the UUID of the domain for synchronization which is provided by the engineering system and issued by a master. The UUID is used for a logical grouping of PTCP clocks that synchronize each other using the PTCP protocol. Clocks in another PTCP domains shall not be synchronized.

Attribute type: UUID

Sequence ID

This attribute contains the sequence number of the sync message. It shall be incremented with every new sync messages. By means of the Sequence ID the receiver detects duplications.

Attribute type: Unsigned16

Sync Send Factor

This attribute contains the send interval for sync messages.

Attribute type: Unsigned16

Allowed values: 0 to 64 000

PTCP Takeover Timeout

This attribute contains a timeout value for detection the loss of sync messages from the current sync master. The sync slave shall try to find a new sync master. The PTCP takeover timeout value shall be in the range from 32 ms to 16 352 ms.

Attribute type: Unsigned32

Default value: 96 ms

PTCP Timeout

This attribute contains a timeout value for detection the loss of any sync messages. The PTCP timeout value shall be in the range from 32 ms to 16 352 ms.

Attribute type: Unsigned32

Default value: 192 ms

Role

This attribute contains the role to time synchronization.

Attribute type: Unsigned16

Allowed values: MASTER, SLAVE

Priority

This attribute shall be the priority of master issuing sync messages.

Attribute type: Unsigned8

Allowed values: PRIMARY, SECONDARY

Accuracy

This attribute shall be the accuracy of master time in comparison with the precision of the local clock.

Attribute type: Unsigned8

Allowed values: 25_ns, 100_ns, 250_ns, 1_μs, 2,5_μs, 10_μs, 25_μs, 100_μs, 250_μs, 1_ms, UNKNOWN

Default value: UNKNOWN

Variance

This attribute shall be the value for the quality of the clock. Each master clock shall maintain an estimation of its inherit precision. The PTCP variance is based on the theory of Allan deviation. The Allan deviation is $\sigma_y(\tau)$ is estimated as follows:

$$\sigma_y(\tau) = \left[\frac{1}{2(N-2)\tau^2} \times \sum_{k=1}^{N-2} (x_{k+2} - 2x_{k+1} + x_k)^2 \right]^{\frac{1}{2}} \quad (16)$$

where

$\sigma_y(\tau)$ is the Allan deviation.

x_k, x_{k+1} and x_{k+2} are time residual measurements, made at times $t_k, t_k + \tau$ and $t_k + 2\tau$, between the time provided by the measured clock and a local reference clock.

N is the number of data samples.

The PTCP variance is defined by:

$$\sigma_{PTCP}^2 = \tau^2 \times \frac{1}{3} \sigma_y^2 \quad (17)$$

where

σ_{PTCP}^2 is PTCP variance.

σ_y is the Allan deviation.

τ is the quantity

An unbiased estimate of the PTCP variance shall be computed as follows:

$$\sigma_{PTCP}^2 = \frac{1}{3} \left[\frac{1}{2(N-2)} \times \sum_{k=1}^{N-2} (x_{k+2} - 2x_{k+1} + x_k)^2 \right] \quad (18)$$

where

σ_{PTCP}^2 is the unbiased estimate of the PTCP variance

x_k, x_{k+1} and x_{k+2} are time residual measurement, made at times $t_k, t_k + \tau$ and $t_k + 2\tau$, between the time provided by the measured clock and a local reference clock

N is the number of data samples

For PTCP variance the quantity τ , the sample period, shall be the value defined as sync interval.

PTCP variances shall be represented as follows:

An estimate of the variance, σ_{PTCP}^2 , is computed in units of seconds squared.

The logarithm to the base 2 of this estimate is computed. The computation of the logarithm need not be more precise than the precision of the estimate of the variance.

The logarithm is multiplied by 28 to produce a scaled value.

The value is represented as a 2's complement Integer16. The value 0x8000 is added to the reported value represented in this form and any overflow is ignored. The result, i.e., the offset scaled reported value, is cast as an Unsigned16.

This offset scaled value, represented as Unsigned16, shall be the value of the log variances.

The largest possible positive number, 0xFFFF, for the offset scaled log variance attribute shall indicate that the variance is either too large to be represented, or has not been computed. The variance values are used in the selection of the best master clock.

Attribute type: Unsigned16

NOTE 1 For example, suppose the PTCP variance value is $1\,414 \times 2^{-73} = 1\,497 \times 10^{-22} \text{ s}^2$. Then, $\log_2(1\,414 \times 2^{-73}) = -73 + 0,5 = -72,5$. If this were expressed as an Integer16, it would truncate to -72. To retain some precision the value is scaled by 2^8 to yield a scaledLogVariance of -18 560, 0xB780, which retains 8 bits more precision. To this is added 0x8000 to yield the offset scaled reported value 0x3780.

NOTE 2 The smallest variance that can be represented is 2^{-128} or $\sim 3 \times 10^{-39} \text{ s}^2$, which results in an offset scaled log variance of 0x0000. The maximum variance that can be represented is $\sim 2^{+127,99609}$, which results in an offset scaled log variance of 0xFFFF.

PTCP Master Startup Time

This attribute shall be the duration of the startup phase of the sync master. During this time the sync master shall check if there is a sync master with a higher clock quality in the sync domain.

Attribute type: Unsigned16

Allowed values: 0 to 300 s

Default value: 10 s

WORKING_CLOCK

This attribute consists of the following elements for the time representation:

PTCP Time

This attribute consists of the following elements for the PTCP time representation.

Epoch Number

This attribute shall be the current number of times the 32-bit Seconds clock has rolled over since start of PTCP time. The meaning is the "upper" part of the seconds to achieve a 48 bit value.

Attribute type: Unsigned16

Seconds

This attribute contains the current number of seconds since start of the current PTCP epoch number. The meaning is the "lower" part of the seconds to achieve a 48 bit value.

Attribute type: Unsigned32

Nano Seconds

This attribute contains the current number of nanoseconds.

Attribute type: Unsigned32

Allowed values: 0 to 999 999 999

Current UTC Offset

This attribute contains in the context of the WORKING_CLOCK the value zero.

Attribute type: Integer16

Allowed values: 0

Local Time

This attribute consists of a local time value in nanoseconds. This value shall be used for PTCP clock synchronization and shall be the basis for the cycle counter.

Attribute type: Unsigned64

PLL Window

This attribute shall characterize the synchronization accuracy. The deviation between the local time and the reference time issued by a master shall not exceed the value of the PLL Window.

Attribute type: Unsigned32

Sync State Info

This element consists of the following error attributes. The synchronization achieved the state synchronous if there is no error.

Jitter out of Boundary

This attribute identifies whether the necessary synchronization accuracy (inside or outside PLL Window) is achieved or not.

Attribute type: Boolean

No Sync Message Received

This attribute identifies whether synchronization messages have been received or not.

Attribute type: Boolean

Wrong PTCP Domain UUID

If synchronization messages with wrong Domain UUID have been received, this attribute will be set.

Attribute type: Boolean

Rate Compensation Factor

This attribute shall be used to adjust the bridging time from sync and delay messages.

Attribute type: Unsigned32

List of Ports

This attribute consists of the following elements:

PortID

This attribute identifies one Port of a bridge.

Attribute type: Unsigned8

Borderline

Both subsequent attributes shall contain the value ON for end-ports of a PTCP Domain. The value of the Ingress attribute shall be ON and the value for the Egress attribute shall be OFF for a port coupling two PTCP segments, for the hierarchical higher segment. The value of the Ingress attribute shall be OFF and the value for the

Egress attribute shall be ON for a port coupling two PTCP segments, for the hierarchical lower segment.

NOTE 3 It is intended to avoid transmission from PTCP sync and PTCP announce messages from a hierarchical lower segment to a higher segment within a PTCP domain

This attribute consists of the following elements:

Ingress

This attribute identifies the end-ports of a PTCP Domain.

Attribute type: Unsigned8

Allowed values: ON, OFF

Egress

This attribute identifies the end-ports of a PTCP Domain or a level / segment borderline.

Attribute type: Unsigned8

Allowed values: ON, OFF

Line Delay

This attribute contains the transmit latency from this port to the connected port.

Attribute type: Unsigned32

Rate Compensation Factor Peer

This attribute contains the ratio between local and remote crystal frequency of the device at the connected port.

Attribute type: Float64

List of OUIs

This attribute consists of the following elements:

OUI

This attribute is an Organizationally Unique Identifier as defined by IANA.

Attribute type: Octet[3]

Subtype

This attribute is an organizationally specific parameter valid in the context of the attribute OUI.

Attribute type: Unsigned8

Data Block

This attribute is an organizationally specific parameter valid in the context of the attributes OUI and Subtype.

Attribute type: UUID

6.3.11.3 PTCP service specification

6.3.11.3.1 Start bridge

The start bridge service creates an instance of the PTCP protocol machine. Table 74 shows the parameter of the service.

Table 74 – Start bridge

Parameter name	req	cnf
Argument	M	
Sync ID	M	
List of Port Parameter	U	
Port ID	M	
Borderline	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

List of Port Parameter

This optional list of parameter consists of the following parameters:

Port ID

This parameter contains the ID of the port which issues the delay request message

Borderline

This parameter contains the end-ports of a clock-domain

Result(+)

This parameter indicates that the service request succeeded.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(-)

This parameter indicates that the service request failed.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Error code

The parameter Error code contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: ENTRY_NOT_POSSIBLE, SYNC_ID_EXISTS

6.3.11.3.2 Start slave

The start slave service adds synchronization properties to the PTCP protocol machine with. Table 75 shows the parameter of the service.

Table 75 – Start slave

Parameter name	req	cnf
Argument	M	
SyncID	M	
Subdomain UUID	M	
Sync Send Factor	M	
PTCP Timeout Factor	M	
PLL Window	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Subdomain UUID

This parameter contains the UUID of the synchronization subdomain. The UUID is used for a logical grouping of PTCP clocks that synchronize each other using the PTCP protocol.

Sync Send Factor

This parameter contains the reception interval of sync messages.

PTCP Timeout Factor

This parameter contains a timeout value to detect the loss of sync messages.

PLL Window

This parameter contains the required synchronization accuracy. The deviation between the local time and the reference time issued by a grandmaster shall not exceed the value of the PLLWindow.

Result(+)

This parameter indicates that the service request succeeded.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(-)

This parameter indicates that the service request failed.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Error code

The parameter Error code contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: SLAVE_NOT_POSSIBLE, SLAVE_EXISTS

6.3.11.3.3 Start master

The start master service adds synchronization properties with the qualification to become synchronization master to the PTCP protocol machine. Table 76 shows the parameter of the service.

Table 76 – Start master

Parameter name	req	cnf
Argument	M	
Sync ID	M	
Subdomain UUID	M	
Sync Send Factor	M	
Sync Class	M	
Clock Stratum	M	
Current UTC Offset	M	
Epoch Number	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Subdomain UUID

This parameter contains the UUID of the synchronization subdomain. The UUID is used for a logical grouping of PTCP clocks that synchronize each other using the PTCP protocol.

Sync Send Factor

This parameter contains the send interval of sync messages.

Sync Class

This parameter contains the recommended role (primary, secondary) as a master.

Clock Stratum

This parameter contains the quality of the own time.

Current UTC Offset

This parameter contains the current offset to the UTC.

Epoch Number

This parameter contains the current number of times the 32-bit Seconds clock has rolled over since the begin of the PTCP epoch. The PTCP epoch began at 0 hours on 1. January 1970.

Result(+)

This parameter indicates that the service request succeeded.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(-)

This parameter indicates that the service request failed.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Error code

The parameter Error code contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: MASTER_NOT_POSSIBLE, MASTER_EXISTS, MASTER_NOT_EXISTS

6.3.11.3.4 Stop bridge

This service shall be used to stop the PTCP protocol machine. Table 77 shows the parameter of the service.

Table 77 – Stop bridge

Parameter name	req	cnf
Argument	M	
Sync ID	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(+)

This parameter indicates that the service request succeeded.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(-)

This parameter indicates that the service request failed.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Error code

The parameter ErrCode contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: SYNC_ID_NOT_EXISTS

6.3.11.3.5 Stop slave

This service shall be used to stop the PTCP protocol machine. Synchronization will be stopped and bridge functionality remains. Table 78 shows the parameter of the service.

Table 78 – Stop slave

Parameter name	req	cnf
Argument	M	
Sync ID	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(+)

This parameter indicates that the service request succeeded.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(-)

This parameter indicates that the service request failed.

SyncID

This is the key attribute to identify the instance of the protocol machine.

Error code

The parameter Error code contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: SLAVE_NOT_EXISTS

6.3.11.3.6 Stop master

This service shall be used to reduce the properties of the PTCP protocol machine. Master functionality will be removed and the protocol machine is forced to takeover slave role. Table 79 shows the parameter of the service.

Table 79 – Stop master

Parameter name	req	cnf
Argument	M	
Sync ID	M	
Result(+)		S
Sync ID		M
Result(-)		S
Sync ID		M
Error code		M

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(+)

This parameter indicates that the service request succeeded.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Result(-)

This parameter indicates that the service request failed.

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Error code

The parameter Error code contains the error code of the specific error.

Attribute type: Unsigned16

Allowed Values: MASTER_NOT_EXISTS

6.3.11.3.7 Sync state change

This service shall be used to indicate a change of the synchronization state (i.e. synchronized or out of synchronization) and/or a change in the synchronization quality (i.e. jitter out of boundary, no sync message received). A change can only appear for a synchronization slave. Table 80 shows the parameter of the service.

Table 80 – Sync state change

Parameter name	ind
Argument	M
Sync ID	M
Sync State	M
Reason List	C

Argument

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

Sync ID

This is the key attribute to identify the instance of the protocol machine.

Sync State

This is the attribute shall contain the value SYNCHRONIZED or NOT_SYNCHRONIZED.

Reason List

This conditional attribute consists of the following elements:

ErrorType

This attribute identifies a synchronization error.

Attribute type: Unsigned16

Allowed Values: JITTER_OUT_OF_BOUNDARY, NO_SYNC_MESSAGE_RECEIVED, WRONG_PTCP_DOMAIN

Appear

This attribute identifies whether the error appears or disappears.

Attribute type: Boolean

Allowed Values: TRUE, FALSE

6.3.11.3.8 Line Delay change

This local service shall be used to indicate a change of certain Local Systems Data. Table 81 shows the parameters of the service.

Table 81 – Line Delay change

Parameter name	ind
Argument	M
Local Port ID	M
Local Line Delay	M

Argument

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

Local Port ID

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

Local Line Delay

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

6.3.11.3.9 PTCP behavior**6.3.11.3.9.1 Synchronization**

The synchronization is done with a sync- and an optional follow up-frame. The follow up-frame is necessary, if the hardware isn't able to add bridge and line delays on the fly.

The clock issuing the sync frames in a PTCP subdomain is called PTCP master.

Clocks which receive sync frames and adjust their time to the PTCP master are called PTCP slaves. Their synchronization accuracy depends on the accuracy of the time stamping units from each bridge which measures the line and bridge delays.

A bridge with the following enhancements is called a PTCP transparent clock:

- Line Delay measurement
- Bridge Delay measurement
- Rate compensation for line and bridge delays
- Support media redundancy mechanism for PTCP

6.3.11.3.9.2 Rate control compensation

The rate control compensation equalizes the clock rate of a PTCP transparent clock to the PTCP master. Due to this the accumulated bridge and line delays, received by the PTCP slave to achieve higher synchronization accuracy, are measured with the clock rate of the PTCP master.

A PTCP transparent clock uses sync frames to measure the clock rate relating to the PTCP master. For details, see IEC 61158-6-10, chapter precision working time control.

6.3.11.3.9.3 Best master algorithm

The Best Master Algorithm (BMA) specifies an algorithm how a clock determines the preferred clock issuing sync messages. The algorithm runs independently on each clock (master/slave) in a PTCP system. Each clock makes its own decision for the best master.

In computing the best master a clock shall:

- Consider the quality of the most recent sync messages received.
- Include the results of the previous best master computing. However if there are more recent qualified sync messages, the values from that messages shall be considered.

For analyzing the quality of sync message a clock shall consider:

- The master issuing the sync message belongs to the own subdomain.
- Receiving the sync messages periodically.
- The stratum of the master clock.
- The variance which describes the characteristic of the master clock.

If stratum and variance of different masters are equal the BMA use the Source-MAC-Address to decide for the preferred clock (smallest one wins).

6.3.11.4 Media Redundancy support for synchronization with PTCP

To improve the reliability of a peer-to-peer network redundant communication paths are necessary. PTCP sync messages use the redundant communication paths to be independent from the reconfiguration of redundant networks.

The forwarding algorithm for PTCP takes care that on each communication path only one sync message will be transmitted. The first received sync message shall be used for synchronization.

6.3.11.5 Invocation of the PTCP object

There is one PTCP object for each interface supporting PTCP.

6.3.12 IEEE 802.1AS ASE

6.3.12.1 Overview

The concepts according to IEEE 802.1AS shall be applied. This document defines the utilization of the IEEE 802.1AS.

The formal model of the IEEE 802.1AS ASE is presented by the IEEE 802.1AS class specification, containing a description of its attributes and services.

6.3.12.2 IEEE 802.1AS class specification

6.3.12.2.1 Template

An IEEE 802.1AS object is described by the following template:

ASE: IEEE 802.1AS ASE

CLASS: IEEE 802.1AS

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m) Key Attribute:	not used
2	(m) Attribute:	Role
3	(m) Attribute:	Time
4	(m) Attribute:	List of Ports
4.1	(m) Attribute:	PortID
4.2	(m) Attribute:	Path Delay Boundary

SERVICES:

1	(m) OpsService:	Local Get Time
2	(m) OpsService:	Local Time State Info

6.3.12.2.2 Attributes

This key attribute is not used.

Expected Role

This attribute identifies the expected role of the interface regarding time synchronization.

Attribute type: Unsigned8

Allowed values: OFF, MASTER, SLAVE

Time

This attribute holds the time and the status.

Attribute type: TimeStamp

List of Ports

This attribute consists of the following elements:

PortID

This attribute identifies one Port of a bridge.

Attribute type: Unsigned8

Path Delay Boundary

This attribute shall contain the value ON for end-ports of a 802.1AS Domain.

Attribute type: Unsigned8

Allowed values: ON, OFF

6.3.12.3 IEEE 802.1AS service specification

6.3.12.3.1 Local Get Time

This local service may be used to get the local time. Table 82 shows the parameters of the service.

Table 82 – Local Get Time

Parameter name	req	cnf
Argument	M	
Result(+)		S(=)
Time		M(=)
Result(-)		S(=)

Argument

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

Result(+)

This parameter indicates that the service request succeeded.

Time

This parameter contains the value of the equivalent ASE attribute.

Result(-)

This parameter indicates that the service request failed.

6.3.12.3.2 Local time state info

This local service shall be used to indicate a change of the local time state. This event is initiated whenever the time was set to a new value or if the time stamp's state information has changed.

If the time has been set to a new value, at least one of the fields SecondsHigh, SecodsLow or Nanoseconds differ between OldTime and NewTime. If only the field Status differs, the status has changed only (e.g. from "TimeStamp related to the global synchronized time" to "TimeStamp related to the local time"). Note also that both Status and value fields of the time stamps may change (e.g. when changing from "TimeStamp related to the local time (arbitrary time scale)" to "TimeStamp related to the global synchronized time"; e.g. when going to sync after power on).

Table 83 shows the parameters of the service.

Table 83 – Local time state info

Parameter name	ind
Argument	M
Grandmaster	M
Time domain	M
OldTime	M
NewTime	M

Argument

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

Grandmaster

This parameter indicates which Grandmaster is used as source of time.

Attribute type: OctetString

Time domain

This parameter indicates which time domain is used.

Attribute type: OctetString

OldTime

This parameter indicates the point of time in the former time scale, when the new time has been set or the time stamp's status changed.

Attribute type: TimeStamp

NewTime

This parameter indicates the value of the time, to which the time has been set or when the time stamp's status changed.

Attribute type: TimeStamp

6.3.12.4 Invocation of the IEEE 802.1AS object

There is one IEEE 802.1AS object for each interface.

6.3.13 Virtual Bridges ASE**6.3.13.1 Overview**

The concepts according to IEEE 802.1Q (MAC bridges) shall be applied. This document defines the utilization of the IEEE 802.1Q (Virtual bridged local area networks). It defines extensions for RT_CLASS_3 regarding prioritization of frames.

Application processes contain data that are transmitted in different priorities according IEEE 802.1Q. Furthermore, the usage of these priorities shall be extended in different time intervals according to this specification. The Virtual Bridges ASE defines attributes to determine the timing behavior.

The formal model of the Virtual Bridges ASE is presented by the Virtual Bridges class specification, containing a description of its attributes.

6.3.13.2 Virtual Bridges class specification**6.3.13.2.1 Template**

An IEEE 802.1Q object is described by the following template:

ASE: Virtual Bridges ASE

CLASS: Virtual Bridges

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m) Key Attribute: not used
2	(o) Attribute: Network Control High Queue
3	(m) Attribute: Network Control Low Queue
4	(m) Attribute: Prio 7 Send Queue
5	(m) Attribute: Prio 6
5.1	(o) Attribute: Red Send Queue
5.2	(m) Attribute: Prio 6 Send Queue
6	(m) Attribute: Prio 5 Send Queue
7	(m) Attribute: Prio 4 Send Queue
8	(m) Attribute: Prio 3 Send Queue
9	(m) Attribute: Prio 2 Send Queue
10	(m) Attribute: Prio 1 Send Queue

11	(m) Attribute:	Prio 0 Send Queue
12	(m) Attribute:	Transmission Period
13	(o) Attribute:	Clock Domain Phase Counter

6.3.13.2.2 Attributes

For details on the usage and meaning of the attributes see IEC 61158-6-10, Annex I.

This key attribute is not used.

Network Control High Queue

This optional attribute contains the list of DLPDUs for synchronization for those DLPDUs which are not planned within IR Data records or for those received outside the planned time. After transmission the DLPDU shall be removed from the queue.

Network Control Low Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q organizational priority for management protocols like LLDP, PTCP line delay measurement, MRP and others. After transmission the DLPDU shall be removed from the queue.

Prio 7 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 7. After transmission the DLPDU shall be removed from the queue.

Prio 6

This attribute consists of the following attributes:

Red Send Queue

This optional attribute contains the list of DLPDUs for RT_CLASS_3 which were planned within a IR Data record and received in time. The sequence of frames and the exact sending time shall be determined by other ASE attributes. After transmission the DLPDU shall be removed from the queue. With the start of a new phase, the Red Send Queue shall be filled with the DLPDUs for the current phase.

NOTE 1 The application layer for decentralized periphery defines a list of IR Frame Data elements to determine the sequence and exact sending time of frames.

Prio 6 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 6 including DLPDUs for RT_CLASS_1 and RT_CLASS_UDP. After transmission the DLPDU shall be removed from the queue.

Prio 5 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 5. After transmission the DLPDU shall be removed from the queue.

Prio 4 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 4. After transmission the DLPDU shall be removed from the queue.

Prio 3 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 3. After transmission the DLPDU shall be removed from the queue.

Prio 2 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 2. After transmission the DLPDU shall be removed from the queue.

Prio 1 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 1. After transmission the DLPDU shall be removed from the queue.

Prio 0 Send Queue

This attribute contains the list of DLPDUs for IEEE 802.1Q priority 0. After transmission the DLPDU shall be removed from the queue.

Transmission Period

This attribute contains the value of the current transmission period of the port. It is aimed to control the usage of the bandwidth within the network.

NOTE 2 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, a fair mechanism to restrict the sending performance of each device is provided.

Within the RED period, only DLPDUs from the Red Send Queue shall be transmitted.

Within the GREEN period, DLPDUs from all queues shall be transmitted according to their priorities.

Within the GREEN period, for the local scheduling the following rules shall be applied:

- The local queuing process shall drop all RT_CLASS_1 frames for the next phase if at the beginning of a new GREEN period the Prio 6 Send Queue is not empty. Each drop shall be signaled to the local application.
- In average, not more than 90 percent of bandwidth shall be used to avoid overload of the network.

Within the YELLOW period, DLPDUs from all queues shall be transmitted according to their priorities if the transmission of the frame ends within this period. If the calculated transmission time for a frame exceeds the period, the first frame with lesser priority can be tested optionally.

The sequence of periods shall be RED, GREEN, and YELLOW. The RED period may be omitted. The YELLOW period shall exist if RED exists and shall be omitted otherwise.

The transmission period shall be controlled by a local clock. This clock shall be synchronized over the network using GREEN periods. The duration of all periods shall not exceed the time value calculated by means of the attribute Send Clock Factor, which shall be defined by other ASE. The accuracy is determined by conformance classes.

Attribute type: Unsigned8

Allowed Values: RED (optional), GREEN, YELLOW (optional)

As shown in Figure 5, the sending of frames is divided into different periods at the local port. Each phase is defined by $T_{\text{Sendclock}}$ which is between $31,25 \mu\text{s}$ and 4 ms . The number of frames of each period depends on the scheduled frames for the current phase and may be also zero.

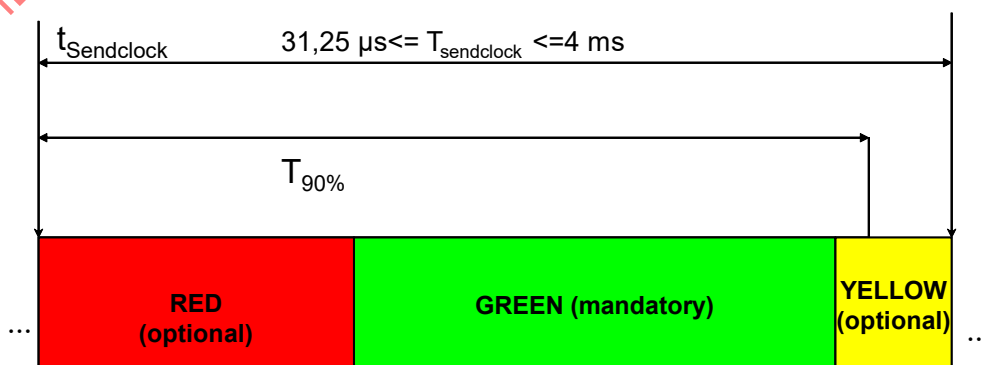


Figure 5 – Example of periods at a local port

The first and the last invocation of the RED period shall be synchronized between the local and the neighborhood port by means of Remote Systems Data Change (Remote RT_CLASS_3 Port Status).

Clock Domain Phase Counter

This attribute contains the count of passed phases. All nodes used with RT_CLASS_3 shall use the same value. It is gained by PTCP clock synchronization.

Attribute type: Unsigned64

6.3.13.3 Virtual Bridges service specification

There are no services specified.

6.3.13.4 Invocation of the Virtual Bridges object

There is one Virtual Bridges object for each port.

6.3.14 MAC Bridges ASE

6.3.14.1 Overview

The MAC Bridges ASE controls the bridging according to IEEE 802.1Q (MAC bridges) extended with the special requirements for RT_CLASS_3 and additionally uses the Virtual Bridges ASE for each port.

The concepts according to IEEE 802.1Q shall be applied. Subclause 6.3.14 defines the utilization of the IEEE 802.1Q regarding MAC bridges.

The bridging mechanisms of IEEE 802.1Q shall be temporary disabled within the RED phase. In this case, the bridging of frames out of Frame ID range 3 (see IEC 61158-6-10) shall be done according to the attributes defined by this ASE with the behavior according to the RT_CLASS_3 Relay protocol machine. The values of the attributes define a special forwarding table.

6.3.14.2 MAC Bridges class specification

6.3.14.2.1 General

The MAC Bridges ASE defines one MAC Bridge object type:

6.3.14.2.2 Template

A MAC Bridge object is described by the following template:

ASE: MAC Bridges ASE

CLASS: MAC Bridges

CLASS ID: not used

PARENT CLASS: top

ATTRIBUTES:

1	(m) Key Attribute: Implicit
2	(m) Attribute: PD IR Global Data
2.1	(m) Attribute: IR Data UUID
2.2	(m) Attribute: Max Bridge Delay
2.3	(m) Attribute: Number of Ports
2.4	(m) Attribute: List of Port Delays
2.4.1	(m) Attribute: Max Port Tx Delay
2.4.2	(m) Attribute: Max Port Rx Delay
2.4.3	(m) Attribute: Max Line Rx Delay

2.4.4	(m) Attribute:	Yellow Time
3	(o) Attribute:	PD IR Frame Data
3.1	(m) Attribute:	Frame Data Properties
3.2	(o) Attribute:	List of PD IR Frame Data Elements
3.2.1	(m) Attribute:	Frame Send Offset
3.2.2	(m) Attribute:	Data Length
3.2.3	(m) Attribute:	Reduction Ratio
3.2.4	(m) Attribute:	Phase
3.2.5	(m) Attribute:	Frame ID
3.2.6	(m) Attribute:	Ethertype
3.2.7	(m) Attribute:	Rx Port
3.2.8	(m) Attribute:	Frame Details
3.2.8.1	(m) Attribute:	Meaning Frame Send Offset
3.2.9	(m) Attribute:	Tx Port Group
3.2.9.1	(m) Attribute:	Number Of Tx Port Group Array Elements
3.2.9.2	(m) Attribute:	Tx Port Group Array
4	(m) Attribute:	PD IR Begin End Data
4.1	(m) Attribute:	Red Guard
4.1.1	(m) Attribute:	Start of Red Frame ID
4.1.2	(m) Attribute:	End of Red Frame ID
4.2	(m) Attribute:	Number of Ports
4.3	(m) Attribute:	List of Ports
4.3.1	(m) Attribute:	Number of Assignments
4.3.2	(m) Attribute:	List of Assignments
4.3.2.1	(m) Attribute:	Tx Begin End Assignment
4.3.2.1.1	(m) Attribute:	Red Period Begin
4.3.2.1.2	(m) Attribute:	Green Period Begin
4.3.2.2	(m) Attribute:	Rx Begin End Assignment
4.3.2.2.1	(m) Attribute:	Red Period Begin
4.3.2.2.2	(m) Attribute:	Green Period Begin
4.3.3	(m) Attribute:	Number of Phases
4.3.4	(m) Attribute:	List of Phases
4.3.4.1	(m) Attribute:	Tx Phase Assignment
4.3.4.1.1	(m) Attribute:	Assigned Value For Reserved Begin
4.3.4.1.2	(m) Attribute:	Assigned Value For Reserved End
4.3.4.2	(m) Attribute:	Rx Phase Assignment
4.3.4.2.1	(m) Attribute:	Assigned Value For Reserved Begin
4.3.4.2.2	(m) Attribute:	Assigned Value For Reserved End
5	(m) Attribute:	List of Ports
5.1	(m) Attribute:	Port
5.1.1	(m) Attribute:	Transmission Period Sending
5.1.2	(m) Attribute:	Transmission Period Receiving
5.1.3	(m) Attribute:	Local Port State
5.1.4	(m) Attribute:	Local RTClass 3 Port State
6	(o) Attribute:	Clock Domain Phase Counter

SERVICES:

1	(m) OpsService:	Port State Change
2	(o) OpsService:	Set Port State
3	(o) OpsService:	Flush filtering data base

6.3.14.2.3 Attributes**Implicit**

The key attribute Implicit indicates that the MAC Bridge object is implicitly addressed by the service.

PD IR Global Data

This attribute contains the global IR data used by the engineering system as computation base. The device shall adjust its physical characteristics to match the global data. The attribute is composed of the following elements:

IR Data UUID

This attribute contains the unique identifier of the isochronous realtime project which is configured by the engineering system.

Attribute type: UUID

Max Bridge Delay

This attribute indicates the maximum bridge delay of the interface in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000 ("unknown"), 0x00000001 to 0x3B9AC9FF

Number of Ports

This attribute indicates the number of ports the List of Port Delays contains.

Attribute type: Unsigned32

Allowed Values: 0x00000001 to 0x000000FF

List of Port Delays

This list contains the port transmit and receive delays for each port, starting with Port 1 up to the port number indicated by the attribute Number of Ports. A list element is composed of the following elements:

Max Port Tx Delay

This attribute indicates the maximum port transmission delay in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000 ("unknown"), 0x00000001 to 0x3B9AC9FF

Max Port Rx Delay

This attribute indicates the maximum port receive delay in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000 ("unknown"), 0x00000001 to 0x3B9AC9FF

Max Line Rx Delay

This attribute indicates the maximum line receive delay in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000 ("no delay, forward as soon as possible"), 0x00000001 to 0x3B9AC9FF

Yellow Time

This attribute indicates the yellow time in multiples of 1 ns.

The yellow time defines the length of the YELLOW period. It allows a bridge to optimize the forwarding of frames belonging to the protocols PTCP, MRP, RT and IEEE 802.1AS.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00001AE0 to 0x0001E848

PD IR Frame Data

This attribute contains the IR frame data. The attribute is composed of the following elements:

Frame Data Properties

This attribute contains of the following elements:

Forwarding Mode

This attribute describes the settings for the forwarding mode being used for RT_CLASS_3 frames.

Attribute type: Unsigned8

The allowed values are shown in Table 84:

Table 84 – Allowed values of Forwarding Mode

Value(decimal)	Meaning
0	Absolute mode The FrameSendOffset contains the sending time for a forwarded frame
1	Relative mode The FrameSendOffset shall contain the sending time of a forwarded frame or the "Best effort" value.

Fast Forwarding Multicast MAC Add

This attribute describes the settings for the destination address being used for RT_CLASS_3 frames.

Attribute type: Unsigned8

The allowed values are shown in Table 85:

Table 85 – Allowed values of Fast Forwarding Multicast MAC Add

Value(decimal)	Meaning
0	Use interface MAC destination unicast address (legacy)
1	Use RT_CLASS_3 destination multicast address
2	Use Fast Forwarding Multicast MAC address

List of PD IR Frame Data Elements

This list contains information for all frames in the frame ID range 3 (see IEC 61158-6-10). This covers path information (i.e. on which ports a particular frame is to be forwarded) and time information (i.e. when a particular frame is to be sent).

The path information is described by the attribute Tx Port Group, the time information by Frame Send Offset.

Typically the Frame Send Offset is set to a value composed by the engineering system. A special case is setting Frame Send Offset to 0, indicating that the frame is to be sent “as fast as possible”; i.e. within RED the frame will leave the device with a slight time variation. For a frame fed in by the device, the Frame Send Offset indicated here has to match the Frame Send Offset as indicated in the Communication Relationship class.

If the forwarding mode is set to “absolute mode”, the device acts as absolute forwarder. In this case the list contains all frames which are injected, received and forwarded by the device.

If the forwarding mode is set to “relative mode”, the device acts as a relative forwarder. In this case the list contains the frames injected and received by the device. The rules for frames to be forwarded are as follows.

If a particular frame with a Frame ID in the RT_CLASS_3 singlecast range is received on some port and the device is not end point for it, the frame is transmitted with a constant delay on the other port.

If a particular frame with a Frame ID in the RT_CLASS_3 multicast range is received on some port, the frame is transmitted with a constant delay on the other port; independent whether the device is end point for it or not.

Since frames are forwarded “on the other port”, the relative mode is only applicable for a two port device.

The list is composed of the following list elements:

Frame Send Offset

This attribute indicates the offset relative to the beginning of the related send clock cycle when the particular frame shall be sent in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000 to 0x003D08FF

NOTE 1 The minimal planned gap between two frames out of frame ID range 3 (see IEC 61158-6-10) is 1120 ns.

Data Length

This attribute contains the overall length of the transmitted Data (CSDU or else CSFSDU Length); thus for a dynamically packed frame it indicates the length of all subframes including the subframe this IO device contributes.

Attribute type: Unsigned16

Allowed Values for RT_CLASS_3: 40 to 1 440

NOTE 2 The lower bound of the C_SDU length is derived from the minimum frame size of an Ethernet frame (= 64 octets). On subtracting the size of the header (i.e. destination address, source address and type) and the size of the footer (i.e. FCS) 46 octets are left. On subtracting the size of the RT frame header (i.e. Frame ID) and footer (i.e. cycle counter, data status and transfer status) the minimum length of the C_SDU is 40 octets.

Reduction Ratio

This attribute contains the reduction ratio of the frame compared to the send clock.

Attribute type: Unsigned16

The allowed values are shown in IEC 61158-6-10.

Phase

This attribute indicates the phase when the particular frame shall be sent.

Attribute type: Unsigned16

Allowed Values: 1 until current value of the attribute Reduction Ratio

Frame ID

This attribute contains the identifier of the particular frame which is configured by the engineering system.

Attribute type: Unsigned16

The allowed values are shown in table “FrameID range 3” in IEC 61158-6-10.

Ethertype

This attribute indicates the Ethertype to use for the frame.

Attribute type: Unsigned16

Allowed Values: 0x8892

Rx Port

This attribute contains the number of the receiving port. The value LOCAL_INJECTION shall be used for frames received by this device. The value PORT_1 to PORT_255 shall be used for frames to be forwarded.

Attribute type: Unsigned8

Allowed Values: LOCAL_INJECTION, PORT_1 to PORT_255

Frame Details

This attribute consists of the following elements:

Meaning Frame Send Offset

This attribute indicates the detailed meaning of the attribute Frame Send Offset.

The attribute shall be set to TX_FRAME, when the attribute Frame Send Offset indicates the time, when a frame out of frame ID range 3 (see IEC 61158-6-10) is to be transmitted.

NOTE 3 A frame is locally received, when the attribute Rx Port is set to LOCAL_INJECTION.

Attribute type: Unsigned8

Allowed Values: TX_FRAME

Tx Port Group

This attribute indicates where the particular frame shall be transmitted. It consists of the following elements:

Number Of Tx Port Group Array Elements

This field shall count the succeeding Tx Port Group Array entries.

Attribute type: Unsigned8

Allowed values: 1, 3, 5, 7, 9, 11, 13, 15, ..., 31, 33.

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:

$$N = M_{\text{highest}} \text{ DIV } 8 + 1 \quad (19)$$

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 unused TxPort entries shall be set to zero.

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 4 The term DIV stands for division without remainder. The term MOD stands for the remainder of the division.

Bit 0: TxPortEntry_0

This bit shall be set with the values according to Table 86 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 86 – 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. Thus local injection is mapped onto port 0.

Bit 1: TxPortEntry_1

This bit shall be set with the values according to Table 86 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 86 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 86 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 86 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 86 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 86 if the TxPort with the number that meets $m \text{ MOD } 8 = 6$ 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 86 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 $1 \leq m \leq N$.

PD IR Begin End Data List

This list contains the Red Guard and the bandwidth administration information for the traffic classes RED and GREEN. It is composed of the following list elements:

Red Guard

This attribute controls the behavior of the receiving ports on Realtime Class 3 Port State Machine (RTC3PSM) being in state “Run”, i.e. the Rx port is in mode RED.

The Red Guard prevents wrong frames of entering the traffic class RED as a first filtering layer. All frames with a frame ID below Start of Red Frame ID or above End of Red Frame ID are blocked during the RED period.

The second filtering layer is provided by the IR Frame Data Elements. Only a frame crossing both filters is entering the traffic class RED. If the IR Frame Data is missing, the second filtering layer is omitted.

Start of Red Frame ID

This attribute defines the lower boundary of the Red Guard. The value shall be set to a lower value than the content of the attribute End of Red Frame ID.

Attribute type: Unsigned16

Allowed values: 0x0080 to 0x7FFF

Default value: 0x0100

End of Red Frame ID

This attribute defines the upper boundary of the Red Guard.

Attribute type: Unsigned16

Allowed values: 0x0080 to 0x7FFF

Default value: 0x7FFF

Number of Ports

This attribute contains the number of elements in the List of Ports.

Attribute type: Unsigned32

Allowed Values: 0x00000001 to 0x000000FF

List of Ports

This list contains the bandwidth administration information. It is composed of the following list elements.

Number of Assignments

This attribute contains the number of elements in the List of Assignments. A device has to support at a minimum 5 different assignments (four for phases with reserved periods (RED) and one for phases without any reserved period).

Within the assignments at a minimum up to two different Red Period Begin values (one of them for phases without any reserved period) and up to five different Green Period Begin values (one of them for phases without any reserved period) shall be supported.

A phase without any reserved period is indicated by having Red Period Begin and Green Period Begin set to 0x0.

The number of assignments supported by the device is indicated by GSD.

Attribute type: Unsigned32

Allowed Values: 0x00000005 to 0x00000010

List of Assignments

This list contains the assignments. A particular list element contains bandwidth information. A list element is composed of the following attributes:

Tx Begin End Assignment

This attribute defines the RED and GREEN period for Tx.

Red Period Begin

This attribute indicates the offset relative to the beginning of the send clock cycle when the RED period shall begin in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000

Green Period Begin

This attribute indicates the offset relative to the beginning of the send clock cycle when the GREEN period shall begin in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000 to 0x003D08FF

Rx Begin End Assignment

This attribute defines the RED and GREEN period for Rx.

Red Period Begin

This attribute indicates the offset relative to the beginning of the send clock cycle when the RED period shall begin in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: 0x00000000

Green Period Begin

This attribute indicates the offset relative to the beginning of the send clock cycle when the GREEN period shall begin in multiples of 1 ns.

Attribute type: Unsigned32

Time base: 1 ns

Allowed Values: shall be set to the end of the RED period.

Number of Phases

This attribute contains the number of elements in the List of Phases used for assignments. Up to 16 different phase assignments are supported.

A particular frame's phase is mapped onto the assignment phase by equation (20)

$$\text{Assignment Phase} = \text{Frame Phase} \text{ MOD } \text{Number of Phases.} \quad (20)$$

where

Assignment Phase is the assignment phase
Frame Phase is the particular frame phase
Number of Phases is the number of elements in the List of Phases

Attribute type: Unsigned32

Allowed Values: 1, 2, 4, 8, 16

List of Phases

This attribute assigns to a particular phase the appropriate period begins.

For the assignment of the particular begins of that phase, the rules as stated in Table 87 apply.

Table 87 – Dependencies of RedPeriodBegin and GreenPeriodBegin

Value (hexadecimal)	Meaning
RedPeriodBegin < GreenPeriodBegin	Only a RED period is defined
RedPeriodBegin = GreenPeriodBegin = 0	No reserved period is defined
RedPeriodBegin > GreenPeriodBegin	Not used

Tx Phase Assignment

This attribute assigns the values of the RED and GREEN period on all Tx ports for a particular phase.

Assigned Value For Reserved Begin

This attribute contains the positional number of a particular element in the List of Assignments. The content of the attribute Red Period Begin within element's attribute Tx Begin End Assignment is used as begin of the RED period on the Tx port.

Attribute type: Unsigned8

Allowed Values: 0 to Number of Assignments - 1

Assigned Value For Reserved End

This attribute contains the positional number of a particular element in the List of Assignments. The content of the attribute Green Period Begin within element's attribute Tx Begin End Assignment is used as begin of the GREEN period on the Tx port.

Attribute type: Unsigned8

Allowed Values: 0 to Number of Assignments - 1

Rx Phase Assignment

This attribute assigns the values of the RED and GREEN period on all Rx ports for a particular phase.

Assigned Value For Reserved Begin

This attribute contains the positional number of a particular element in the List of Assignments. The content of the attribute Red Period Begin within element's attribute Rx Begin End Assignment is used as begin of the RED period on the Rx port.

Attribute type: Unsigned8

Allowed Values: 0 to Number of Assignments - 1

Assigned Value For Reserved End

This attribute contains the positional number of a particular element in the List of Assignments. The content of the attribute Green Period Begin within element's attribute Rx Begin End Assignment is used as begin of the GREEN period on the Rx port.

Attribute type: Unsigned8

Allowed Values: 0 to Number of Assignments - 1

List of Ports

This attribute shall consist of the following elements:

Port

This attribute consists of the following elements:

Transmission Period Sending

This attribute contains the value of the current transmission period of the port in sending direction. The scheduling of the sending process according IEEE 802.1Q ASE shall be applied.

The following transmitting rules for RED shall be applied:

The local queuing process shall transmit a frame according to Frame Send Offset and Tx Port Group Array if the calculation of Phase and Reduction Ratio fits Phase Number.

The following forwarding rules for RED shall be applied:

The local queuing process shall transmit a prior received frame according to Frame Send Offset, Data Length, Frame ID, and Rx Port. If the calculation of Phase and Reduction Ratio fits Phase Number, then the frame shall be transmitted on all ports stored in Tx Port Group Array.

Within the YELLOW period the MAC bridge forwarding rules of IEEE 802.1Q shall be omitted.

The following transmitting rules shall be applied:

The local queuing process shall transmit a frame according if its size is shorter than the remaining YELLOW interval.

The following forwarding rules shall be applied:

The local queuing process shall transmit a prior received frame if its size is shorter than the remaining YELLOW interval.

Within the GREEN period, the MAC Bridge forwarding rules shall be applied.

The transmission period shall be controlled by a local clock. This clock shall be synchronized over the network using GREEN periods. The duration of all periods shall not exceed the time value calculated by means of the attribute Send Clock Factor, which shall be defined by other ASE. The accuracy is determined by conformance classes.

Attribute type: Unsigned8

Allowed Values: RED (optional), GREEN, YELLOW (optional)

NOTE 5 As shown in Figure 5, the sending of frames is divided into different periods at the local port. Each phase is defined by $T_{Sendclock}$ which is between 31,25 μ s and 4 ms. The number of frames of each period depends on the scheduled frames for the current phase and may be also zero.

Transmission Period Receiving

This attribute contains the value of the current transmission period of the port in receiving direction.

Within the RED period, only in time RT_CLASS_3 DLPDUs and in time PTCP Sync DLPDUs shall be forwarded according RED Relay rules. All other received DLPDUs shall be handled by the MAC Relay according to the forwarding rules of IEEE 802.1Q.

The following receiving rules for RED shall be applied:

The local queuing process shall receive a frame according to Frame Send Offset, Data Length, Frame ID, Rx Port and the calculation of Phase and Reduction Ratio fits Phase Number.

$$\text{Phase} := (\text{Clock Domain Phase Counter MOD Reduction ratio}) + 1 \quad (21)$$

where

<i>Phase</i>	is the phase value
<i>Clock Domain Phase Counter</i>	is the clock domain phase counter
<i>Reduction ratio</i>	is the reduction ratio

The received IEEE 802.3 source MAC addresses shall not be integrated in the forwarding address table to avoid influencing the transmission of all other frames.

The following receiving rules shall be applied:

Within the GREEN period, the IEEE 802.1Q (MAC Relay) shall be applied.

The transmission period shall be controlled by a local clock. This clock shall be synchronized over the network using GREEN periods. The duration of all periods shall not exceed the time value calculated by means of the attribute Send Clock Factor, which shall be defined by other ASE. The accuracy is determined by conformance classes.

Attribute type: Unsigned8

Allowed Values: RED (optional), GREEN

Local Port State

This attribute contains the value for the port according IEEE 802.1Q bridge management.

Attribute type: Unsigned8

Allowed Values: DISCARDING, BLOCKED, FORWARDING

Local RTClass 3 Port State

This attribute contains the value for the port according to the RED phase. The value OFF means not used or configured for RED phase or the preconditions could not be reached so far. The value RTCLASS3_UP means that the transmission direction is in the RED phase. The value RTCLASS3_RUN means that both directions are in the RED phase.

Attribute type: Unsigned8

Allowed Values: OFF, RTCLASS3_UP, RTCLASS3_RUN

Clock Domain Phase Counter

This attribute contains the count of passed phases. All nodes used with RT_CLASS_3 shall use the same value. It is gained by PTCP clock synchronization.

Attribute type: Unsigned64

6.3.14.3 MAC Bridge service specification

6.3.14.3.1 Port state change

This local service shall be used to indicate a change of certain Local Systems Data. Table 88 shows the parameters of the service.

Table 88 – Port state change

Parameter name	ind
Argument	M
Local Port ID	M
Local Port State	M
Local RTClass 3 Port State	M

Argument

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

Local Port ID

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

Local Port State

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

Local RTClass 3 Port State

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

6.3.14.3.2 Set port state

This local service shall be used to set a certain Local Systems Data. Table 89 shows the parameters of the service.

Table 89 – Set port state

Parameter name	req
Argument	M
Local Port ID	M
Local Port State	U
Local RTClass 3 Port State	U

Argument

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

Local Port ID

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

Local Port State

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

Local RTClass 3 Port State

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

6.3.14.3.3 Flush filtering data base

This local service shall be used to flush the local filtering data base. Table 90 shows the parameters of the service.

Table 90 – Flush filtering data base

Parameter name	req
Argument	M

Argument

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

6.3.14.4 Invocation of the MAC Bridge object

For the invocation of the MAC Bridge object the following rules apply:

- There is one MAC Bridge object for each interface
- The services with the IFW prefix shall exist for each RT_CLASS_3 interface

6.3.15 Fragmentation ASE

6.3.15.1 Overview

The length of a transmission period is limited on the lower end by the maximum size of an Ethernet frame. The fragmentation reduces, with a peer to peer approach, the conveyed size of an Ethernet frame and allows to use a LengthOfPeriod down to 31,25 μ s.

The fragmentation ASE hides the issue fragmentation from the upper layers by acting as an intermediate application service entity and providing the same service interface as the IEEE 802.3 ASE.

For this the Fragmentation ASE is derived from the IEEE 802.3 ASE and overwrites the operational service M_UNITDATA to hide the fragmentation specific details from the upper layers.

Frames to be transmitted which are subject for fragmentation will get fragmented by the operational service and transmitted in fragments by using the basic M_UNITDATA service of the IEEE 802.3 ASE. Fragmented frames are received by the IEEE 802.3 ASE, collected together to restore the original frame and then indicated by the M_UNITDATA indication to the upper layers.

The frames related to the protocols PTCP, MRP, RTC (RT_CLASS_1, RT_CLASS_2 and RT_CLASS_3) and IEEE 802.1AS are never fragmented.

The fragmentation ASE is optional; it shall be supported, if the residual time for NRT frames drops below 125 μ s.

6.3.15.2 Fragmentation class specification

6.3.15.2.1 General

The Fragmentation ASE defines one Fragmentation object type.

6.3.15.2.2 Template

A Fragmentation object is described by the following template:

ASE: Fragmentation ASE

CLASS: Fragmentation

CLASS ID: not used

PARENT CLASS: IEEE 802.3 ASE

ATTRIBUTES:

1 (m) Key Attribute: Implicit

SERVICES:

1 (m) OpsService: M_UNITDATA

6.3.15.2.3 Attributes

Implicit

The key attribute Implicit indicates that the Fragmentation object is implicitly addressed by the service.

6.3.15.3 Fragmentation service specification

6.3.15.3.1 M_UNITDATA

For the definition of this service see IEEE 802.3.

This local service shall be used to transmit (M_UNITDATA.req) or receive (M_UNITDATA.ind) a PDU.

6.3.15.4 Invocation of the Fragmentation object

There is one Fragmentation object for each interface.

6.3.16 IEEE 802.3 ASE

6.3.16.1 Overview

The IEEE 802.3 ASE controls the medium access.

The concepts according to IEEE 802.3 (Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications) shall be applied. Subclause 6.3.16 defines the utilization of the IEEE 802.3.

6.3.16.2 IEEE 802.3 class specification

6.3.16.2.1 General

The IEEE 802.3 ASE defines one IEEE 802.3 object type.

6.3.16.2.2 Template

An IEEE 802.3 object is described by the following template:

ASE: IEEE 802.3 ASE

CLASS: IEEE 802.3

CLASS ID: not used

PARENT CLASS: top

ATTRIBUTES:

1 (m) Key Attribute: Implicit

2 (m) Attribute: List of Ports

2.1 (m) Attribute: Port

2.1.1 (m) Attribute: MAU Type

2.1.2 (o) Attribute: MAU Type Extension

2.1.3 (m) Attribute: Autonegotiation State

2.1.4 (m) Attribute: Link State

SERVICES:

- 1 (m) OpsService: M_UNITDATA
- 2 (m) OpsService: MAU Type Change
- 3 (m) OpsService: Set MAU Type

6.3.16.2.3 Attributes**Implicit**

The key attribute Implicit indicates that the IEEE 802.3 object is implicitly addressed by the service.

List of Ports

This attribute shall consist of the following elements:

Port

This attribute consists of the following elements:

Attribute type: Unsigned8

MAU Type

This attribute contains the value of the real MAU type according IETF RFC 4836. If the value of the attribute Link State is DOWN or OFF, the MAU type shall be zero.

Attribute type: Unsigned16

MAU Type Extension

This optional attribute defines an extension to the real IEEE 802.3 MAUType. The extension is used to specify additional physics not covered by IEEE 802.3. The information derived from the IEEE 802.3 MAU Type (duplexity, speed, and media) shall however still be valid. If the value of the attribute Link State is DOWN or OFF, the MAU Type Extension shall be zero.

The absence of the attribute resp. its presence with value 0x0 indicates that the MAU Type is not extended.

Attribute type: Unsigned16

Autonegotiation State

This attribute contains the value of the real autonegotiation setting.

Attribute type: Unsigned8

Allowed Values: ENABLE, DISABLE

Link State

This attribute contains the value of the real port status setting. It shall be OFF, if the media attachment unit is disabled. It shall be DOWN, if the media attachment unit is enabled but no Link is detected. It shall be UP, if the media attachment unit is enabled and a Link is detected.

Attribute type: Unsigned8

Allowed Values: see IEC 61158-6-10

6.3.16.3 IEEE 802.3 service specification**6.3.16.3.1 M_UNITDATA**

For the definition of this service see IEEE 802.3.

This local service shall be used to transmit (M_UINITDATA.req) or receive (M_UNITDATA.ind) a PDU.

6.3.16.3.2 MAU Type Change

This local service shall be used to indicate a change of certain Local Systems Data. Table 91 shows the parameters of the service.

Table 91 – MAU Type change

Parameter name	ind
Argument	M
Local Port ID	M
Local MAU Type	U
Local Autonegotiation State	U
Local Link State	U

Argument

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

Local Port ID

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

Local MAU Type

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

Local Autonegotiation State

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

Local Link State

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

6.3.16.3.3 Set MAU Type

This local service shall be used to indicate a change of certain Remote Systems Data. Table 92 shows the parameters of the service.

Table 92 – Set MAU Type

Parameter name	ind
Argument	M
Local Port ID	M
Local MAU Type	U
Local Autonegotiation State	U
Local Link State	U

Argument

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

Local Port ID

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

Local MAU Type

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

Local Autonegotiation State

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

Local Link State

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

6.3.16.4 Invocation of the IEEE 802.3 object

There is one IEEE 802.3 object for each interface.

6.3.17 Common DL mapping ASE**6.3.17.1 Overview**

This ASE provides a common interface for DL mapping.

6.3.17.2 DL Mapping class specification**6.3.17.2.1 General**

The DL Mapping ASE defines one DL Mapping object type:

6.3.17.2.2 Template

A DL Mapping object is described by the following template:

ASE: DL Mapping ASE

CLASS: DL Mapping

CLASS ID: not used

PARENT CLASS: top

ATTRIBUTES:

1 (m) Key Attribute: Implicit

SERVICES:

1 (m) OpsService: P Data

2 (m) OpsService: N Data

3 (m) OpsService: A Data

4 (m) OpsService: C Data

6.3.17.2.3 Attributes**Implicit**

The key attribute Implicit indicates that the DL Mapping object is implicitly addressed by the service.

6.3.17.3 DL Mapping service specification**6.3.17.3.1 P Data**

This service is used to transfer time stamped data from the IO device to the IO controller and vice versa. Table 93 shows the parameters of the service.

Table 93 – P Data

Parameter name	req	ind	cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
T Stamp		M	
DA	M	M(=)	
SA	M	M(=)	
VLAN Prio	M	M(=)	

Parameter name	req	ind	cnf
VLAN ID	M	M(=)	
LT	M	M(=)	
N SDU	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.

CREP

This parameter is the local identifier for the desired CR.

D Port

This parameter contains the desired Port ID.

T Stamp

This parameter contains the time stamp.

DA

This parameter contains the destination address according to IEEE 802.3.

SA

This parameter contains the source address according to IEEE 802.3.

VLAN Prio

This parameter contains the priority according to IEEE 802.1Q.

VLAN ID

This parameter contains the VLAN ID according to IEEE 802.1Q.

LT

This parameter contains the priority according to IEEE 802.3.

N SDU

This parameter contains the APDU.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Status

This parameter contains the reason of the error.

6.3.17.3.2 N Data

This service is used to transfer non RTA and RTC data from the IO device to the IO controller and vice versa. Table 94 shows the parameters of the service.

Table 94 – N Data

Parameter name	req	ind	cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
DA	M	M(=)	
SA	M	M(=)	
VLAN Prio	M	M(=)	
VLAN ID	M	M(=)	
LT	M	M(=)	
N SDU	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.

CREP

This parameter is the local identifier for the desired CR.

D Port

This parameter contains the desired Port ID.

DA

This parameter contains the destination address according to IEEE 802.3.

SA

This parameter contains the source address according to IEEE 802.3.

VLAN Prio

This parameter contains the priority according to IEEE 802.1Q.

VLAN ID

This parameter contains the VLAN ID according to IEEE 802.1Q.

LT

This parameter contains the priority according to IEEE 802.3.

N SDU

This parameter contains the APDU.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Status

This parameter contains the reason of the error.

6.3.17.3.3 A Data

This service is used to transfer RTA data from the IO device to the IO controller and vice versa. Table 95 shows the parameters of the service.

Table 95 – A Data

Parameter name	req	ind	cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
DA	M	M(=)	
SA	M	M(=)	
VLAN Prio	M	M(=)	
VLAN ID	M	M(=)	
LT	M	M(=)	
A SDU	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.

CREP

This parameter is the local identifier for the desired CR.

D Port

This parameter contains the desired Port ID.

DA

This parameter contains the destination address according to IEEE 802.3.

SA

This parameter contains the source address according to IEEE 802.3.

VLAN Prio

This parameter contains the priority according to IEEE 802.1Q.

VLAN ID

This parameter contains the VLAN ID according to IEEE 802.1Q.

LT

This parameter contains the priority according to IEEE 802.3.

A SDU

This parameter contains the APDU.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Status

This parameter contains the reason of the error.

6.3.17.3.4 C Data

This service is used to transfer RTC data from the IO device to the IO controller and vice versa. Table 96 shows the parameters of the service.

Table 96 – C Data

Parameter name	req	ind	cnf
Argument	M	M(=)	
CREP	M	M(=)	
D Port	M	M(=)	
DA	M	M(=)	
SA	M	M(=)	
VLAN Prio	M	M(=)	
VLAN ID	M	M(=)	
LT	M	M(=)	
N SDU	M	M(=)	
APDU Status	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.

CREP

This parameter is the local identifier for the desired CR.

D Port

This parameter contains the desired Port ID.

DA

This parameter contains the destination address according to IEEE 802.3.

SA

This parameter contains the source address according to IEEE 802.3.

VLAN Prio

This parameter contains the priority according to IEEE 802.1Q.

VLAN ID

This parameter contains the VLAN field according to IEEE 802.1Q.

LT

This parameter contains the priority according to IEEE 802.3.

C SDU

This parameter contains the APDU.

APDU Status

This parameter contains the field APDU Status.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

Status

This parameter contains the reason of the error.

7 Communication model for decentralized periphery

7.1 Concepts

7.1.1 User requirements

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 24 V digital signals or 4 mA to 20 mA analogue signals.

This communication model defines a decentralized system and the connection is based on IEEE 802.3 information and telecommunication technology. The defined object model is an enhancement of the IEC 61158-5-3 object model.

7.1.2 Features

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 with high accuracy.
- Parameterization and diagnosis can be done without additional wiring.
- Commissioning of subparts without additional installations.

Table 97 gives an overview of the requirements and features of the system.

Table 97 – Requirements and features

Requirement	Feature
Short reaction time	Cyclic exchange of more than 1 000 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 IEC 61158-5-3 / IEC 61158-6-3	Similar application process object model with compatibility modus
Unified protocols	Use of the same real-time transport protocol mechanism for different application models
Excellent diagnosis	Diagnosis and alarm object model for IO devices

Requirement	Feature
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 μ s, signaling to the application
Synchronization of clocks	High precision time synchronization
System redundancy	IO device and IO controller redundancy, support for media redundancy
Interoperability	Precise and complete definitions including the definition of the system behavior
Modification during operation	Dynamic reconfiguration

7.1.3 Associations

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

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

The 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 6). 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.

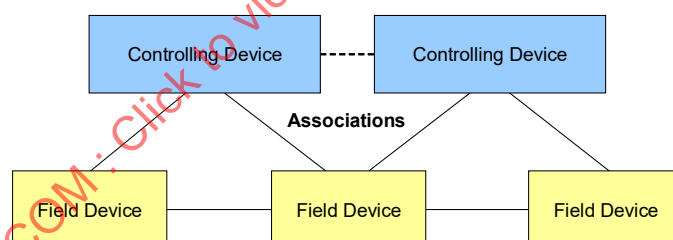


Figure 6 – Example of communication between controlling devices and field devices

Furthermore, the communication between an engineering device and several controlling or field devices is also supported (see Figure 7). 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).

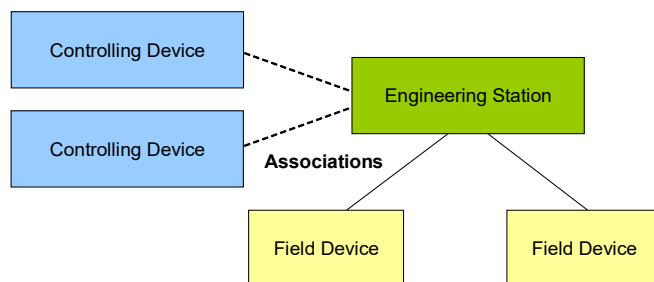


Figure 7 – Example of communication between an engineering station and several controlling and field devices

In addition to these communication models, it also supports the communication between a server station and field devices (see Figure 8). The association between a server station and field device includes transmission of record data via connectionless RPC protocol.

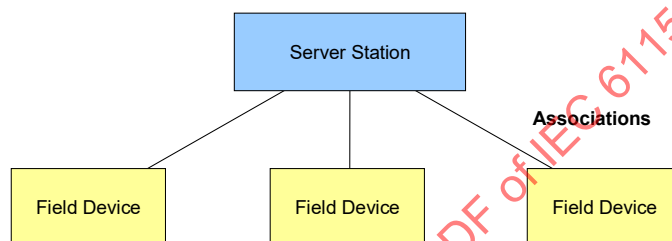


Figure 8 – Example of communication between field devices and a server station

In addition to these communication models, it also supports the communication between field devices (see Figure 9). This communication model includes a controlling device to setup and control the association between the field devices.

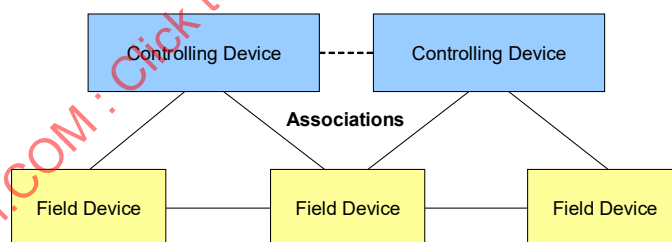


Figure 9 – Example of communication between field devices

The association between controlling devices may include for example the transmission of IO data, diagnosis, alarms and parameter.

7.1.4 Device types

7.1.4.1 General

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

7.1.4.2 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
 - system redundancy
 - dynamic reconfiguration
 - isochronous operation
 - provider/consumer communication between IO devices and IO controllers
 - clock synchronization

7.1.4.3 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.

7.1.4.4 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).

7.1.4.5 IO device

7.1.4.5.1 General

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

7.1.4.5.2 Functionalities

The functionalities are:

- a) cyclic functionalities
 - exchange of IO data with one or 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
 - system redundancy
 - dynamic reconfiguration
 - isochronous operation
 - clock synchronization

IO device conformance classes are defined to support common functions.

7.1.4.5.3 Hierarchical composition

7.1.4.5.3.1 General

In general, an IO device is hierarchically composed using the following elements:

- the IO device itself, which contains
- one or more modules, which contain
- one or more submodules, which contain
- one or more channels

The IO device is composed of different structural units which reflect hardware components or virtual functional units of the field device. These structural units are referred to as modules, submodules, and channels which may reflect physical units, sub-units, and single connections to the process.

The address hierarchy uses the following attributes:

- one or more IO device instances (see Annex A), which contain
 - one or more application processes referenced by its identifier API, which contain
 - one or more slots, which contain
 - one or more subslots, which contain
 - one or more channel numbers

The elements of the IO device are addressed by means of Object UUID (addressing the device instance), API Number, Slot Number, Subslot Number, and Channel Number.

7.1.4.5.3.2 Instance and instance number

The application layer uses instances of IO devices identified via instance number to reflect the structuring of functions or components (e.g. hardware modules, logical units). An instance shall have one or more application processes which represent the further level to structure the data.

Each instance shall be addressed by an instance number, which is part of the RPC object UUID.

7.1.4.5.3.3 Application process

The application layer application processes are identified via identifier API. An application process shall have one or more modules which represent the further level to structure the data.

Each application process shall be addressed by an API.

The default AP (identified over API = 0) shall provide device related information. Other APs are uniquely assigned to device profiles and further standardized use

7.1.4.5.3.4 Modules and slots

The application layer uses modules plugged in slots to reflect the structuring of functions or components (e.g. hardware modules, logical units) inside an IO device. A module shall have one or more submodules which represent the further level to structure the data.

Each module shall be addressed by a slot number 0 to 0x7FFF. The actual use of the slot number is manufacturer specific. Numbering may contain gaps.

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 general station description for configuration purpose.

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

In modular IO devices modules plugged in slots determine the real identification. The number of slots and thus the maximum number of pluggable modules 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 real identifications.

7.1.4.5.3.5 Submodules and subslots

The application layer uses submodules plugged into subslots to reflect the structuring of functions or further components (e.g. hardware units, logical units) inside a module. A submodule shall have one or more channels which represent the further level to structure the data.

Each submodule shall be addressed by a subslot number 0 to 0x9FFF. The actual use of the subslot number is manufacturer specific. Numbering may contain gaps.

If ARProperties.PullModuleAlarmAllowed is not used, subslot number zero is used in conjunction with AlarmType "Pull" to address a certain module within the IO device. In this case, subslot number zero is not usable to plug a submodule.

If ARProperties.PullModuleAlarmAllowed is used, subslot number zero acts as a real subslot.

A submodule provides the following objects:

- Record Data (e.g. parameter)
- Identification and maintenance data
- IO Data (as a summary of the channels)
- Alarms

7.1.4.5.3.6 Channel and channel numbers

The application layer uses channels assigned to a submodule for structuring of functions or further components (e.g. hardware units, logical units) inside a submodule.

These channels are visible in parameterization, IO data and diagnosis.

Figure 10 shows the affiliation of data objects and diagnosis data to the submodule. All diagnosis data is assigned to a channel.

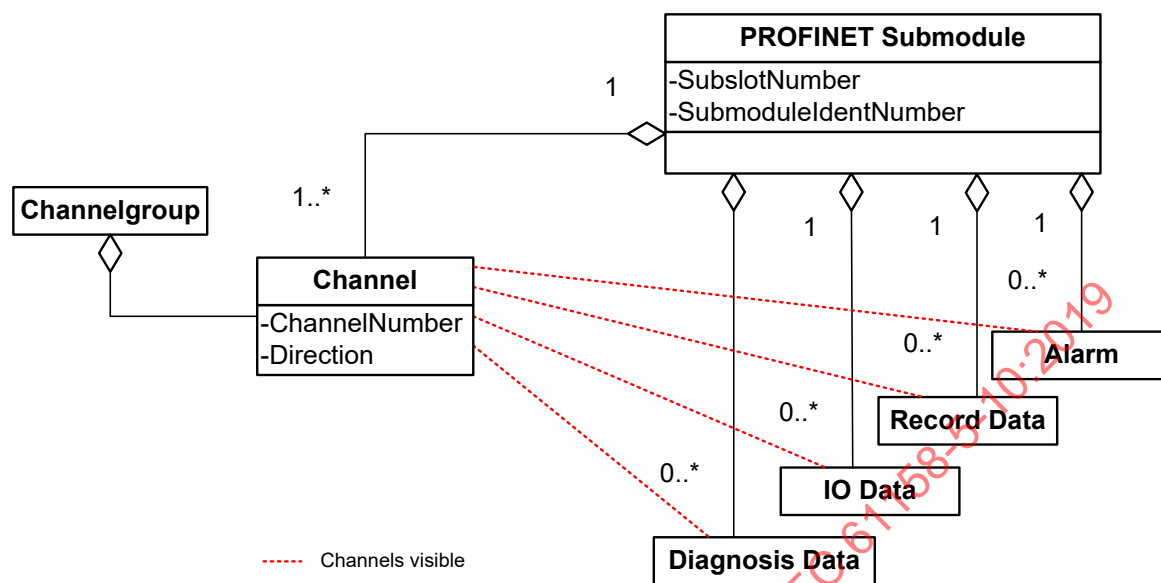


Figure 10 – Data Objects and Diagnosis Data Model

A PROFINET submodule consists of one or more channels.

A particular channel is uniquely identified by its ChannelNumber and Direction.

Each channel shall be addressed by a channel number 0 to 0x7FFF. The manufacturer decides whether the channel numbers are counted independently for in and out direction, or have a common numbering. The actual use of the channel number and direction is manufacturer specific. Numbering may contain gaps.

The special channel number 0x8000 means “the whole submodule”.

Channel group is a vendor specific aggregation of multiple channels for diagnosis purposes.

NOTE A sample for the usage of channel groups: A submodule offers eight digital input channels, numbered 0 to 7 with direction IN. Power is supplied separately for channel 0 to 3 and for channel 4 to 7. A power fail in the first group results in diagnosis related to channel group represented by channel 0, in the second group by channel 4.

For diagnosis purposes, a channel is equivalent to a monitored entity. The monitoring may be realized by the submodule itself (e.g. short circuit, wire breakage or over temperature) or else by the attached sensor or actor (e.g. sensor fault).

Diagnosis related to actor, sensor, or its connection (e.g. wire break) is stated with the channel number and direction as assigned by the vendor.

Diagnosis related to the channel group (e.g. power supply failure of the channel group) is stated with the lowest channel number of the channel group (i.e. the representative).

The special channel with channel number 0x8000 and direction “manufacturer specific” is representing the “whole submodule”. Diagnosis related to the submodule itself (e.g. submodule is defective) is stated with this special channel number.

Submodules without IO data (e.g. interface and port submodules) have one channel only; the special channel “whole submodule”.

Submodules with IO data (e.g. digital input submodules) have more than one channel: one or more channels representing actors/sensors, and in addition the special channel “whole submodule”.

For the relation between IO data and channels, refer to 7.3.1.3.6.1.6.

For the relation between record data and channels, refer to 7.3.1.6.1.

The manufacturer is responsible for modeling the channels of a submodule. Examples are illustrated in Figure 11.

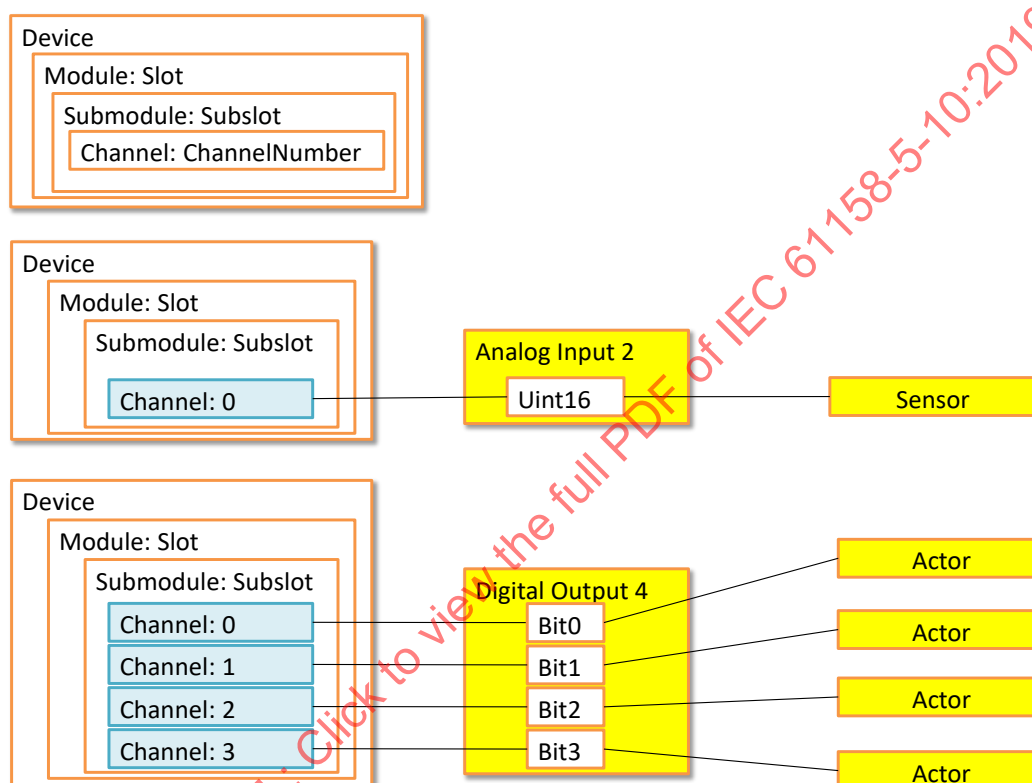


Figure 11 – Example for channel modelling

Figure 12 shows an example of mapping. Both “real peripherals” and “virtual subcomponent” may exist and fit to the PROFINET device model.

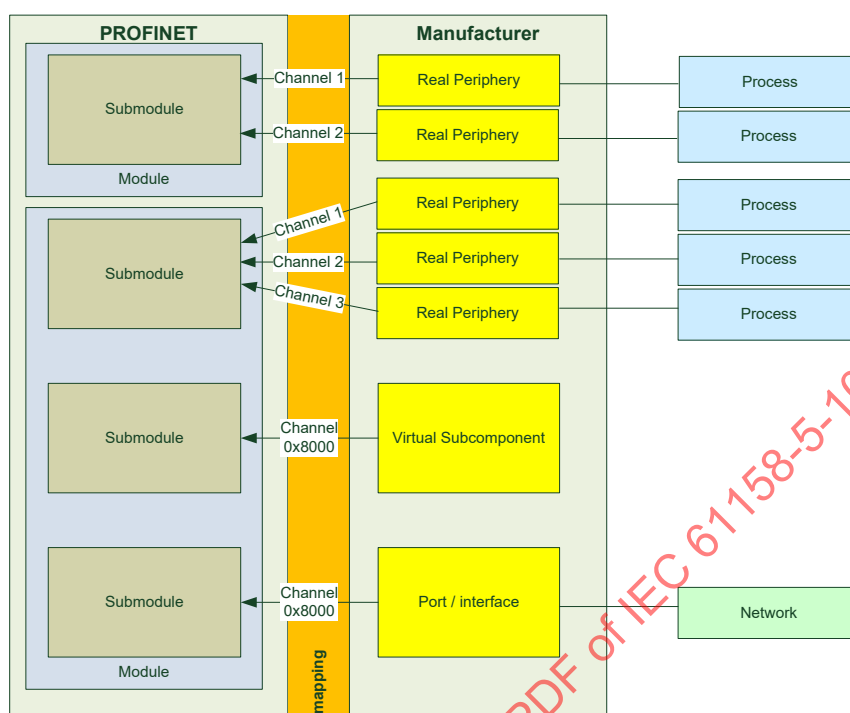


Figure 12 – Mapping to PROFINET device model

7.1.4.5.4 Identification hierarchy

The following identification hierarchy is defined for the application layer elements:

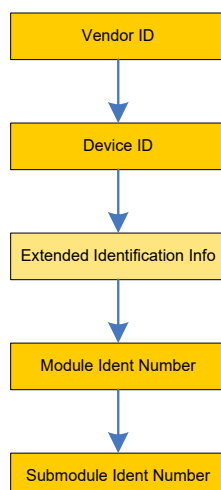


Figure 13 – Identification hierarchy

A particular IO device is identified over Vendor ID and Device ID. The Vendor ID is a globally organizational administered number identifying a particular vendor; the Device ID is uniquely assigned to the device type by the vendor of the device. See also 7.4.1.

The Extended Identification Info allows extending the identification rules. The Extended Identification Info is provided as an AR Vendor Block during the Application Relationship class's service Connect. If no extended identification info is provided, no additional identification rules apply.

The Module Ident Number shall uniquely designate all modules which may appear in a specific device. If a module can be placed in different devices, the ModuleIdentNumber shall be assigned uniquely across all these devices.

The Submodule Ident Number shall uniquely designate all submodules which may appear in a specific module. Beyond identifying the submodule's hardware, it identifies the representation of the submodule's input and/or output data regarding its length and data type, its diagnosis and its parameterization. If a submodule can be placed in different modules, the SubmoduleIdentNumber shall be assigned uniquely across all these modules.

If the same hardware submodule has different representations of input and/or output data, the submodule shall have different Submodule Ident Numbers for each of those representations. Note that each of these Submodule Ident Numbers shall still be able to identify the specific submodule, so each of these ident numbers shall be different from the ident numbers of other submodule hardware on the same module.

So, while the Vendor ID, Device ID and Module Ident Number shall be one-to-one, i.e. there is exactly one ident number for a particular hardware type, a submodule may have several unique SubmoduleIdentNumbers.

The identification numbers are stored in the hardware and can be read out, e.g. via the RPC Object-UUID (Vendor ID and Device ID), via I&M functions (Vendor ID and Order ID [as a synonym for Device ID]), via AR Data (Extended Identification Info) and via Real Identification Data (Module Ident Number and Submodule Ident Number).

7.1.4.5.5 Combined Object

An IO device may define so called "combined objects". A particular combined object is a group of related submodules, e.g. an interface and its ports. One of the submodules is defined as the combined object's "access point", e.g. the interface. All members of a combined object are known by the IO device.

The transport of the parameters for the submodules of a combined object is done by writing the combined object container to its access point; addressed by API, slot, subslot and index. The combined object container holds all records for all member submodules; for this it is sub structured in subblocks; each having an address for the submodule (API, slot and subslot), an index for the data record to write and the data itself, which are written during startup or plug sequence. The combined object container shall only contain data records assigned to submodules of the combined object. The records shall be sorted according to the transfer sequence defined in 7.3.1.2.6.3.1.

One or more combined object containers are transmitted to the access point of a particular group. The number of containers is defined by the amount of record data to be transmitted and the allowed record size for a single record.

A combined object container could be integrated into MultipleWrite.

Immediate error response (i.e. as a result of the Record Data write service) is given only if the combined object container contains records which are not part of the combined object or when the access point does not exist.

For errors due to parameter checking and applying 7.3.1.6.6.1 applies.

The access point acts as proxy to a local entity holding the Record Data objects located in the combined object container for all submodules of the combined object group. This entity distributes the Record Data objects dedicated for a particular submodule of the combined object, whenever parameterization is required for it (in the scenarios Connect, Prm Begin, Plug or Release).

This procedure allows plugging a bunch of submodules at the same time (e.g. a pluggable interface with its ports) and having an implicit parameterization bracket for all submodules together (this is especially important in the context of the physical device). This however works only if all record data for the combined object is placed into the combined object container.

A port or interface parameter record change, a change of a parameter of a COC group member, shall be executed by a PrmBegin/End sequence for the corresponding COC group access point.

Record data is always read directly from the particular submodule; in particular the “transport box” combined object container is not readable.

A special case for the combined object is used for the physical device (e.g. in the context of pluggable interfaces and ports). The CO access point in this case is the interface submodule. The combined object container contains the Record Data objects for all interface and port submodules.

If ARProperties.CombinedObjectContainer is set to COC, the IO controller shall send all parameters during the parameterization sequence using one or more combined object containers to the group. The IO device may deny the write of single data records during this sequence.

7.1.5 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.

7.1.6 Application process

7.1.6.1 Application process objects

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

In order to permit an AP to communicate with an AP of another device, APOs have to be available. Application process objects (virtual objects) represent existing process objects (real objects), that an application process has made visible and accessible to the communication (see Figure 14).

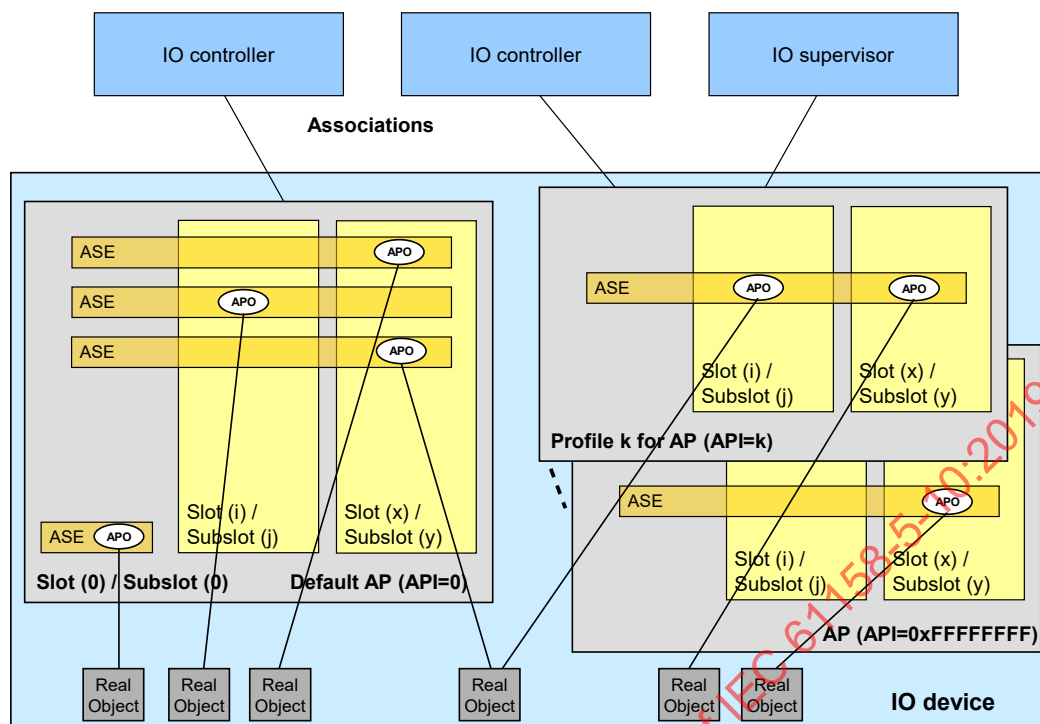


Figure 14 – Application Process with application process objects (APOs)

In Figure 15, 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.

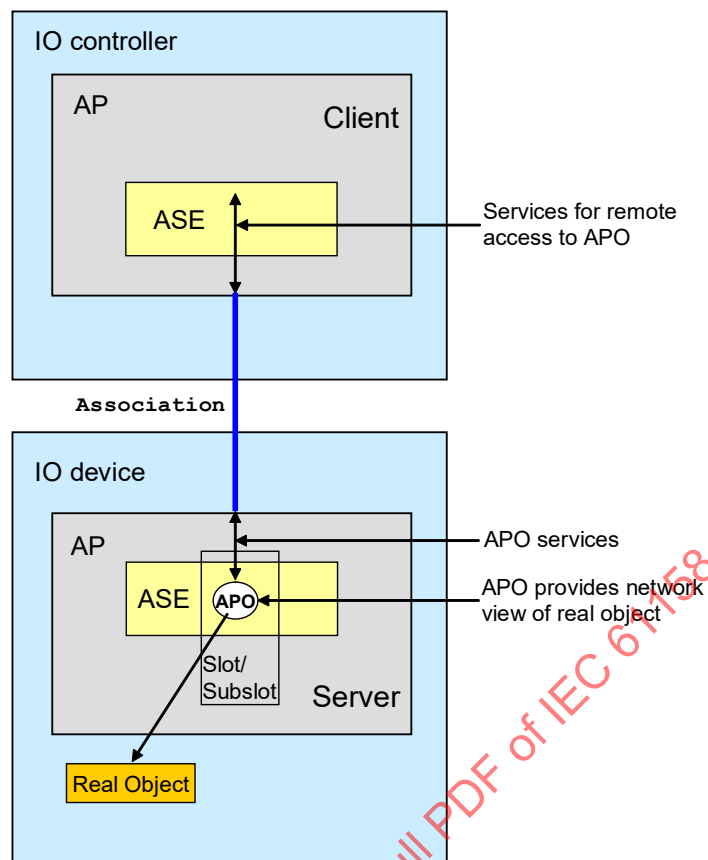


Figure 15 – Access to a remote APO

In Figure 16, 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 Communication Relationship class of AR 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.

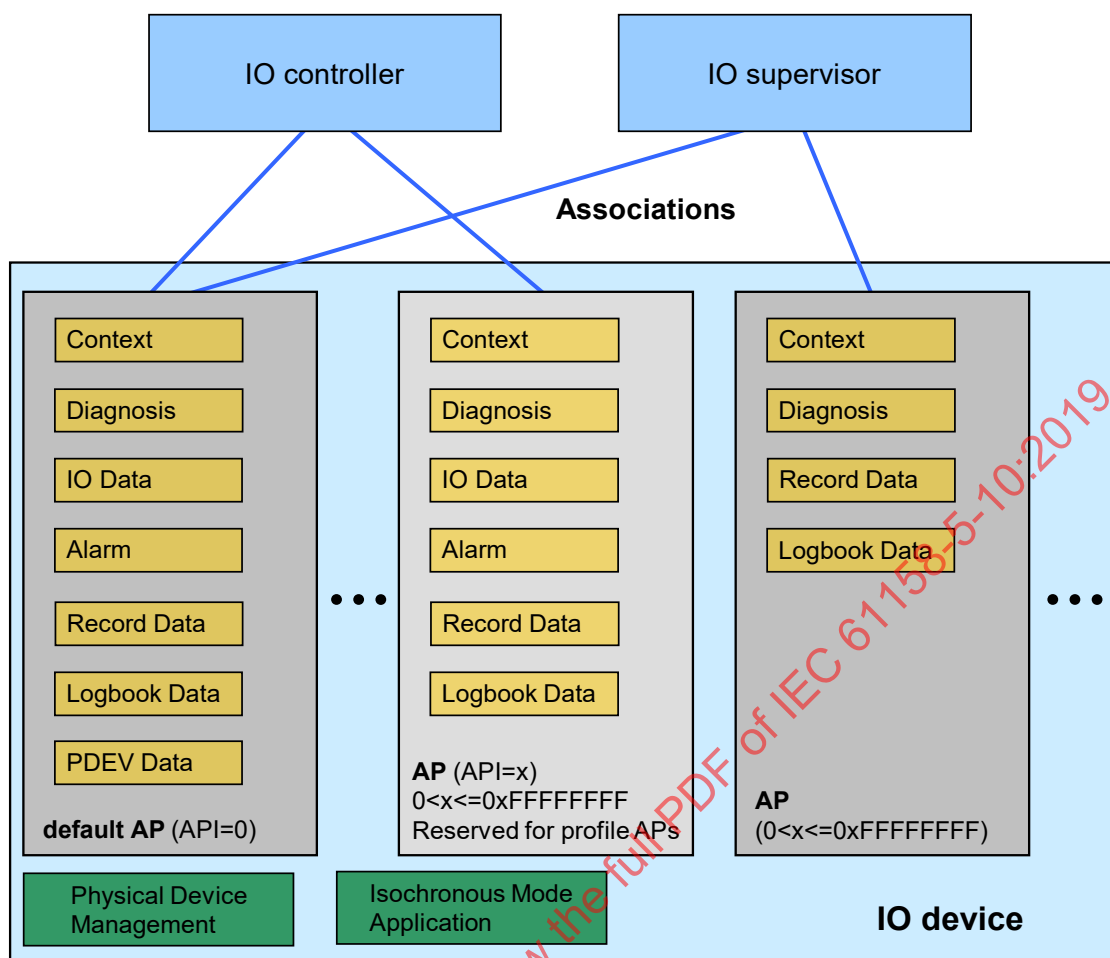


Figure 17 – Overview of application processes

An AP may be distributed to several slots and subslots. Figure 18 shows the relationship between the APs, the data elements, the slots, and subslots of an IO device.

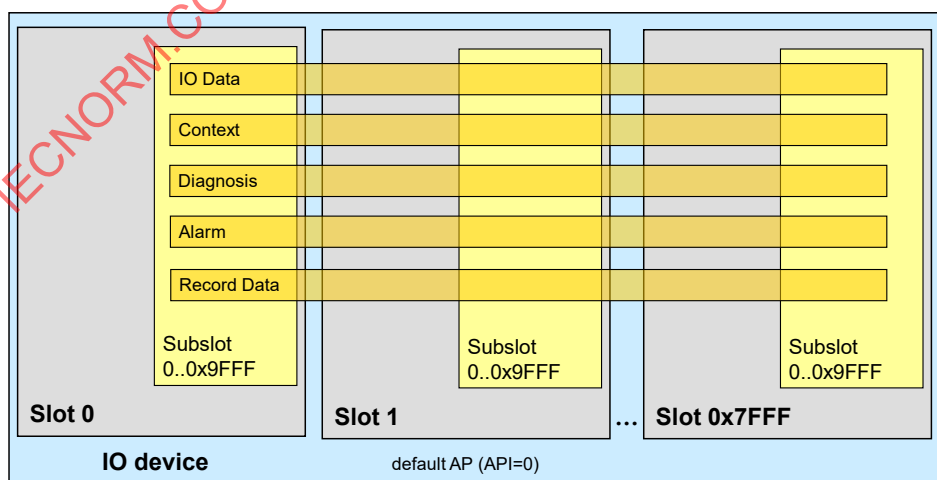


Figure 18 – IO device with APs, slots and subslots

NOTE 1 The mapping of a real object to the 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 Real Identification ASE shall always be identical regardless of the addressed AP
- API 0 addresses the application process
- API 1 to 0xFFFFFFFF is reserved for IO profiles which shall be unique.

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

- The slots 0 to 0x7FFF are freely 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.
- The slots 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 0x0000 is freely usable within the scope of the addressed AP, if ARProperties.PullModuleAlarmAllowed is used.
Subslot 0x0000 is used to address the module within the scope of the Pull Alarm to signal pulling of a module, if ARProperties.PullModuleAlarmAllowed is not used.
- The subslots 0x0001 to 0x7FFF are freely 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.
- The subslots 0x8000 to 0x8FFF shall be used to address interface and port submodules. If used by system redundancy, the interface is mounted left or up into the rack.
They shall only be used in the default API (zero).
- The subslots 0x9000 to 0x9FFF shall be used by system redundancy (R1/R2) only for interface and port submodules within a redundant interface mounted right or down into the rack.
They shall only be used in the default API (zero).
- The subslots 0xA000 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 Index:

- The indices 0 to 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 2 Index 0-0xFF can be used for migration of IEC 61784-1 CP3/1 conform devices to access the same Process Data Objects with same index.

- The indices 0x8000 to 0xAFFF are reserved for IEC 61158-6-10 use. The scope of the Record Data object is a particular subslot within the scope of the addressed AP.
- The indices 0xB000 to 0xBFFF are reserved for profile use. The scope of the Record Data object is a particular subslot within the scope of the addressed AP.
- The indices 0xC000 to 0xCFFF are reserved for IEC 61158-6-10 use. The scope of the Record Data object is a particular slot within the scope of the addressed AP.

- The indices 0xD000 to 0xDFFF are reserved for profile use. The scope of the Record Data object is a particular slot within the scope of the addressed AP.
- The indices 0xE000 to 0xEBFF are reserved for IEC 61158-6-10 use. The scope of the Record Data object is a particular application relation.
- The indices 0xEC00 to 0xEFFF are reserved for profile use. The scope of the Record Data object is a particular application relation.
- The indices 0xF000 to 0xF3FF are reserved for IEC 61158-6-10 use. The scope of the Record Data object is a particular AP.
- The indices 0xF400 to 0xF7FF are reserved for profile use. The scope of the Record Data object is a particular AP.
- The indices 0xF800 to 0xFBFF are reserved for IEC 61158-6-10 use. The scope of the Record Data object is the device.
- The indices 0xFC00 to 0xFFFF are reserved for profile use. The scope of the Record Data object is the device.

7.1.6.3 Additional rules for interfaces and ports

An IO device shall contain at least one interface with ports.

Therefore, at least one module (e.g. plugged in Slot Number 0), shall contain up to 16 further special submodules plugged to subslots from 0x8000 (Interface 1), 0x8100 (Interface 2), to 0x8F00 (Interface 16) referred to as Interface submodules.

Furthermore, each interface submodule shall have at least one and up to 255 assigned port submodules, representing one of the interface's ports.

These port submodules are plugged to subslots from 0x8i01 (Port 1) to 0x8iFF (Port 255) referred to as Port submodules.

Figure 19 depicts an example with several ports on the same slot belonging to one interface. The values “port-001” to “port-255” shall be used for naming the port submodules if all port submodules belonging to this interface are located in the same slot as the interface submodule.

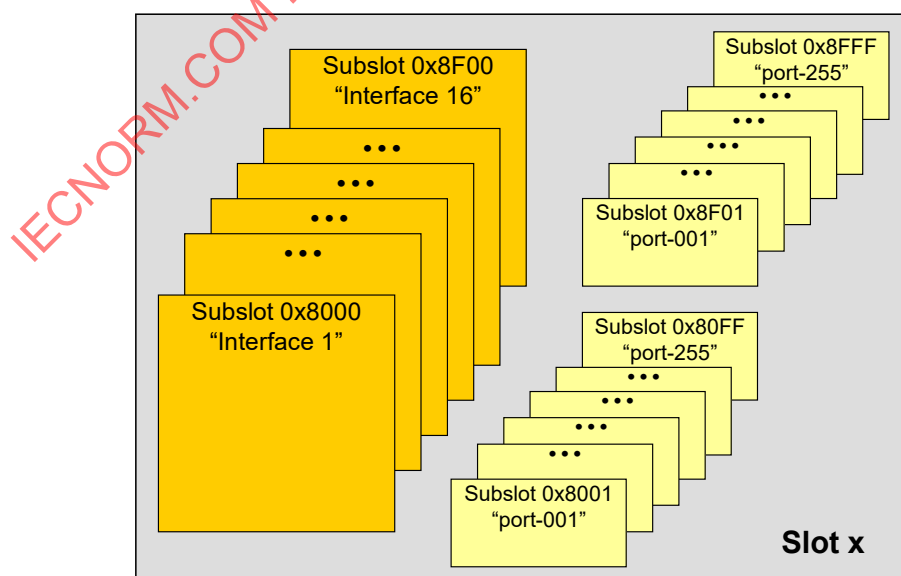


Figure 19 – Example 1 structural units for interfaces and ports within API 0

Figure 20 depicts an example where several port submodules belonging to an interface are plugged to several slots. The values “port-001-xxxx” to “port-255-zzzz” shall then be used for naming the port submodules, whereas “xxxx” denoted the slot number with leading “0” (e.g. port-002-00005”).

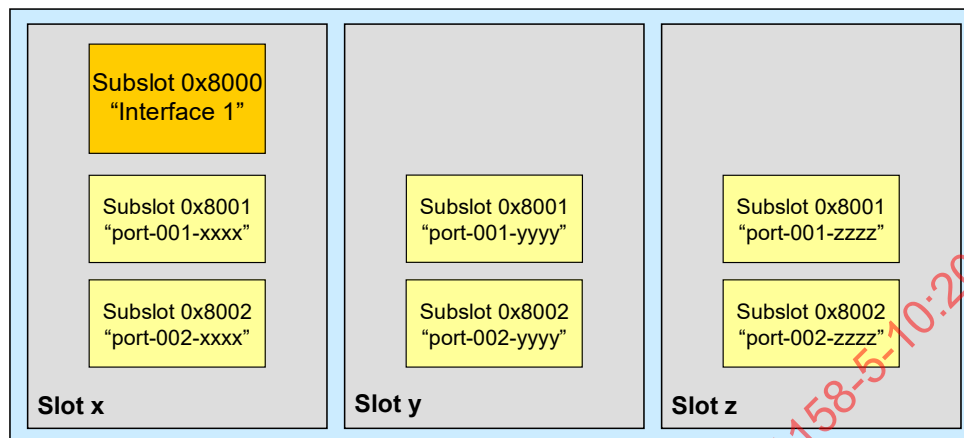


Figure 20 – Example 2 structural units for interfaces and ports within API 0

7.1.6.4 Additional rules for system redundancy

In the special case system redundancy with redundant interface modules the subslot numbering scheme for interfaces and ports is extended as follows:

- Subslot number 0x8000 to 0x8FFF indicates an interface or port submodule in the interface mounted left or up into the rack.
- Subslot number 0x9000 to 0x9FFF indicates an interface or port submodule in the interface mounted right or down into the rack.

For the subslot number range 0x9000 to 0x9FFF the same rules apply as for range 0x8000 to 0x8FFF; i.e. 0x9i00 for an interface submodule with “i” in the range from 0 to 15, denominating the number of the interface; 0x9ipp for a port submodule with “pp” in the range from 1 to 255, denominating the number of the port.

7.1.6.5 Additional rules for RT class 3

In the special case RT class 3 only a structure of the device shall be used, where interface and ports are located on one slot as shown in Figure 19.

7.1.7 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.

7.1.7.1 FAL ASEs

Figure 21 shows the decentralized periphery FAL ASEs according to their OSI layer, including the common services FAL ASEs.

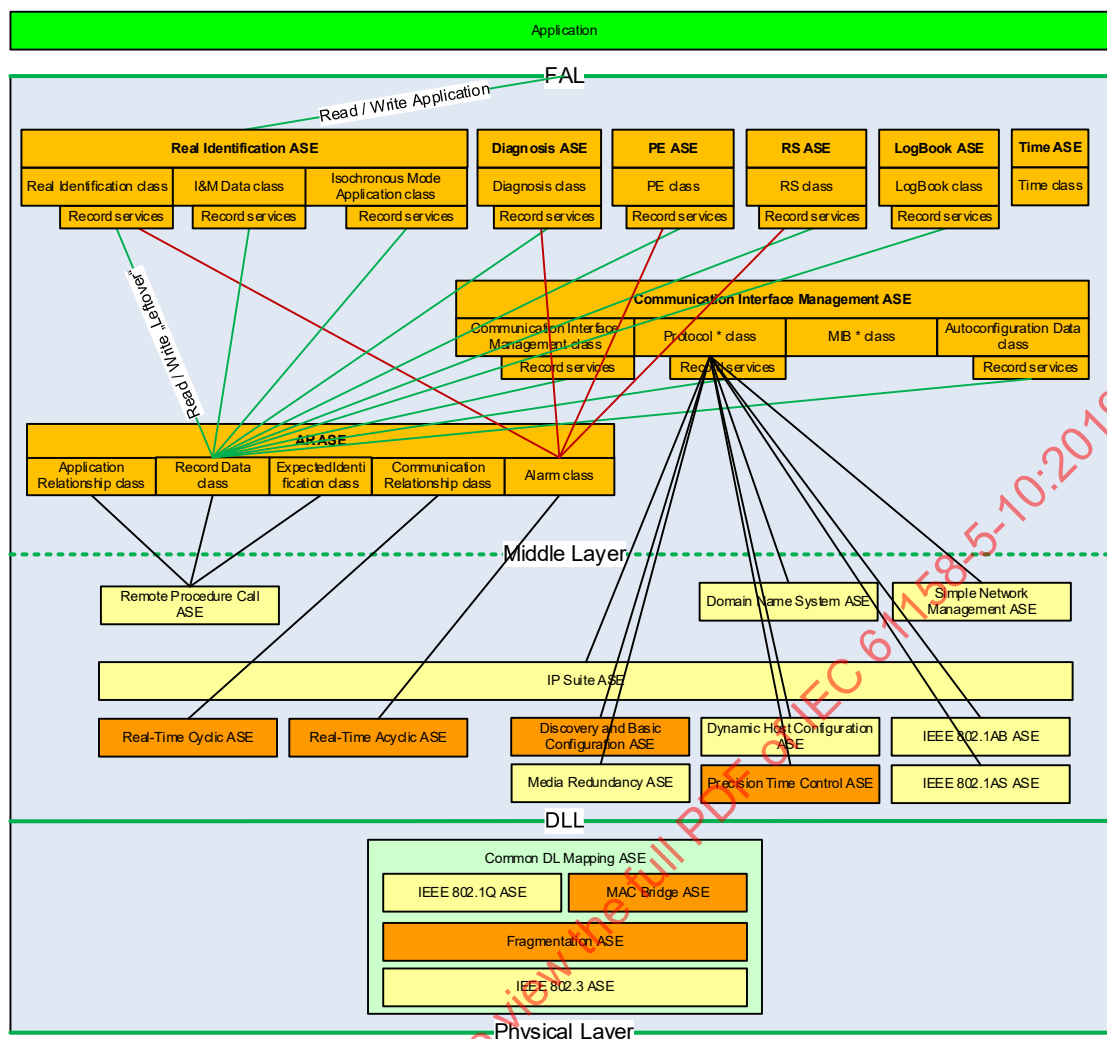


Figure 21 – FAL ASEs communication architecture

Structure

The architecture has 3 layers. The upper layer is the FAL with the shown ASEs and their classes. Furthermore there are the related information's bases.

The middle layer shows all protocols that are specified elsewhere and are used here. These are described in Clause 6. Data of these protocols (DCP, IP, DHCP, DNS, IEEE 802.1AB, IEEE 802.1AS, PTP, MRP, DNS, SNMP) is mapped into the upper layer via Communication Interface Management ASE. Data of the protocols (CL RPC, RTC, RTA) from the middle layer is mapped to the AR ASE. This mapping is shown in Figure 21 by black lines between the ASEs and classes. In the middle and bottom layer, protocols specific for this communication system are marked in orange, others are marked in yellow. The lower layer is the DLL as access point for sending and receiving frames.

There are two places where IO data exist: In the Communication Relationship class the IO data are stored in the Expected Identification Format. At the APO (in the Real Identification ASE), the IO data are stored in the Real Identification Format. The Read Input/Output services access the IO data via the APO.

The Diagnosis ASE, PE ASE and RS ASE are dependent from the RI ASE, entries are only available in relation with submodule entries in RI ASE. For example a plug of submodule creates an entry; a pull of submodule removes an entry.

All profiles and applications only use the services of the FAL interface. This prevents that every profile redefines the general behavior.

7.1.7.2 ASEs in application layer

The application layer offers the following ASEs:

Application Relationship ASE

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

Real Identification ASE

The Real Identification ASE (RI ASE) provides a description model for the current device configuration (real identification data), i. e. plugged modules / submodules, their properties and status.

Communication Interface Management ASE

The Communication Interface 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 device types within the automation device. Anyway, it will be only one instance for most devices.

Diagnosis ASE

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

PE ASE

The PE ASE is a data base located at the device used for the energy saving functionality.

LogBook ASE

The LogBook 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 ASE can be related to all APs of an IO device.

Isochronous Mode Application ASE

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

7.1.8 Application relationship

7.1.8.1 General

An Application Relationship (AR) is a co-operative relationship between two or more APs for the purpose of exchange of information and coordination of their joint operation (see Figure 22). This relationship is activated by the exchange of Application Protocol Data Units (APDU). The application layer uses different types of ARs which distinguish in their conveyance characteristic.

7.1.8.2 Application relationship endpoint

An AP has access to the communication using application relationship endpoints (AREP, see ISO/IEC 7498-1). 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 22 shows an example of one AR with two AREPs.

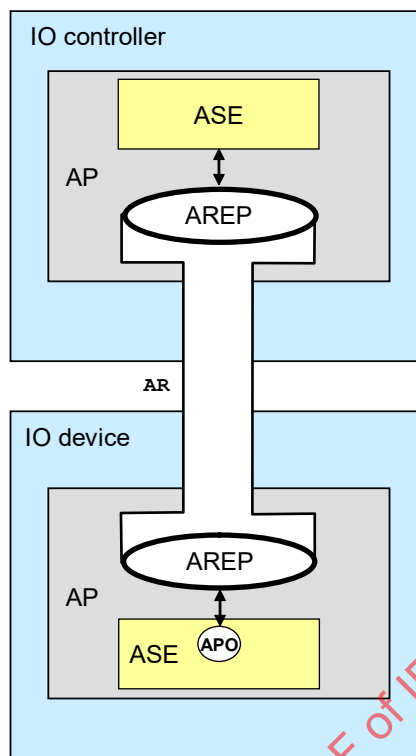


Figure 22 – Example of one AR with two AREPs

7.2 ASE data types

The data types defined in Clause 5 are supported.

7.3 ASEs

7.3.1 AR ASE

7.3.1.1 Overview

The AR ASE is the remote access point. It is the end point for the protocols CL RPC, RTC and RTA. It handles the communication relationships to the remote partners.

The AR ASE is split into different classes:

- Application Relationship class: managing the application relations.
- Expected Identification class: managing the requested configuration of the ARs, managing the Module Diff Block in comparison to the Real Identification.
- Record Data class: The Record Data class is a central distributor for record data sets. Some known record data sets are mapped to their corresponding information base (class), others are assigned to the APO via Real Identification ASE (green lines in Figure 21).
- Communication Relationship class: managing the communication relations and the transportation of IO data with IOPS, IOCS, APDU Status.
- Alarm class: The Alarm class is a central distributor for alarms. It manages the alarms and the alarm queues. Alarms are assigned to the corresponding information base or the APO (via Real Identification ASE) according to their Alarm type (red lines in Figure 21).

The AR ASE is one of the two locations for IO data (the other one is the Real Identification ASE). In the Communication Relationship class the IO data are stored in the Expected Identification Format. IO data are conditioned for the APO and can differ from the Real Identification Format. Values of IO data can be substituted or reduced here by the application.

The AR ASE provides a set of services to establish application relationships, parameterize and configure the APO in the IO device and the submodules containing IO data.

A particular APO of an IO device may need a 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 IO controller's AP. 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. After PowerOn the attributes of the APO of an IO device have manufacturer specific default values. If values are assigned by an IO controller, these are valid until PowerOff of the IO device or reassignment by this or another IO controller.

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 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.

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 application layer defines an implicit AR. It is used for read access to APOs without establishing a dedicated AR. Write access is not permitted over the implicit AR. This AR is locally addressed by an implicit AREP.

Each IO device shall provide one implicit AR that allows processing 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 one explicit AR. An IO device may be able to provide more ARs in parallel according to local resources.

7.3.1.2 Application Relationship class

7.3.1.2.1 Overview

7.3.1.2.1.1 General

The application layer offers the following application relationships:

- IO application relationship
- Supervisor application relationship
- Device access application relationship
- Implicit application relationship

7.3.1.2.1.2 IO application relationship

The IO AR reflects the application relationship (see Figure 23) 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 dynamic reconfiguration by means of using primary and backup IO ARs.

The IO AR provides the application relationship for the following purposes:

- cyclic exchange of input data with the IO controller using
 - mandatory at least one BBUU CR to transmit input data with its provider status IOPS from the IO device to the IO controller and/or to transmit consumer status IOCS for output data from the IO device to the IO controller
 - and optionally one BBUU-OM CR to transmit input data with its provider status IOPS from the IO device to several IO controller and/or IO devices.
- cyclic exchange of output data with the IO controller using
 - mandatory at least one BBUU CR to transmit output data with its provider status IOPS from the IO controller to the IO device and/or to transmit consumer status IOCS for input data from the IO controller to the IO device
 - and optionally one BBUU-OM CR to transmit output data with its provider status IOPS from the IO controller to several IO controller and/or IO devices.
- mandatory acyclic data transfer to read and write record data for e.g. parameterization, configuration and diagnosis (QQCB-CO CR)
- mandatory transmission of alarm data from IO device to IO controller by means of Alarm CR (QQCB-CO CR).

IECNORM.COM : Click to view the full PDF of IEC 61158-5-10:2019

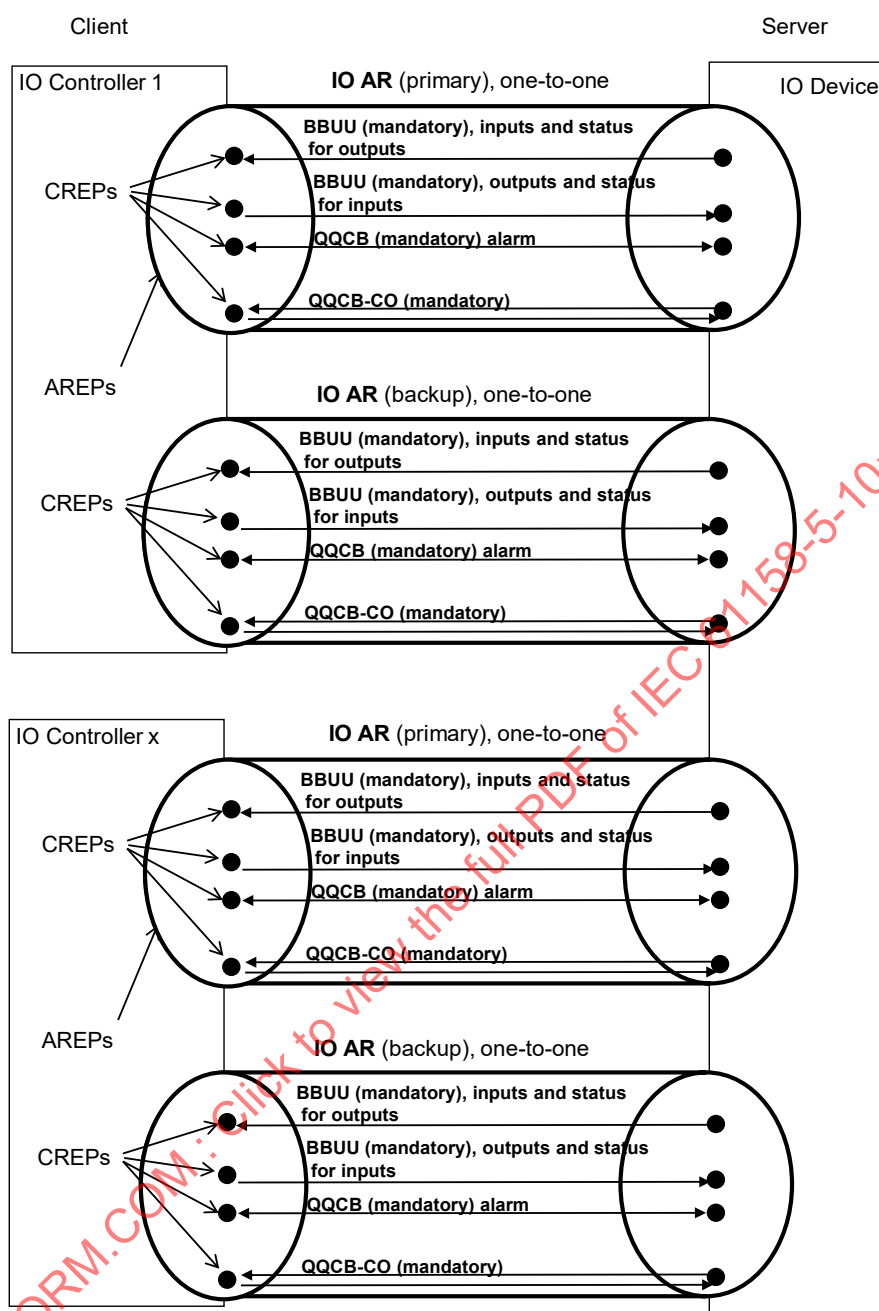


Figure 23 – Example IO application relationship (one-to-one)

Figure 24 illustrates the IO AR with one M CR for one to many uses.

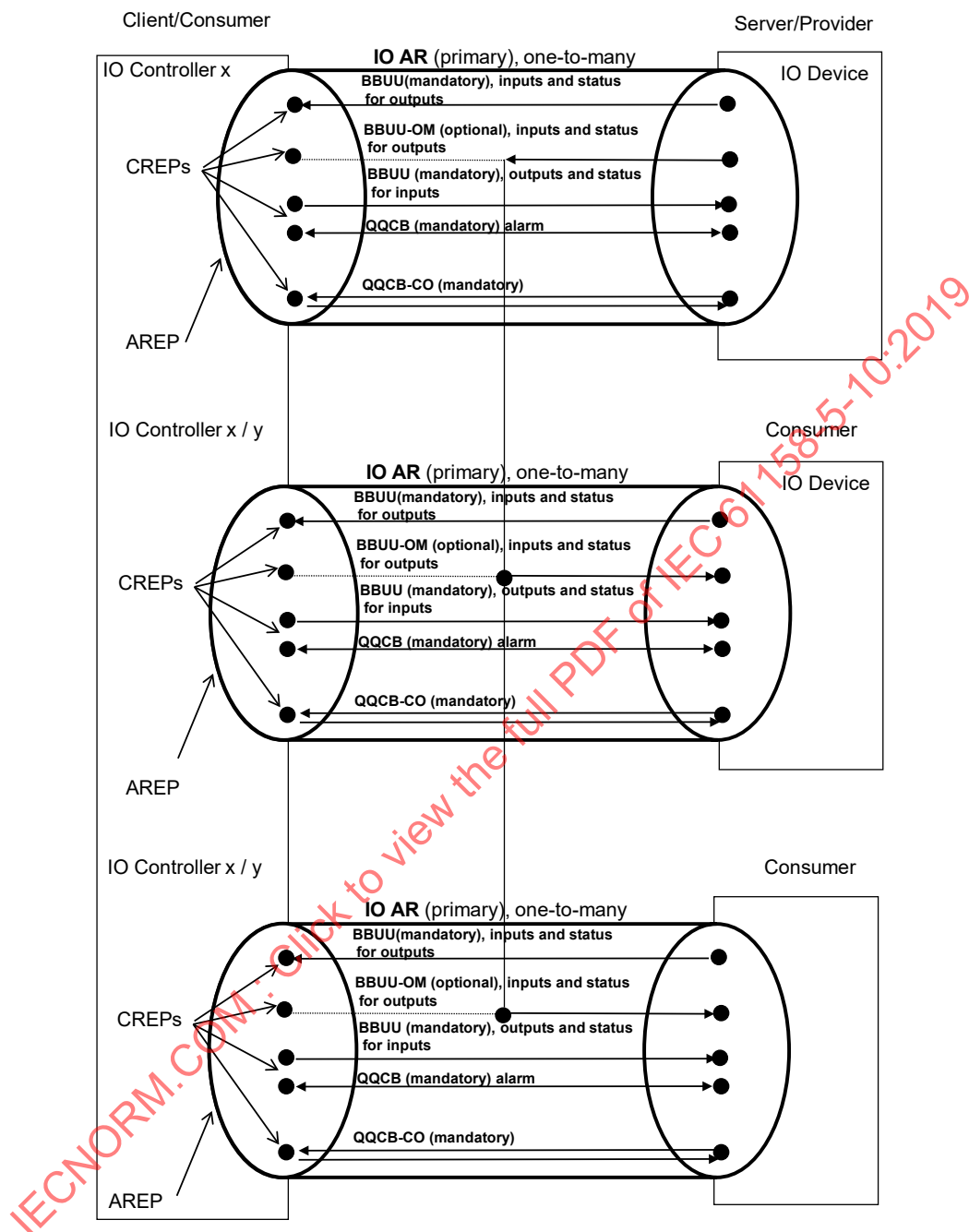


Figure 24 – Example IO application relationship one-to-many

The IO AR has two facets in relation to the startup behavior:

- Normal mode.
- Fast startup mode, to shorten the time for connection establishment.

The IO device shall support the advanced startup mode and may support the legacy startup mode.

7.3.1.2.1.3 Supervisor application relationship

The supervisor AR is a specialized IO AR. This kind of AR allows to takeover a submodule from an IO AR; an IO AR may prevent this supervisor takeover with the attribute Takeover Allowed set to NOT_ALLOWED.

A supervisor AR allows parameterization and commissioning of submodules (including setting and getting IO data cyclically); e.g. for testing of wires. Additionally it allows to take over the submodules from an IO AR; e.g. to set input values manually.

7.3.1.2.1.4 Device access application relationship

The device access AR is an application relationship between the application processes of an IO supervisor and an IO device, for the following purposes:

- acyclic data transfer to read and write record data for e.g. parameterization, configuration and diagnosis (QQCB-CO CR).

A device access AR allows read access to all record data. Furthermore the IO device application (the APO) decides whether write access of a DA AR is allowed, e.g. for parameterization of APOs.

7.3.1.2.1.5 Implicit application relationship

The implicit AR is an application relationship between the application processes of an IO supervisor and an IO device for the following purposes:

- acyclic data transfer to read record data for e.g. diagnosis (QQCB-CL CR).

An implicit AR allows read access to record data of submodules only. Write access is only allowed via explicit established IO AR, supervisor AR, or (if granted) device access AR.

Each IO device shall provide one implicit AR that allows processing at least one outstanding service all the time. It may be able to process more services in parallel according to local resources.

The endpoints AREP and CREP are unique and implicit addressed by local means. The AR UUID of the implicit AR shall be zero.

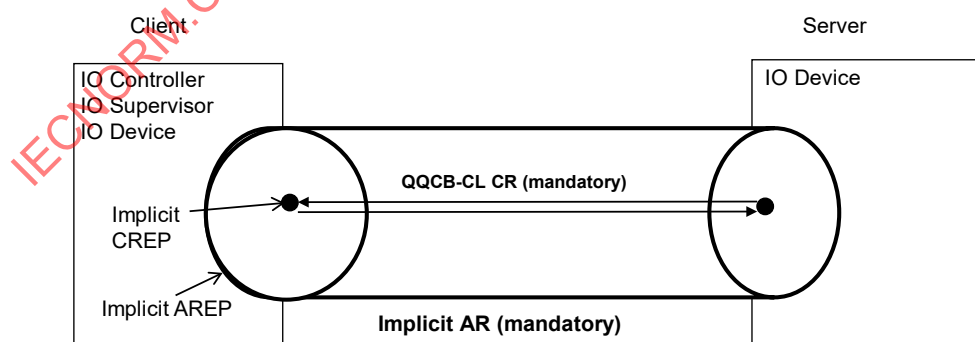


Figure 25 – Implicit application relationship

7.3.1.2.2 Relations

The services are bound to other services as shown in Table 98:

Table 98 – Binding application relationship services

Service	Binding	Usage
Connect (Device Access)	RPC::Connect	used by IO controller or IO supervisor
Release	RPC::Release	used by IO controller or IO supervisor
Prm Begin	RPC::Control	used by IO controller or IO supervisor
Prm End	RPC::Control	used by IO controller or IO supervisor
Application Ready	RPC::Control	used by IO device
Abort	AR:Alarm:Alarm Notification	used by AR class
Local AR Abort	Local	used by FAL
Local Set AR State	Local	used by FAL
Local AR In Data	Local	used by FAL
Read AR Data	RPC::Read	used by IO controller or IO supervisor
Read Module Diff Block for one AR	RPC::Read	used by IO controller or IO supervisor
Read Expected Fast Startup Data	RPC::Read	used by IO controller or IO supervisor
Write Expected Fast Startup Data	RPC::Write	used by IO controller or IO supervisor

7.3.1.2.3 Template

The Application Relationship object is described by the following template:

ASE:	AR ASE
CLASS:	Application Relationship
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
2.8.1	(m) Attribute: Supervisor Takeover Allowed
2.8.2	(m) Attribute: Parameterization Server
2.8.3	(m) Attribute: Device Access
2.8.4	(m) Attribute: Companion AR
2.8.5	(m) Attribute: Acknowledge Companion AR
2.8.6	(m) Attribute: Combined Object Container
2.8.7	(m) Attribute: Startup Mode
2.8.8	(o) Attribute: Pull Module Alarm Allowed
2.9	(m) Attribute: CM Initiator Activity Timeout Factor
2.10	(m) Attribute: Remote Application Ready Timeout
2.11	(m) Attribute: Initiator UDP RT Port
2.12	(m) Attribute: Responder UDP RT Port
2.13	(m) Attribute: CM Initiator Station Name

2.14	(o) Attribute:	Prm Server Block
2.14.1	(m) Attribute:	Parameter Server Object UUID
2.14.2	(m) Attribute:	Parameter Server Properties
2.14.3	(m) Attribute:	CM Initiator Activity Timeout Factor
2.14.4	(m) Attribute:	Parameter Server Station Name
2.15	(m) Attribute:	AR RPC Server Block
2.15.1	(m) Attribute:	Initiator RPC Server Port
2.15.2	(m) Attribute:	Responder RPC Server Port
2.16	(o) Attribute:	IR Info Block
2.16.1	(m) Attribute:	IR Data UUID
2.16.2	(m) Attribute:	Number of IO CRs
2.16.3	(o) Attribute:	List of DFP IO CRs
2.16.3.1	(m) Attribute:	IO CR Reference
2.16.3.2	(m) Attribute:	Subframe Offset
2.16.3.3	(m) Attribute:	Subframe Data
2.16.3.3.1	(m) Attribute:	Subframe Position
2.16.3.3.2	(m) Attribute:	Subframe Data Length
2.17	(o) Attribute:	SR Info Block
2.17.1	(m) Attribute:	Redundancy Data Hold Factor
2.17.2	(m) Attribute:	SR Properties
2.17.2.1	(m) Attribute:	Input Valid on Backup AR
2.17.2.2	(m) Attribute:	Mode
2.18	(o) Attribute:	Redundancy Info
2.18.1	(m) Attribute:	End Point 1
2.18.2	(m) Attribute:	End Point 2
2.19	(o) Attribute:	List of AR Vendor Blocks
2.19.1	(m) Attribute:	AP Structure Identifier
2.19.2	(m) Attribute:	API
2.19.3	(m) Attribute:	Data Request
2.19.4	(m) Attribute:	Data Response
2.20	(o) Attribute:	RS Info Block
2.20.1	(m) Attribute:	RS Alarm Transport Mode
2.21	(o) Attribute:	Fast Startup Data
2.21.1	(o) Attribute:	FS Parameter Block
2.21.1.1	(m) Attribute:	FS Parameter Mode
2.21.1.2	(m) Attribute:	FS Parameter UUID
2.21.2	(o) Attribute:	Manufacturer Specific Fast Startup Data
2.22	(m) Attribute:	AR State
2.23	(m) Attribute:	Sync State

SERVICES:

1	(m) OpsService:	Connect
2	(m) OpsService:	Connect Device Access
3	(m) OpsService:	Release
4	(o) OpsService:	Prm Begin
5	(m) OpsService:	Prm End
6	(m) OpsService:	Application Ready
7	(m) OpsService:	Abort
8	(m) OpsService:	Local AR Abort
9	(m) OpsService:	Local Set AR State
10	(m) OpsService:	Local AR In Data
11	(m) OpsService:	Read AR Data for one API
12	(m) OpsService:	Read AR Data for device
13	(o) OpsService:	Read Expected Fast Startup Data

14

(o) OpsService: Write Expected Fast Startup Data

7.3.1.2.4 Attributes

Implicit

The key attribute Implicit indicates that the AR 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. The attribute is used to uniquely identify a particular AR. It is also used for cross-referencing (e.g. communication relationships).

AR type

This attribute defines the type of the AR entry.

The AR type indicates the intended usage of the IO device.

- The usage by an IO controller in the context of non-system-redundant and non-RT_CLASS_3 is indicated by IOCARSsingle.
- The usage by an IO controller in the context of RT_Class_3 is indicated by IOCARSsingle using RT_CLASS_3.
- The usage by an IO controller in the context of system redundancy or dynamic reconfiguration is indicated by IOCARSR.
- The usage by an IO supervisor is indicated using IOSAR.

Attribute type: Unsigned16

Allowed values: IOCARSsingle, IOCARSsingle using RT_CLASS_3, IOCARSR, IOSAR

AR UUID

This attribute defines the UUID of the AR entry which is provided by the engineering system or the CM Initiator.

Special Case: For an AR entry used in the context of AR type IOCAR_SR, several ARs build an AR set. In this case a part of the UUID indicates the positional number of the AR within the AR-Set and the number of AR resources to allocate at this communication endpoint. A further part of the AR UUID contains a signature over the configuration data records. The last part of the AR UUID contains a signature over the IO device's name of station.

For coding of the AR UUID see IEC 61158-6-10.

Attribute type: UUID

Session Key

This attribute shall be increased by one for each connect service by the CM Initiator and stored within the whole session. 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 attribute contains the value of the IEEE 802 MAC address of the initiator (IO controller or IO supervisor).

Attribute type: OctetString[6]

CM Responder MAC Add

This attribute contains the value of the IEEE 802 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 contains the initial state of the AR.

Allowed values: ACTIVE

Supervisor Takeover Allowed

This attribute determines whether a supervisor takeover is allowed or disallowed for all submodules used by the AR.

Allowed values: ALLOWED, NOT_ALLOWED

Parameterization Server

This attribute contains the source for parameter data.

Allowed values: EXTERNAL_PRM_SERVER, CM_INITIATOR

Device Access

This attribute contains the value for the access policy to record data objects.

Allowed values shall be according to Table 99.

Table 99 – Device Access

Value	Meaning
AR_CONTEXT	Only the submodules from the ExpectedSubmoduleBlock are accessible
DEVICE_CONTEXT	Submodule access over an IOSAR is controlled by IO device application

Companion AR

This attribute contains the value to identify companion ARs.

Allowed values shall be according to Table 100.

Table 100 – Companion AR

Value	Meaning
SINGLE_AR	RT_CLASS_1, RT_CLASS_2, RT_CLASS_3, or RT_CLASS_UDP connection establishment
FIRST_COMPANION_AR	For future use
SECOND_COMPANION_AR	For future use

Acknowledge Companion AR

This attribute contains whether acknowledge for companion ARs is needed.

Allowed values shall be according to Table 101.

Table 101 – Acknowledge Companion AR

Value	Meaning
UNACKNOWLEDGED	RT_CLASS_1, RT_CLASS_2, RT_CLASS_3, or RT_CLASS_UDP connection establishment
ACKNOWLEDGED	For future use.

Combined Object Container

This attribute indicates the usage of the Combined Object Container, dependent from Startup Mode.

For allowed values and their meaning, see IEC 61158-6-10.

Startup Mode

This attribute indicates the startup mode of the application relation.

Allowed values shall be according to Table 102.

Table 102 – Startup Mode

Value	Meaning
Legacy	Recommended Compatibility mode for legacy nodes. RT_CLASS_3: The two provider mode for the RT_CLASS_3 startup is required. Compatibility mode for legacy nodes.
Advanced	Mandatory for this document. RT_CLASS_3: The one provider mode for the RT_CLASS_3 startup is required.

Pull Module Alarm Allowed

This optional attribute contains whether issuing the pull module alarm is allowed for the IO device.

Allowed values shall be according to Table 103.

Table 103 – Pull Module Alarm Allowed

Value	Meaning
NOT_ALLOWED	The AlarmType(=Pull) shall signal pulling of submodule and module. The subslot number zero shall code pulling of module in conjunction with AlarmType(=Pull).
ALLOWED	The AlarmType(=Pull) shall signal pulling of submodule. The AlarmType(=Pull module) shall signal pulling of module. The subslot number 0 to 0x9FFF shall code pulling of submodule in conjunction with AlarmType(=Pull).

The attribute shall not be set to ALLOWED on IO devices not supporting the feature “Pull Module Alarm”. Whether an IO device supports the feature is indicated by GSD.

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 Data Hold Factor.

Attribute type: Unsigned16

Allowed values: 100 to 1 000 using startup mode “Advanced”, 1 to 1 000 using startup mode “Legacy”

Time Base: 100 ms

Remote Application Ready Timeout

This attribute defines the time the IO controller uses to monitor the time between receiving the Prm End confirmation and Application Ready indication.

Attribute type: Unsigned16

Allowed value: 300

Time Base: 1 s

Initiator UDP RT Port

This attribute defines the UDP Port which is used for RT_CLASS_UDP communication.

If RT_CLASS_UDP is not used, the value shall be set to 0x8892, otherwise it is recommended to set the value to 0x8892.

Attribute type: Unsigned16

Allowed value: 0x0000 to 0xFFFF

Responder UDP RT Port

This attribute defines the UDP Port which is used for RT_CLASS_UDP communication.

If RT_CLASS_UDP is not used, the value shall be set to 0x8892, otherwise it is recommended to set the value to 0x8892.

Attribute type: Unsigned16

Allowed value: 0x0000 to 0xFFFF

CM Initiator Station Name

This attribute defines the name of the CM Initiator. The encoding shall be according to IEC 61158-6-10 (encoding of NameOfStationValue).

Attribute type: VisibleString[240]

Prm Server Block

This optional attribute contains the following attributes:

Parameter Server Object UUID

This attribute defines the UUID of the Parameter Server.

Attribute type: UUID

Parameter Server Properties

This attribute defines properties of the Parameter Server.

Attribute type: Unsigned32

Allowed value: 0 (reserved for future use)

CM Initiator Activity Timeout Factor

This attribute defines the CM Initiator Activity Timeout Factor for the connection establishment phase between the IO device and the Parameter Server.

Attribute type: Unsigned16

Allowed value: 1 to 1 000

Time Base: 100 ms

Parameter Server Station Name

This attribute defines the name of the Parameter Server.

The syntax of the attribute CM Initiator Station Name shall be applied.

Attribute type: VisibleString[240]

AR RPC Server Block

This attribute is used to allow direct access to the RPC servers and contains the following attributes:

Initiator RPC Server Port

This attribute defines the UDP Port which is used for CM Initiator RPC server. According IANA it is recommended to use a range from 0xC000 to 0xFFFF.

Attribute type: Unsigned16

Allowed value: 0x0000 to 0x03FF (optional), 0x0400 to 0xBFFF (usable), 0xC000 to 0xFFFF (recommended)

Responder RPC Server Port

This attribute defines the UDP Port which is used for CM Responder RPC server. According IANA it is recommended to use a range from 0xC000 to 0xFFFF.

Attribute type: Unsigned16

Allowed value: 0x0000 to 0x03FF (optional), 0x0400 to 0xBFFF (usable), 0xC000 to 0xFFFF (recommended)

IR Info Block

This optional attribute shall be used by the IO device to indicate whether the application relation uses DFP. The parameter is mandatory when using RT_CLASS_3 with advanced startup mode.

The attribute consists of the following attributes:

IR Data UUID

This attribute contains the unique identifier of the isochronous realtime project which is configured by the engineering system.

Attribute type: UUID

Number of IO CRs

This parameter contains the number of IO CRs related to DFP. The parameter shall be set to "0" if no DFP is used; otherwise to the number of IO CRs related to DFP. DFP is supported for one input CR and one output CR.

Attribute type: Unsigned16

Allowed value: 0, 2

List of DFP IO CRs

This list consists of the following attributes:

IO CR Reference

This parameter contains the reference to the related DFP IO CR.

Attribute type: Unsigned16

Subframe Offset

This parameter contains the Number of octets from the begin of the frame C_SDU to the begin of the own subframe C_SDU.

Attribute type: Unsigned16

Allowed values: 0x0000 to 0x059B

Subframe Data

This parameter consists of the following attributes:

Subframe Position

This attribute defines the position number of the own subframe.

Attribute type: Unsigned8

Allowed values: 0x01 to 0x7F

Subframe Data Length

This attribute defines the length of the own subframe.

Attribute type: Unsigned8

Allowed values: 0x01 to 0xFF

SR Info Block

This optional attribute defines the system redundancy parameters of the desired AR. This attribute is present only if the startup mode of the AR is set to "Advanced".

It consists of the following attributes:

Redundancy Data Hold Factor

This attribute contains the time which shall be used to hold the outputs after loss of the primary application relation while waiting for an application relation to become primary.

Attribute type: Unsigned16

Time base: 1 ms

Allowed value: 0x0001 to 0x00C8 (optional), 0x00C9 to 0xFFFF (mandatory)

SR Properties

This attribute is used to define system redundancy behavior. It consists of the following attributes:

Input Valid on Backup AR

This attribute defines whether the IO controller awaits valid input data also on a backup AR.

Allowed values shall be according to Table 104.

Table 104 – Input Valid on Backup AR

Value	Meaning
NOT_VALID	Using Mode 0, the IO controller shall not evaluate the input data in a backup AR. Using Mode 1, the IO device shall mark the data as invalid using APDU_Status.DataStatus.DataValid == Invalid.
VALID	The device shall deliver valid input data in a backup AR

Mode

This attribute defines whether the IO controller supports the use of APDU_Status.DataStatus.DataValid for marking invalid data.

Allowed values shall be according to Table 105.

Table 105 – Mode

Value	Meaning
0	The IO controller does not support APDU_Status.DataStatus.DataValid == Invalid if input data is request as not valid.
1	The IO device shall use APDU_Status.DataStatus.DataValid == Invalid if input data is request as not valid.

Redundancy Info

This optional attribute indicates which of the two redundant interface modules hosts the primary AR. This attribute is present only if the startup mode of the AR is set to “advanced”. It consists of the following attributes:

End Point 1

This attribute contains the network interface, where the AR is hosted.

Allowed values: NEITHER_LEFT_NOR_ABOVE, LEFT_OR_ABOVE

End Point 2

This attribute contains the network interface, where the AR is hosted.

Allowed values: NEITHER_RIGHT_NOR_BELOW, RIGHT_OR_BELOW

List of AR Vendor Blocks

This optional attribute defines the vendor block info of the desired AR. This attribute is present only if the startup mode of the AR is set to “advanced”. The AR Vendor Block allows to transport additional information related to the AR. The definition allows (in spite of its name) to transport information specified within this specification (e.g. for common profiles), profile specific information and vendor specific information.

The key is the combination of API and APStructureIdentifier. Any possible combination may exist only once in this list.

It consists of the following attributes:

AP Structure Identifier

This attribute defines the structure of the attributes Data Request and Data Response.

Attribute type: Unsigned32

Allowed values shall be according to Table 106 and Table 107

Table 106 – APStructureIdentifier with API := 0

Value (hexadecimal)	Meaning
0 – 0x7FFF	Vendor specific This key defines in combination with the key API the structure of Data in ARVendorBlockReq and ARVendorBlockRes.
0x8000	Extended identification rules
0x8001 – 0xFFFF	Administrative number This key defines in combination with the key API the structure of Data in ARVendorBlockReq and ARVendorBlockRes.

Table 107 – APStructureIdentifier with API != 0

Value (hexadecimal)	Meaning
0 – 0xFFFF	Administrative number for application profiles This key defines in combination with the key API the structure of Data in ARVendorBlockReq and ARVendorBlockRes.

API

This attribute defines to which application process the AR Vendor Block belongs to.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Data Request

This attribute contains the vendor block request data

Attribute type: OctetString[512]

Data Response

This attribute contains the vendor block response data

Attribute type: OctetString[512]

RS Info Block

This optional attribute defines the reporting system parameters of the desired AR.

It consists of the following attributes:

RS Alarm Transport Mode

This attribute describes the way how RS events are transported from the IO device to the IO controller.

Table 108 – RS Alarm Transport Mode

Value	Meaning
0	Reporting system events need to be read by record
1	Reporting system events shall be forwarded to the IO controller using the alarm transport

Fast Startup Data

This optional attribute allows adjusting the fast startup behavior per individual application relation

It consists of the following attributes:

FS Parameter Block

This optional attribute contains the Fast Startup Parameter Data for the AR. It is composed of the following attributes:

FS Parameter Mode

This attribute contains the currently active Parameter Mode of the AR.

Attribute type: Unsigned32

Allowed values: ON, OFF

FS Parameter UUID

This attribute contains a UUID given by an engineering system to indicate unchanged parameter sets.

Attribute type: UUID

Manufacturer Specific Fast Startup Data

This optional attribute shall contain manufacturer specific fast startup data.

Attribute type: OctetString

AR State

This attribute defines the state of the AR.

This attribute influences the Write service behavior of the data record ASE (see 7.3.1.6.6.3 item a)). If the attribute is set to BACKUP the Write service is neglected with Error code 1 set to "backup".

It shall be set to PRIMARY, if it is the primary AR of an AR set or if it is a "standalone".

For the first application relation having ARType IOCARSR and belonging to a new AR Set this attribute shall be set to FIRST up to ApplicationReady. After ApplicationReady the AR state of an AR having ARType IOCARSR is set similar to the state (i.e. to PRIMARY or BACKUP) as intended by the IO controller.

For ARs not having ARType IOCARSR the attribute is set to PRIMARY always.

Attribute type: Unsigned16

Allowed values: FIRST, BACKUP, PRIMARY

Default value: PRIMARY

SYNC State

This attribute is the local synchronization status for the desired AR. It shall only be present if synchronization is used.

Attribute type: Unsigned16

Allowed values: SYNCHRONIZED, SYNC_NOT_AVAILABLE

7.3.1.2.5 Services

7.3.1.2.5.1 Connect

7.3.1.2.5.1.1 General

Two types of connect service are supported:

- The Connect service (see 7.3.1.2.5.1.2) to establish an application relation for the use of an IO controller or an IO supervisor with status and control functions.
- The Connect Device Access service (see 7.3.1.2.5.1.3) to establish an application relation for the use of an IO supervisor without status and control functions.

Implicit AR does not require a Connect service.

7.3.1.2.5.1.2 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 class and the Communication Relationship class. The Connect service returns successfully even if there is a total mismatch of expected and real configuration.

Table 109 shows the parameter of the service.

NOTE 1 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 the server side this information is also handled inside the application layer and does not appear as an explicit service parameter. The same model applies to Read services over the implicit AR.

Local implementation may pass this service parameter to the application process.

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 and Submodule Data Length shall only be checked by the user application.

The resources for an application relation are allocated with the Connect of the application relation. This includes the resources for the application relation itself and the resources of all necessary CRs (IOCR, Alarm-CR, M-CR, DR-CR). The resources are freed when the application relation is terminated.

For the first application relation having ARType IOCARSR and belonging to a new AR Set (as indicated by the ARUUID; for details, see IEC 61158-6-10), the resources of the complete AR Set are allocated; i.e. for all ARs of the complete AR Set. The resources of an application relation belonging to the same AR Set established at some later point are assigned from this allocated memory. The resources of the AR Set are freed when the last application relation of the AR Set is terminated.

In case of a RT_Class_3 AR with Startup Mode Advanced, the IR Info Block is mandatory. In case of a SR AR, the SR Info Block is mandatory.

If the application relation cannot be established due to scarce resources, the Connect is rejected with an “out of AR/ARset resources” error.

If an application relation contains PDev submodules which are already owned by another application relation, the Connect may be rejected with an “PDev already owned” error.

The send clock factors of all application relations established in parallel to a particular IO device have to be the same. Thus the send clock factor of the IO device is determined by the application relation first established. The Connect of a succeeding application relation having a different send clock factor is rejected with an error for the corresponding block / parameter.

Table 109 – Connect

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
AREP	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(=)		

Parameter name	req	ind	rsp	cnf
AR Properties	M	M(=)		
CM Initiator Activity Timeout Factor	M	M(=)		
Initiator UDP RT 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(=)		
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(=)		
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(=)		
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(=)		

Parameter name	req	ind	rsp	cnf
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(=)		
Prm 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(=)		
IO CR Reference	M	M(=)		
Address Resolution Properties	M	M(=)		
MCI Timeout Factor	M	M(=)		
Provider Station Name	M	M(=)		
AR RPC Server Block	U	U(=)		
Initiator RPC Server Port	M	M(=)		
IR Info Block	U	U(=)		
IR Data UUID	M	M(=)		
Number of IO CRs	M	M(=)		
List of Subframe Data	U	U(=)		
IO CR Reference	M	M(=)		
Subframe Offset	M	M(=)		
Subframe Data	M	M(=)		
SR Info Block	U	U(=)		
Redundancy Data Hold Factor	M	M(=)		
SR Properties	M	M(=)		
Input Valid on Backup AR	M	M(=)		
Mode	M	M(=)		
AR Server Block	U	U(=)		
CM Initiator Station Name	M	M(=)		
List of AR Vendor Block Requests	U	U(=)		
AP Structure Identifier	M	M(=)		
API	M	M(=)		
Data Request	M	M(=)		
RS Info Block	U	U(=)		
RS Alarm Transport Mode	M	M(=)		
AR FSU Block	U	U(=)		
FS Parameter Block	U	U(=)		
FS Parameter Mode	M	M(=)		
FS Parameter UUID	M	M(=)		
Manufacturer Specific Fast Startup Data	U	U(=)		

Parameter name	req	ind	rsp	cnf
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(=)
Responder UDP RT Port			M	M(=)
List of IO CR Response Blocks			M	M(=)
IO CR type			M	M(=)
IO CR Reference			M	M(=)
Frame ID			M	M(=)
Alarm CR Response Block			M	M(=)
Local Alarm Reference			M	M(=)
Max Alarm Data Length			M	M(=)
Module Diff Block			C	C(=)
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(=)
AR RPC Server Block			C	C(=)
Responder RPC Server Port			M	M(=)
List of AR Vendor Block Responses			C	C(=)
AP Structure Identifier			M	M(=)
API			M	M(=)
Data Response			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
PNIO Status			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.

AR Parameter Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

List of IO CR Parameter Blocks

The parameter list contains the attributes as in Table 109 (parameter description in 7.3.1.3.4).

Alarm CR Parameter Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.5.4).

List of Expected Submodule Blocks

The parameter list contains the attributes as in Table 109 (parameter description in 7.3.1.4.4).

NOTE 2 The list includes only “existing” submodules but does not include “empty” subslots.

Prm Server Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

List of Multicast CR Block

The list contains the attributes as in Table 109 (parameter description in 7.3.1.3.4).

AR RPC Server Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

IR Info Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

SR Info Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

AR Server Block

This optional attribute shall be present only if the startup mode of the AR is advanced. It allows to check, whether the device responding to the connect service is the expected one by checking its name. It consists of the following attributes:

CM Responder Station Name

This parameter contains the name of the CM responder.

The motivation for this block is the phase between a DCP::Identify(name) and the subsequent connect to the IP address of the device. If in-between the name of the device has been modified e.g. by DCP::Set, the connect would not detect this situation. With this block the IO controller is able to check for name == CM responder station name. If unequal, the IO controller shall then abort the connect.

List of AR Vendor Block Requests

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

RS Info Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

AR FSU Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

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 parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

List of IO CR Response Blocks

The parameter list contains the attributes as in Table 109 (parameter description in 7.3.1.3.4).

Alarm CR Response Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.5.4).

Module Diff Block

The parameter contains the Module Diff Block; see 7.3.1.4.6.2 for details.

AR RPC Server Block

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

List of AR Vendor Block Response

The parameter contains the attributes as in Table 109 (parameter description in 7.3.1.2.4).

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned16

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.

Attribute type: Unsigned16

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

7.3.1.2.5.1.3 Connect Device Access

The Connect service is used to establish a Device Access AR.

Table 110 shows the parameter of the service.

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.

Table 110 – Connect Device Access

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
AREP	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(=)		
Initiator UDP RT Port	M	M(=)		
CM Initiator 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(=)
Responder UDP RT Port			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
PNIO Status			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.

AR Parameter Block

The parameter contains the attributes as in Table 110 (parameter description in 7.3.1.2.4).

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 parameter contains the attributes as in Table 110 (parameter description in 7.3.1.2.4)

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.2.5.2 Release

This service shall be used by an IO controller or IO supervisor to close an IO AR or Supervisor AR.

Table 111 shows the parameter of the service.

Table 111 – 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(=)
PNIO Status			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.

Session Key

The parameter is described in 7.3.1.2.4.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.2.5.3 Prm Begin

This Service allows an acknowledged parameterization in the context of an established AR.

The service Prm Begin is oriented one-to-one and onto an application relation indicated via the parameter AREP. Issuing the service Prm Begin is allowed at any time after the IN-DATA notification for the desired AR (after the Application Ready.rsp). The service Prm Begin allows starting an acknowledged re-parameterization sequence for the submodules indicated via the List of Submodules.

All submodules indicated are parameterized within the parameterization sequence; during the parameterization sequence the IO controller delivers a consistent set of parameters by issuing IODWrite.req or IODWriteMultiple.req services. The IO controller completes parameter delivery by issuing the Prm End service. The absence of parameters for a submodule to be parameterized means that the submodule shall use the default parameters.

The parameterization sequence is finished by the IO device issuing Application Ready to the IO controller, again narrowed onto the submodules contained in the parameterization sequence and containing a Module Diff Block oriented to the submodules indicated in the Prm Begin Request. After this Application Ready the submodules are again usable for the IO controller.

For the IOs of the affected submodules the rules for the “normal” startup parameterization sequence apply: They are set to “Good” by the IO device as soon as the parameterization of the appropriate submodule is finished and the submodule is able to deliver or receive data according to the new parameters.

The list of submodules shall have the same or a subset of those submodules listed, as the expected submodule list within the Connect service for the desired AR:

- Variant a (mandatory): Prm Begin request contains all submodules of the expected identification
- Variant b (mandatory): Prm Begin request contains only one submodule of the expected identification (in case of CoC only the access point)
- Variant c (optional): Prm Begin request contains a subset of the expected identification (more than one, less than all)

If the list of submodules contains submodules which are not contained in the expected submodule list of the AR-Connect, the device rejects the service related to the PDU checking

rules (see IEC 61158-6-10). If the device does not support variant c, the device rejects the service related to the PDU checking rules (see IEC 61158-6-10).

After a device receives a Prm Begin request, an affected submodule shall not issue an alarm until the corresponding Application Ready.rsp; it shall store those alarms and issue them after the corresponding Application Ready.

Already issued Alarms except Plug and Release shall be acknowledged by the IO controller. To get into a consistent state the IO device waits with its Application Ready until all outstanding alarms are acknowledged.

Plug and Release shall be stored by the IO controller. They shall be processed and acknowledged after the Application Ready.

If a device does not support the service Prm Begin, then it shall acknowledge with protocol error and abort the AR.

Table 112 shows the parameter of the service.

Table 112 – Prm Begin

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
AREP	M	M(=)		
Session Key	M	M(=)		
List of Submodules	U	U(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Session Key			M	M(=)
Module Diff Block			C	C(=)
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(=)
PNIO Status			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.

Session Key

The parameter is described in 7.3.1.2.4.

List of Submodules

The parameter list contains the attributes as in Table 112 (parameter description in 7.3.1.2.5.3).

Result(+)

This parameter indicates that the service request succeeded.

Module Diff Block

The parameter contains the Module Diff Block; see 7.3.1.4.6.2 for details.

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.2.5.4 Prm End

This service shall be used by the client (IO controller or IO Parameter server) to inform the IO device that all parameters are written by means of the Write service of the Record Data Class in AR ASE.

Special case: If the attribute AR State of the desired AR is set to BACKUP, all data records are neglected during the parameterization sequence (see 7.3.1.6.6.3 item b)). Thus if the AR state is set to BACKUP when the service Prm End is initiated, the Submodule State.AR Info of all submodules related to the parameterization sequence is set to the submodule's state according to the AR with AR state set to PRIMARY or if not existing set to FIRST). If submodules are added (dynamic reconfiguration), which are not yet contained in the PRIMARY or FIRST AR, the submodule's AR state is set to NO_SUBMODULE. If later on the AR used for dynamic reconfiguration is set to PRIMARY and added submodules within this AR are existent or have a decent substitute, those submodules are reported to the AR with a plug alarm and will be parameterized within the plug sequence.

Table 113 shows the parameter of the service.

Table 113 – Prm End

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(=)
Session Key			M	M(=)
Alarm Sequence Number			C	C(=)
Result(-)			S	S(=)
AREP			M	M(=)
PNIO Status			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.

Session Key

The parameter is described in 7.3.1.2.4.

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 (see Sequence Number as defined in 7.3.1.5.4).

Result(+)

This parameter indicates that the service request succeeded. It has the same parameter as the request.

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.2.5.5 Application Ready

By this service the user of an IO device indicates to the IO controller that the parameterization is applied to the APOs.

This service is used in the context of

- Parameterization driven by application relation establishment (Connect, Prm End Sequence),
- Parameterization driven by a plug/released of a submodule (Plug, Prm End) (Released, Prm End), and
- Parameterization driven by a Prm Begin / Prm End sequence.

Table 114 shows the parameters of the service.

Table 114 – 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(=)		
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			C	C(=)
Result(-)			S	S(=)
AREP			M	M(=)
PNIO Status			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.

Session Key

The parameter is described in 7.3.1.2.4.

Alarm Sequence Number

This parameter shall only be used in conjunction with parameterization driven by a plug/release of a submodule. The value shall be taken from the Plug/Released Alarm Notification (see Sequence Number as defined in 7.3.1.5.4).

Module Diff Block

The parameter contains the Module Diff Block; see 7.3.1.4.6.2 for details.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.2.5.6 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 no longer possible. Table 115 shows the parameter of the service.

Table 115 – Abort

Parameter name	req
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.

7.3.1.2.5.7 Local AR Abort

This service shall be used by an IO device or an IO or Supervisor AR to indicate a local abort of an existing AR. Table 116 shows the parameter of the service.

Table 116 – Local AR Abort

Parameter name	req	ind	cnf
Argument	M(=)	M(=)	M(=)
AREP	M(=)	M(=)	M(=)
Session Key	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.

Session Key

The parameter is described in 7.3.1.2.4.

7.3.1.2.5.8 Local Set AR State

This service shall be used to set the state of the addressed AR. Table 117 shows the parameter of the service.

Table 117 – Local Set AR State

Parameter name	req	cnf
Argument	M	M(=)
AREP	M	M(=)
AR State	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 State

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

7.3.1.2.5.9 Local AR In Data

This service indicated that the AR is in state "IN DATA". The next New Data_ind () after the "Local AR In Data.ind" contains the first valid data from the IO controller.

Table 118 shows the parameter of the service.

Table 118 – Local AR In Data

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.

7.3.1.2.5.10 Read AR Data

7.3.1.2.5.10.1 General

All services to read the AR Data object have the same parameter layout. To ease the description the parameter is thus described once upfront to the services. The services themselves act as a filter for the information from the AR ASE.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy of the record data object 7.3.1.6.6.1 applies.

The parameter Data has a service specific structure and meaning. Table 119 shows the specific elements of Data for this service. Note that dependent on the particular service described below only parts of the information are returned.

Table 119 – Data elements of Read AR Data

Parameter name	service specific elements	rsp	cnf
Data	List of ARs	M	M(=)
	AR UUID	M	M(=)
	CM Initiator Object UUID	M	M(=)
	Parameter Server Object UUID	M	M(=)
	AR Properties	M	M(=)
	AR type	M	M(=)
	Local Alarm Reference	M	M(=)
	Remote Alarm Reference	M	M(=)
	Initiator UDP RT Port	M	M(=)
	Responder UDP RT Port	M	M(=)
	CM Initiator Station Name Length	M	M(=)
	CM Initiator Station Name	M	M(=)
	Parameter Server Station Name Length	M	M(=)
	Parameter Server Station Name	C	C(=)
	Number of IOCRs	M	M(=)
	List of IOCRs	M	M(=)
	IO CR Properties	M	M(=)
	IO CR type	M	M(=)
	Frame ID	M	M(=)
	APDU Status	M	M(=)
	Number of APIs	M	M(=)
	List of APIs	M	M(=)
	API	M	M(=)
	IR Info Block	C	C(=)
	IR Data UUID	M	M(=)
	Number of IO CRs	M	M(=)
	List of Subframe Data	C	C(=)

Parameter name	service specific elements	rsp	cnf
	Frame ID	M	M(=)
	Subframe Offset	M	M(=)
	Subframe Data	M	M(=)
	SR Info Block	C	C(=)
	Redundancy Data Hold Factor	M	M(=)
	SR Properties	M	M(=)
	Input Valid on Backup AR	M	M(=)
	Mode	M	M(=)
	List of AR Vendor Block Requests	C	C(=)
	AP Structure Identifier	M	M(=)
	API	M	M(=)
	Data Request	M	M(=)
	List of AR Vendor Block Responses	C	C(=)
	AP Structure Identifier	M	M(=)
	API	M	M(=)
	Data Response	M	M(=)
	AR FSU Block	C	C(=)
	FS Parameter Block	C	CU(=)
	Mode	M	M(=)
	FS Parameter UUID	M	M(=)
	Fast Startup Block	C	C(=)
	Data	M	M(=)
	SRL Data	C	C(=)
	Redundancy Info	M	M(=)
	End Point 1	M	M(=)
	End Point 2	M	M(=)

Data

This parameter contains the values of the object which has been read.

List of ARs

The parameter contains the attributes as in Table 119.

7.3.1.2.5.10.2 Read AR Data for one API

This service allows reading AR Data using a record data object with API scope (see 7.3.1.6.1).

The filter API filters the ARs contained in the response, i.e. having at least one submodule belonging to this API in their expected identification. However, the response contains the unmodified ARData, e.g. the List of APIs contains all APIs requested by this application relationship. If there are only device access ARs or no AR at all, an empty record data object with length 0 is returned.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 119 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

7.3.1.2.5.10.3 Read AR Data for device

This service allows reading AR Data using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 119 for its structure.

If there are no AR at all, an empty record data object with length 0 is returned.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

7.3.1.2.5.11 Read Expected Fast Startup Data

This service allows reading Expected Fast Startup Data using a record data object with AR scope (see 7.3.1.6.1).

This service is only supported for ARs in legacy startup mode; the service is substituted for ARs in advanced startup mode by the AR FSU Block within the connect service.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 120 for its structure.

The AR context is needed for this service; see 7.3.1.6.6.4 for details.

Table 120 – Data elements of Expected Fast Startup Data

Parameter name	service specific elements	rsp	cnf
Data	Expected Fast Startup Data	M	M(=)
	FS Parameter Block	C	C(=)
	FS Parameter Mode	M	M(=)
	FS Parameter UUID	M	M(=)
	Manufacturer Specific Fast Startup Data	C	C(=)

Data

The parameter contains the attributes as in Table 120 (parameter description in 7.3.1.2.4).

7.3.1.2.5.12 Write Expected Fast Startup Data

This service allows writing Expected Fast Startup Data using a record data object with AR scope (see 7.3.1.6.1).

This service is only supported for ARs in legacy startup mode; the service is substituted for ARs in advanced startup mode by the AR FSU Block within the connect service.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 120 for its structure.

The AR context is needed for this service; see 7.3.1.6.6.4 for details.

Data

See Read Expected Fast Startup Data in 7.3.1.2.5.11.

7.3.1.2.6 Behavior

7.3.1.2.6.1 General

If the attribute AR State of the desired AR is set to BACKUP, all data records are neglected during the parameterization sequence (see 7.3.1.6.6.3 item b)).

AR State influences also the behavior of the Application Relationship class's service Application Ready (see 7.3.1.2.5.5). Note that the Application Ready service confirms a parameterization sequence due to application relation establishment, plug/release of a submodule, or Prm Begin.

If the AR state is set to BACKUP when the service Prm End is initiated by the IO controller, the Submodule State.AR Info of all submodules related to the parameterization sequence is set to the submodule's state according to the AR with AR state set to PRIMARY or (if not existing) set to FIRST.

If submodules are added to the expected identification (dynamic reconfiguration), which are not yet contained in the PRIMARY or FIRST AR, the submodule's AR state is set to NO_SUBMODULE. If later on the AR used for dynamic reconfiguration is set to PRIMARY and the added submodules are existent or have a decent substitute, they are reported to the AR with a plug alarm and will be parameterized within the plug sequence.

7.3.1.2.6.2 IO device during startup

7.3.1.2.6.2.1 Common

This subclause describes the application behavior of an IO device during startup of the user application.

7.3.1.2.6.2.2 Primitive definitions

7.3.1.2.6.2.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by an IO device during startup state machine DEVSM are described in the service definition and shown in Table 121.

Table 121 – Remote primitives issued or received by DEVSM

Primitive	Source	Destination	Associated parameters	Functions
Alarm.cnf (+)	FAL	DEVSM	see 7.3.1.5.5	—
Application Ready.cnf (+)	FAL	DEVSM	see 7.3.1.2.5.5	—
Connect.ind	FAL	DEVSM	see 7.3.1.2.5.1.2	—
Prm End.ind	FAL	DEVSM	see 7.3.1.2.5.4	—
Read Input Data.ind	FAL	DEVSM	see 7.3.2.2.5.10	—
Read Output Data.ind	FAL	DEVSM	see 7.3.2.2.5.11	—
Write.ind	FAL	DEVSM	see 7.3.1.6.5.2.1	—
Alarm.req	DEVSM	FAL	see 7.3.1.5.5	—
Application Ready.req	DEVSM	FAL	see 7.3.1.2.5.5	—
Connect.rsp (+)	DEVSM	FAL	see 7.3.1.2.5.1.2	—
Prm End.rsp (+)	DEVSM	FAL	see 7.3.1.2.5.4	—
Read Input Data.rsp (-)	DEVSM	FAL	see 7.3.2.2.5.10	—
Read Input Data.rsp (+)	DEVSM	FAL	see 7.3.2.2.5.10	—
Read Output Data.rsp (-)	DEVSM	FAL	see 7.3.2.2.5.11	—

Primitive	Source	Destination	Associated parameters	Functions
Read Output Data.rsp (+)	DEVSM	FAL	see 7.3.2.2.5.11	—
Write.rsp (-)	DEVSM	FAL	see 7.3.1.6.5.2.1	—
Write.rsp (+)	DEVSM	FAL	see 7.3.1.6.5.2.1	—

7.3.1.2.6.2.2 Primitives exchanged between local machines

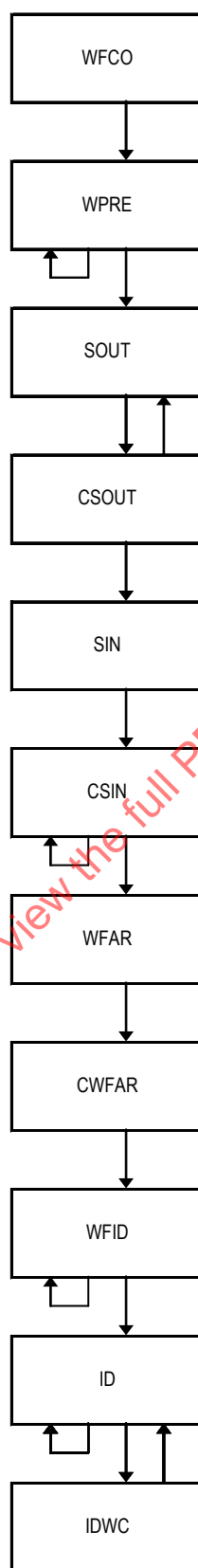
The local service primitives including their associated parameters issued or received by DEVSM are described in the service definition and shown in Table 122.

Table 122 – Local primitives issued or received by DEVSM

Primitive	Source	Destination	Associated parameters	Functions
Local Get Output_cnf (+)	FAL	DEVSM	see 7.3.1.3.5.8	—
Local Get Output_req	DEVSM	FAL	see 7.3.1.3.5.8	—
Local New Output_ind	FAL	DEVSM	see 7.3.1.3.5.10	—
Local Set Input_cnf (+)	FAL	DEVSM	see 7.3.1.3.5.1	—
Local Set Input_req	DEVSM	FAL	see 7.3.1.3.5.1	—
Local Set Output IOCS_cnf (+)	FAL	DEVSM	see 7.3.1.3.5.7	—
Local Set Output IOCS_req	DEVSM	FAL	see 7.3.1.3.5.7	—

7.3.1.2.6.2.3 State transition diagram

Figure 26 shows the state transition diagram DEVSM.

**Figure 26 – State transition diagram DEVSM**

States of the DEVSM

WFCO	Wait for a Connect indication
WPRE	Wait for a Prm End indication
SOUT	Initialize output according parameterization
CSOUT	Wait for output initialization
SIN	Initialize input buffer according to the real inputs
CSIN	Wait for input buffer initialization
WFAR	Wait for user application ready
CWFAR	Wait for Application Ready response
WFID	Wait for IN Data
ID	Working state IN Data
IDWC	Wait for input buffer or output update

7.3.1.2.6.2.4 State machine description

This state machine describes the application behavior of an IO device during startup of the user application.

7.3.1.2.6.2.5 State table

Table 123 defines the state table DEVSM.

Table 123 – State table DEVSM

#	Current State	Event /Condition =>Action	Next State
1	WFCO	Connect.ind () /CheckExpectedCnf()=Ok => SetConnectResponseData("Zero") StoredSessionKey:=AR Parameter Block.SessionKey MCR_IOPS_BAD := FALSE Connect.rsp (+)	WPRE
2	WFCO	Connect.ind () /CheckExpectedCnf()!=Ok => DeltaConfiguration=AdaptConfiguration() StoredSessionKey:=AR Parameter Block.SessionKey SetConnectResponseData(DeltaConfiguration) MCR_IOPS_BAD := FALSE Connect.rsp (+)	WPRE
3	WPRE	Write.ind () /CheckParameter()=Ok => StoreParameter() Write.rsp (+)	WPRE
4	WPRE	Write.ind () /CheckParameter()!=Ok => SetWrErrorResponseData("ErrorCode=PNIORW", OTHER=UserSpecific) Write.rsp (-)	WPRE
5	WPRE	Prm End.ind () => Prm End.rsp (+)	SOUT

#	Current State	Event /Condition =>Action	Next State
6	SOUT	/Substitution Mode == REPLACEMENT_VALUE => For all involved slot/subslots do: Slot.Subslot.IOPS := BAD_BY_DEVICE Output Data object subelement value:=Substitution Value Slot.Subslot.IOCS:=GOOD For all involved slot/subslots do: Local Set Output IOCS_req ()	CSOUT
7	SOUT	/Substitution Mode == 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: Local Set Output IOCS_req ()	CSOUT
8	SOUT	/Substitution Mode == 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: Local Set Output IOCS_req ()	CSOUT
9	CSOUT	Local Set Output IOCS_cnf (+) /not last one =>	SOUT
10	CSOUT	Local Set Output IOCS_cnf (+) /last one =>	SIN
11	CSOUT	Local Set Output IOCS_cnf (+) => ignore (M-Consumer submodule)	SIN
12	SIN	=> For all involved slot/subslots with submodule 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 submodule status BAD do: Slot := Slot Number; Subslot := Subslot Number IOPS := BAD_BY_DEVICE local IOCS:=BAD_BY_DEVICE For all involved slot/subslots do: Local Set Input_req ()	CSIN
13	CSIN	Local Set Input_cnf (+) /not last one => ignore	CSIN
14	CSIN	Local Set Input_cnf (+) /last one =>	WFAR
15	WFAR	UserApplicationReady () => SessionKey:=StoredSessionKey MCR_IOPS_BAD := FALSE Application Ready.req ()	CWFAR
16	CWFAR	Application Ready.cnf (+) =>	WFID

#	Current State	Event /Condition =>Action	Next State
17	WFID	Local New Output_ind () /not last one => ignore	WFID
18	WFID	Local New Output_ind () /last one (exclude CREPs with Multicast Communication) => SumARinData ()	ID
19	ID	Local New Output_ind () => Local Get Output_req ()	IDWC
20	ID	Local New Output_ind () => "real output values are not modified, possible actions are local matter or for future use with redundancy"	ID
21	ID	Local New Output_ind () /(CREP with Multicast Communication) && (CHECK_IOPS(CREP) == FALSE) && (MCR_IOPS_BAD == TRUE) => ignore	ID
22	ID	Local New Output_ind () /(CREP with Multicast Communication) && (CHECK_IOPS(CREP) == FALSE) && (MCR_IOPS_BAD == FALSE) => MCR_IOPS_BAD := TRUE Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelErrorType := Multicast Provider Status Mismatch ExtChannelErrorType := Provider state changed to value bad Alarm.req ()	ID
23	ID	Local New Output_ind () /(CREP with Multicast Communication) && (CHECK_IOPS(CREP) == TRUE) && (MCR_IOPS_BAD == TRUE) => MCR_IOPS_BAD := FALSE Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8000 ChannelNumber := 0x8000 ChannelProperties.Specifier := DISAPPEARS DISAPPEARS_BUT_OTHER_REMAIN ALL_DISAPPEARS ChannelErrorType := Multicast Provider Status Mismatch Alarm.req ()	ID
24	ID	Local New Output_ind () /(CREP with Multicast Communication) && (CHECK_IOPS(CREP) == TRUE) && (MCR_IOPS_BAD == FALSE) => Local Get Output.req ()	IDWC
25	IDWC	Local Get Output_cnf (+) /IOPS==GOOD => Slot.Subslot.IOPS := GOOD Output Data object subelement value:=Subslot Output Data	ID
26	IDWC	Local Get Output_cnf (+) /IOPS==BAD_BY_CONTROLLER & Substitution Mode == REPLACEMENT_VALUE => Slot.Subslot.IOPS := BAD_BY_CONTROLLER Output Data object subelement value:=Substitution Value	ID

#	Current State	Event /Condition =>Action	Next State
27	IDWC	Local Get Output_cnf (+) /IOPS==BAD_BY_CONTROLLER & Substitution Mode == LAST_VALUE => Slot.Subslot.IOPS := BAD_BY_CONTROLLER	ID
28	IDWC	Local Get Output_cnf (+) /IOPS==BAD_BY_CONTROLLER & Substitution Mode == ZERO => Slot.Subslot.IOPS := BAD_BY_CONTROLLER Output Data object subelement value:=ZERO	ID
29	IDWC	Local Get Output_cnf (+) /IOPS==BAD_BY_PROVIDER & Substitution Mode == REPLACEMENT_VALUE/LAST_VALUE/ZERO => Slot.Subslot.IOPS := BAD_BY_PROVIDER Output Data object subelement value:=REPLACEMENT_VALUE/LAST_VALUE/ZERO MCR_IOPS_BAD := TRUE Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8002 ChannelNumber := 0x8000 ChannelProperties.Specifier := Appears ChannelErrorType := Multicast Provider Status Mismatch ExtChannelErrorType := Provider state changed to value bad Alarm.req ()	ID
30	IDWC	Local Get Output_cnf (+) /IOPS==GOOD && MCR_IOPS_BAD == TRUE => Slot.Subslot.IOPS := GOOD Output Data object subelement value:=Output Data MCR_IOPS_BAD := FALSE Alarm Type:=Multicast Communication Mismatch Alarm User Structure Identifier := 0x8000 ChannelNumber := 0x8000 ChannelProperties.Specifier := DISAPPEARS DISAPPEARS_BUT_OTHER_REMAIN ALL_DISAPPEARS ChannelErrorType := Multicast Provider Status Mismatch Alarm.req ()	ID
31	ID	Alarm.cnf () => ignore	ID
32	ID	Read Output Data.ind () /ARUID==AREP.ARUID&& 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.rsp (+)	ID

#	Current State	Event /Condition =>Action	Next State
33	ID	Read Output Data.ind () /(ARUUIID==AREP.ARUUIID&& 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.rsp (-)	ID
34	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: Local Set Input_req ()	IDWC
35	ID	Read Input Data.ind () /ARUUIID==AREP.ARUUIID&& 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.rsp (+)	ID
36	ID	Read Input Data.ind () /(ARUUIID==AREP.ARUUIID&& 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.rsp (-)	ID

7.3.1.2.6.2.6 Functions, Macros, Timers and Variables

Table 124 contains the functions, macros, timers and variables used by DEVSM and their arguments and descriptions.

Table 124 – Functions, Macros, Timers and Variables by DEVSM

Name	Type	Function/Meaning
CHECK_IOPS(CREP)	Function	All IO data objects of CREP dispose of IOPS==GOOD
UserApplicationReady	Function	This local function is issued from the user after all parameterization is done and input data is updated
SumARinData	Function	This local function is issued from DEVSM if all consumer reaches the state Data
DeltaConfiguration	Variable	Is the difference between the expected and real configuration
StoredSessionKey	Variable	This local variable stores the session key from the connect service
SetConnectResponseData	Function	This local function configures the module difference block for the connect response using the delta configuration
AdaptConfiguration	Function	This local function adapt the IO data layout of real configuration to the expected configuration
StoreParameter	Function	This local function stores the parameters with respect to the persistent flag
SetWrErrorResponseData	Function	This local function configures the response negative PDU
MCR_IOPS_BAD	Variable	This variable is used for the delta alarm generation of the multicast communication relation diagnosis

RecordDataObject are stored by the IO device. When receiving the Prm End.ind a “full” RecordDataObjects check (locally defined rules) is done by the application. If the RecordDataObjects suit, they are stored by the application. This may already be done by Prm End.rsp or even later. From an “outside” view the takeover of the RecordDataObjects shall be done with ApplicationReady.req.

7.3.1.2.6.3 IO controller during startup

7.3.1.2.6.3.1 General

This subclause describes the client application behavior of an IO controller during startup CTLSM of the user application as an example.

Note that this subclause provides selected behavior only and is thus not a comprehensive definition of the client behavior.

In the startup phase data records shall be written in the following transfer sequence:

- AR related data records shall be transferred first in ascending order according to their index (index is between 0xE000 and 0xEBFF).

NOTE The only writeable AR related data record is ARFSUDDataAdjust having index 0xE050. This data record is supported only if the startup mode of the AR is legacy.

- Within the IO device, all APIs are processed in ascending order.
- Within each API, all slots carrying a module are processed in ascending order.
- Within each module, all subslots carrying a submodule are processed in ascending order.
- Within each submodule,
 - data records defined by IEC 61158-6-10 (index is between 0x8000 and 0xAFFF) will always be transferred first in ascending order according to their index.
 - manufacturer and profile specific data records (index is between 0x0000 and 0x7FFF resp. between 0xB000 and 0xBFFF) will follow afterwards
 - according to their attribute @TransferSequence as given in the device's GSD,

- or if the attribute @TransferSequence is not stated, in ascending order according to their index.

Data records having an index between 0xC000 and 0xDFFF or between 0xECFF and 0xFFFF are always read-only. Therefore they do not appear within the transfer sequence.

If a data record contains indexable elements, these should be sorted in ascending order (order criteria: Block Type, API, Slot Number, Subslot Number).

7.3.1.2.6.3.2 Primitive definitions

7.3.1.2.6.3.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by CTLSM are described in the service definition and shown in Table 125.

Table 125 – Remote primitives issued or received by CTLSM

Primitive	Source	Destination	Associated parameters	Functions
ApplicationReady.ind	FAL	CTLSM	see 7.3.1.2.5.5	—
Connect.cnf (+)	FAL	CTLSM	see 7.3.1.2.5.1.2	—
Prm End.cnf	FAL	CTLSM	see 7.3.1.2.5.4	—
Write.cnf	FAL	CTLSM	see 7.3.1.6.5.2.1	—
ApplicationReady.rsp ()	CTLSM	FAL	see 7.3.1.2.5.5	—
Connect.req	CTLSM	FAL	see 7.3.1.2.5.1.2	—
Prm End.req	CTLSM	FAL	see 7.3.1.2.5.4	—
Write.req	CTLSM	FAL	see 7.3.1.6.5.2.1	—

7.3.1.2.6.3.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by CTLSM are described in the service definition and shown in Table 126.

Table 126 – Local primitives issued or received by CTLSM

Primitive	Source	Destination	Associated parameters	Functions
Local Set Input IOCS_cnf (+)	FAL	CTLSM	see 7.3.1.3.5.2	—
Local Set Output_cnf (+)	FAL	CTLSM	see 7.3.1.3.5.6	—
Local Set Input IOCS_req	CTLSM	FAL	see 7.3.1.3.5.2	—
Local Set Output_req	CTLSM	FAL	see 7.3.1.3.5.6	—
Local New Input_ind	FAL	CTLSM	see 7.3.1.3.5.5	—

7.3.1.2.6.3.3 State transition diagram

Figure 27 shows the State transition diagram for the startup.

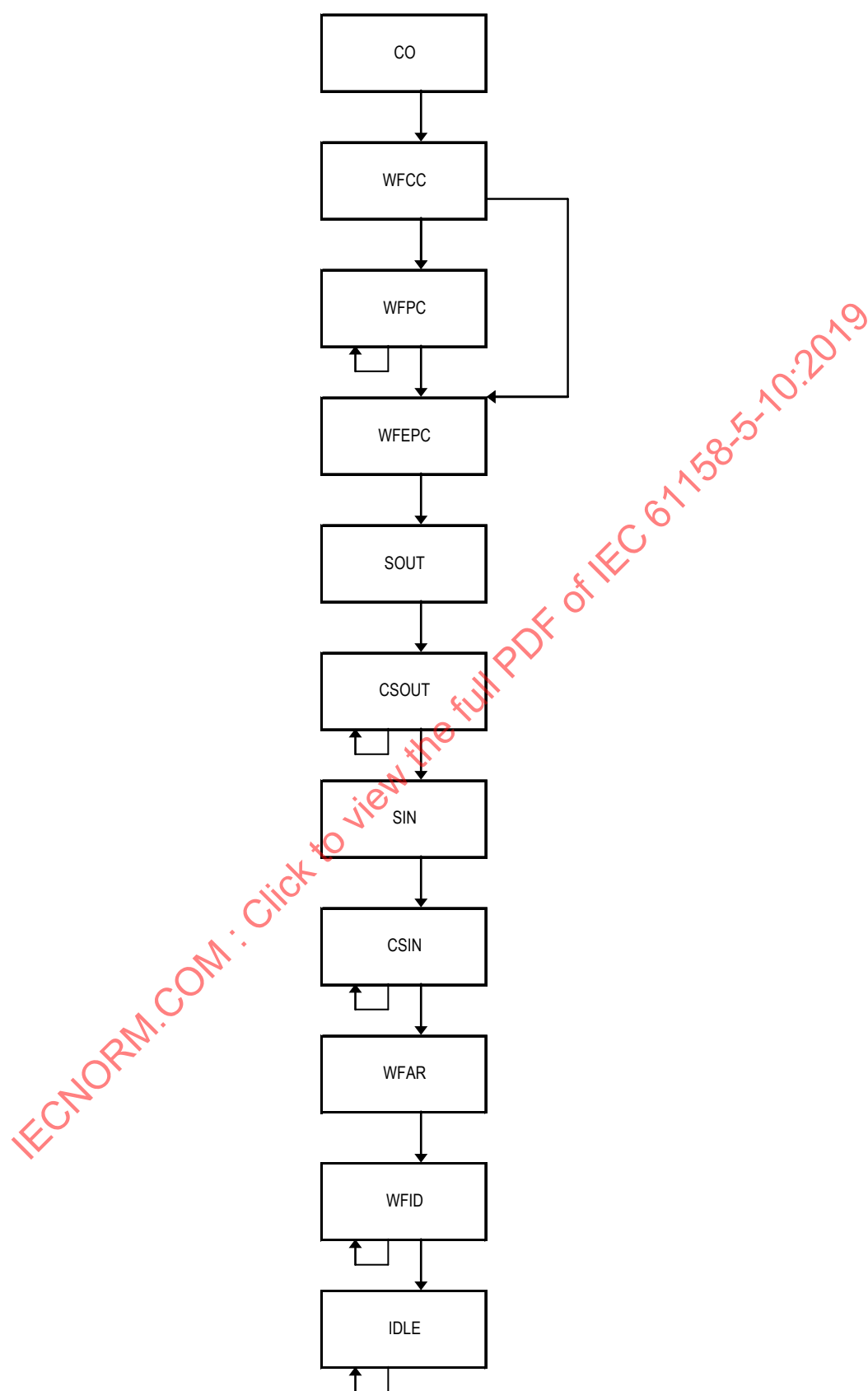


Figure 27 – State transition diagram CTLSM

States of CTLSM

CO	Initiate the connect request
WFCC	Wait for a connect response
WFPC	Wait for a write confirmation
WFEPc	Wait for Prm End confirmation
SOUT	Initialize output according parameterization
CSOUT	Wait for output initialization
SIN	Initialize input buffer according to the real inputs
CSIN	Wait for input buffer initialization
WFAR	Wait for ApplicationReady indication
WFID	Wait for IN Data
IDLE	Working state IN Data

7.3.1.2.6.3.4 State machine description

This state machine describes the application behavior of an IO controller during the startup of a particular application relation to an IO device.

7.3.1.2.6.3.5 State table

Table 127 defines the state table for the IO controller during startup.

Table 127 – State table CTLSM

#	Current State	Event /Condition =>Action	Next State
1	CO	=> SetConnectRequestData("ExpectedConfiguration") AR Parameter Block.SessionKey:=StoredSessionKey+1 StoredSessionKey:=StoredSessionKey+1 Connect.req ()	WFCC
2	WFCC	Connect.cnf (+) /CheckResponseBlocks()=Ok && ParameterToSend()=Ok => PrepareWritePDU () Write.req ()	WFPC
3	WFCC	Connect.cnf (+) /CheckResponseBlocks()=Ok && ParameterToSend()!=Ok => Prm End.req ()	WFEPc
4	WFCC	Connect.cnf (+) /CheckResponseBlocks()!=Ok && ParameterToSend()=Ok => AdaptConfiguration() Write.req ()	WFPC
5	WFCC	Connect.cnf (+) /CheckResponseBlocks()!=Ok && ParameterToSend()!=Ok => AdaptConfiguration() Prm End.req ()	WFEPc

#	Current State	Event /Condition =>Action	Next State
6	WFPC	Write.cnf () /ParameterToSend()=Ok => Session Key=StoredSessionKey PrepareWritePDU () Write.req ()	WFPC
7	WFPC	Write.cnf () /ParameterToSend()!=Ok => Session Key=StoredSessionKey Prm End.req ()	WFEPc
8	WFEPc	Prm End.cnf () =>	SOUT
9	SOUT	=> For all involved slot/subslots do: Slot.Subslot.IOCS := GOOD Subslot Output Data:=Slot.Subslot.Initial Value For all involved slot/subslots do: Local Set Output_req ()	CSOUT
10	CSOUT	Local Set Output_cnf (+) /not last one =>	CSOUT
11	CSOUT	Local Set Output_cnf (+) /last one =>	SIN
12	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: Local Set Input IOCS_req ()	CSIN
13	CSIN	Local Set Input IOCS_cnf (+) /not last one =>	CSIN
14	CSIN	Local Set Input IOCS_cnf (+) /last one =>	WFAR
15	WFAR	ApplicationReady.ind () /Session Key=StoredSessionKey => ApplicationReady.rsp ()	WFID
16	WFID	Local New Input_ind () /not last one =>	WFID
17	WFID	Local New Input_ind () /last one => SumARinData ()	IDLE
18	IDLE	=> ignore	IDLE

7.3.1.2.6.3.6 Functions, Macros, Timers and Variables

Table 128 contains the functions, macros, timers and variables used by CTLSM and their arguments and descriptions.

Table 128 – Functions, Macros, Timers and Variables used by CTLSM

Name	Type	Function/Meaning
CheckResponseBlocks	Function	This local function checks whether the syntax and the semantics of the response is correct
ParameterToSend	Function	This local function indicates the existence of any parameter record
AdaptConfiguration	Function	This local function adapts the list of parameter records according to the connect response
SumARinData	Function	This local function is issued from DEVSM if all consumer reaches the state Data
StoredSessionKey	Variable	This local variable stores the session key from the connect service
SetConnectRequestData	Function	This local function configures the connect request
PrepareWritePDU	Function	This local function fetches the parameter from the local data base and constructs the write multiple PDU

7.3.1.3 Communication Relationship class

7.3.1.3.1 Overview

7.3.1.3.1.1 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.

7.3.1.3.1.2 Communication relationship types

The 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

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

7.3.1.3.1.3 Communication Relationship endpoint

CRs are defined as a set of co-operating Communication Relationship End Points (CREPs). The local context of a CR endpoint is used to control the conveyance of APDUs on the CR. Figure 28 illustrates the Assignment of Communication Relationship to Application Relationship.

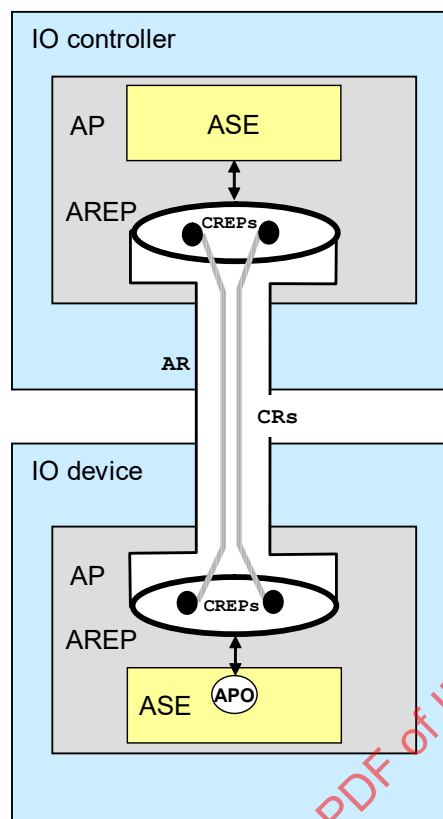


Figure 28 – Assignment of Communication Relationship to Application Relationship

A communication relationship is characterized by all of the following basic roles of the CREP used:

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.

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.

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.

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.

7.3.1.3.1.4 Buffer and queues

CREPs may be modeled as a queue or as a buffer.

Each application data transferred over a queued CREP is delivered in the order received for conveyance. Reading the application data from the queue removes the element from the queue.

Application data transferred over a buffered CREP is different. In this case, application data is placed in an application layer buffer for transfer. If the CREP receives another conveyance request, it replaces the previous contents of the buffer. The contents of the buffer are preserved until the next request updates it. The application shall convey application data, such that the data of a particular submodule is consistently replaced.

The application data is transferred between the transferring and receiving CREP using the real time cyclic ASE. The transport is cyclical and thus asynchronous to the access of the application.

Thus the following behaviors are possible:

- Local set input/output requests are issued faster than the contents of the buffer are transported over the network. In this case only the values provided with the latest service are conveyed over the network.
- Local set input/output requests are issued slower than the contents of the buffer are transported over the network. In this case the values provided with each service are conveyed more than once over the network.
- Local set input/output requests are issued synchronous to the transport of the contents of the buffers over the network. In this case the values provided with each service are conveyed once over the network.

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. 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.

7.3.1.3.1.5 IO data

Submodules plugged into subslots provide input and output data that is cyclically conveyed over the network using BBUU (Buffer Buffer Unconfirmed Unidirectional) and/or BBUU-OM (Buffer Buffer Unconfirmed Unidirectional - One to Many).

Input and output data is located in two classes:

- The real identification ASE provides input and output data according to the real identification of the submodule; i.e. structured as provided by the APOP itself. There is one instance of data per real submodule.
- The communication relationship class provides input and output data according to the expected identification of the submodule; i.e. structured according to the expected identification of a particular application relationship. There is one instance of data per expected identification of a submodule; in other words possible multiple instances of data for a particular submodule (if being used in multiple application relationship; e.g. as shared input) or even no instance of data (if not being used in an application relationship).

The communication relationship provides the data structured according to the expected identification. If the real identification is a proper substitute for the expected identification the structure of the data is adapted to match the expected identification (e.g. if a DI16 as real submodule is a DI8 substitute by converting Unsigned16 to Unsigned8).

The usage of reduced input or output submodule data length (see 7.3.1.4.4) adapts the real submodule's data by omitting all data.

The usage of discard IOXS (see 7.3.1.4.4) adapts the real submodule's data by omitting IOXS.

Input Data objects are partitioned in Input Data Elements which represent the structure of input data of the submodule. Output Data objects are partitioned in Output Data Elements which represent the structure of output data of the submodule.

The Communication Relationship class uses the Client/Server and Provider/Consumer access model.

The client application conveys 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 Communication Relationship class is described next, followed by a description of its services. The relations between real IO data structure and conveyance of IO data via ARs are described within this class.

Figure 29 shows the IO service interactions for the IO controller and IO device.

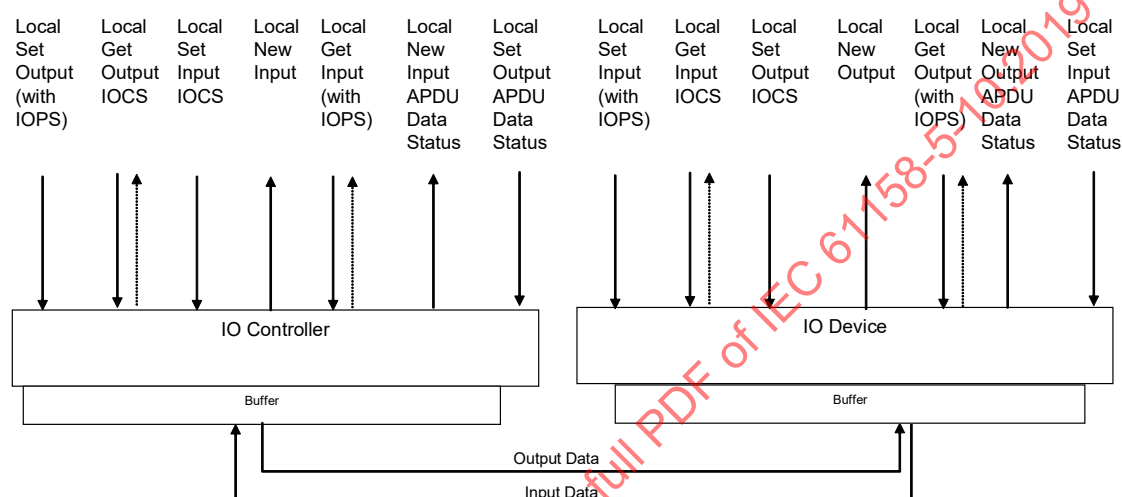


Figure 29 – Overview Communication Relationship Class service interactions

The services interact with local buffers maintained by this class and updated by the application using shown local services in Figure 29. 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 IO Data shall be transmitted consistent from the sending buffer to the receiving buffer completely. On the communication network, the whole frame data shall be consistent. For the application, consistency of each submodule data is required.

If 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.

If 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.

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.

7.3.1.3.2 Relations

The services are bound to other services as shown in Table 129:

Table 129 – Binding communication relationship services

Service	Binding	Usage
Local Set Input	RTC::PPM Set Prov Data	Used by FAL
Local Set Input IOCS	RTC::PPM Set Prov Data	Used by FAL
Local Get Input	RTC::CPM Get Cons Data	Used by FAL
Local Get Input IOCS	RTC::CPM Get Cons Data	Used by FAL
Local New Input	RTC::CPM New Data Indication	Used by FAL
Local Set Output	RTC::PPM Set Prov Data	Used by FAL
Local Set Output IOCS	RTC::PPM Set Prov Data	Used by FAL
Local Get Output	RTC::CPM Get Cons Data	Used by FAL
Local Get Output IOCS	RTC::CPM Get Cons Data	Used by FAL
Local New Output	RTC::CPM New Data Indication	Used by FAL
Local Set Provider State	RTC::PPM Set Prov Status	Used by FAL
Local Set Redundancy	RTC::PPM Set Prov Status	Used by FAL
Local Set State	RTC::PPM Set Prov Status	Used by FAL
Local Data State Changed	RTC::CPM New Data Indication	Used by FAL

7.3.1.3.3 Template

The Communication Relationship object is described by the following template:

ASE:	AR ASE
CLASS:	Communication Relationship
CLASS ID:	not used
PARENT CLASS:	TOP
ATTRIBUTES:	
1	(m) Key Attribute: Implicit
2	(m) Attribute: List of CRs
2.1	(m) Attribute: AREP
2.2	(m) Attribute: CREP
2.3	(m) Attribute: IO CR Type
2.4	(m) Attribute: IO CR Reference
2.5	(m) Attribute: LT Field
2.6	(m) Attribute: IO CR Properties
2.6.1	(m) Attribute: RT Class
2.7	(m) Attribute: C_SDU Length
2.8	(m) Attribute: Frame ID
2.9	(m) Attribute: Send Clock Factor
2.10	(m) Attribute: Reduction Ratio
2.11	(m) Attribute: Phase
2.12	(m) Attribute: Sequence
2.13	(m) Attribute: Frame Send Offset
2.14	(m) Attribute: Data Hold Factor
2.15	(m) Attribute: IO CR Tag Header
2.15.1	(m) Attribute: Vlan ID
2.15.2	(m) Attribute: IO CR User Priority
2.16	(m) Attribute: IO CR Multicast MAC Add
2.17	(m) Attribute: List of IO Data Objects
2.17.1	(m) Attribute: API

2.17.2	(m) Attribute:	Slot Number
2.17.3	(m) Attribute:	Subslot Number
2.17.4	(m) Attribute:	IO Data Object Frame Offset
2.17.5	(m) Attribute:	IO Data Object Format
2.17.5.1	(m) Attribute:	Length Data
2.17.5.2	(m) Attribute:	Data Structure
2.17.5.3	(m) Attribute:	Length IOPS
2.17.5.4	(c) Attribute:	IOPS
2.18	(m) Attribute:	List of IOCS
2.18.1	(m) Attribute:	API
2.18.2	(m) Attribute:	Slot Number
2.18.3	(m) Attribute:	Subslot Number
2.18.4	(m) Attribute:	IOCS Frame Offset
2.18.5	(m) Attribute:	Length IOCS
2.18.6	(c) Attribute:	IOCS
2.19	(o) Attribute:	List of Subframe Data
2.19.1	(m) Attribute:	Subframe Position
2.19.2	(m) Attribute:	Subframe Length
2.20	(o) Attribute:	IR Frame Data
2.20.1	(m) Attribute:	IOCR Tx Ports
2.20.1.1	(m) Attribute:	Port
2.20.1.2	(m) Attribute:	Redundant Port
2.20.2	(m) Attribute:	Frame Send Offset
2.21	(o) Attribute:	Multicast Consumer Data
2.21.1	(m) Attribute:	Address Resolution Properties
2.21.2	(m) Attribute:	MCI Timeout Factor
2.21.3	(m) Attribute:	Provider Station Name
2.21.4	(m) Attribute:	Multicast Provider State

SERVICES:

1	(m) OpsService:	Local Set Input
2	(m) OpsService:	Local Set Input IOCS
3	(m) OpsService:	Local Get Input
4	(m) OpsService:	Local Get Input IOCS
5	(o) OpsService:	Local New Input
6	(m) OpsService:	Local Set Output
7	(m) OpsService:	Local Set Output IOCS
8	(m) OpsService:	Local Get Output
9	(m) OpsService:	Local Get Output IOCS
10	(o) OpsService:	Local New Output
11	(m) OpsService:	Local Set Provider State
12	(m) OpsService:	Local Set Redundancy
13	(m) OpsService:	Local Set State
14	(m) OpsService:	Local Data State Changed

7.3.1.3.4 Attributes**Implicit**

The key attribute Implicit indicates that the CRL object is implicitly addressed by the service.

List of CRs

This attribute is composed of the following list elements:

AREP

This attribute identifies the AR this CR belongs. It is a cross reference to the application relationship class.

CREP

This attribute contains the local identifier of this CR. The attribute is used to uniquely identify a particular CR. It is also used for cross-referencing. It is used for internal handling only.

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. It is defined by the engineering system as a unique identifier for CR blocks within an AR.

Attribute type: Unsigned16

LT Field

This attribute contains the length type field of the Physical Layer Frame.

Attribute type: Unsigned16

Allowed values: 0x8892 = IEC_61158_TYPE_10, others reserved

IO CR Properties

This attribute consists of the following attributes:

RT Class

The value RT Class defines the Frame ID range (see Table 130) to be used for communication and in consequence a traffic class (RED or GREEN). The traffic class is used at the provider of the frame on its injection. The traffic class is used in the bridge for forwarding by selecting the used transport mechanism and the switching model (see 6.3.12, 6.3.13, and 6.3.15).

NOTE 1 This attribute is called RT Class for historic reasons. It does not address the RT Class used in other parts of this document.

Attribute type: Unsigned8

Allowed values: 0x01 (only to be used for legacy devices), 0x02, 0x03 and 0x04

Table 130 – Traffic Classes versus RT Class

RT Class	Startup Mode	Frame ID Range	Traffic Class	Used for
0x01	Legacy	range 7	GREEN	RT_CLASS_1 (legacy)
0x02	Advanced, Legacy	range 6	GREEN	RT_CLASS_1, RT_CLASS_2 (legacy)
0x03	Advanced, Legacy	range 3	RED	RT_CLASS_3
0x04	Advanced, Legacy	range 7	GREEN	RT_CLASS_UDP

For the values of the particular frame ID ranges see IEC 61158-6-10.

In traffic class GREEN IEEE 802.3, IEEE 802.1Q, and IP-Suite (see 6.3.15) are applicable without any extension.

In traffic class RED IEEE 802.3 and IEEE 802.1Q are applicable with extensions specified in 6.3.12, 6.3.13, and 6.3.15.

C_SDU Length

This attribute contains the overall length of the transmitted data (CSDU or CSFSDU length). For a dynamically packed frame it indicates the length of all subframes contained in this frame.

Attribute type: Unsigned16

Allowed Values for RT_CLASS_1, RT_CLASS_2 and RT_CLASS_3: 40 to 1 440

NOTE 1 The lower bound of the C_SDU length is derived from the minimum frame size of an Ethernet frame (= 64 octets). On subtracting the size of the header (i.e. destination address, source address and type) and the size of the footer (i.e. FCS) 46 octets are left. On subtracting the size of the RT frame header (i.e. Frame ID) and footer (i.e. cycle counter, data status and transfer status) the minimum length of the C_SDU is 40 octets.

Allowed Values for RT_CLASS_UDP: 12 to 1 440

NOTE 2 The lower bound of the C_SDU length is derived from the minimum frame size of an Ethernet frame (= 64 octets). On subtracting the size of the header (i.e. destination address, source address and type) and the size of the footer (i.e. CRC) 46 octets are left. From these 46 octets the IP header (20 octets) and the UDP header (8 octets) are subtracted, leaving 18 octets. On subtracting the size of the RT frame header (i.e. Frame ID) and footer (i.e. cycle counter, data status and transfer status) the minimum length of the C_SDU is 12 octets.

Frame ID

This attribute contains the identifier of the data within the frame. The value of the frame ID is provided by either the consumer of the frame or else by the engineering system.

The frame ID shall be in the range as defined by the attribute RT Class.

Attribute type: Unsigned16

The allowed values according to the Frame ID range tables in IEC 61158-6-10 shall be used.

Send Clock Factor

This attribute contains the Send Clock Factor for the CR.

Attribute type: Unsigned16

Time Base: 31,25 µs

Allowed Values: 1 to 128

Default Value: 32

Reduction Ratio

This attribute contains the reduction ratio for the CR. The actual send cycle of the CR is Reduction Ratio multiplied with the Send Clock Factor multiplied with the time base of the Send Clock Factor (31,25 µs).

Attribute type: Unsigned16

Allowed Values: see IEC 61158-6-10

Phase

This attribute indicates in which particular send cycle the data has to be sent.

Attribute type: Unsigned16

Allowed values: 1 to selected Reduction Ratio.

NOTE 3 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.

Allowed Values for RT_CLASS_1, RT_CLASS_2 and RT_CLASS_UDP: 0x0000 (mandatory), 0x0001 to 0xFFFF (optional)

Allowed Values for RT_CLASS_3: 0x0000

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.

This Frame Send Offset is duplicated in the PDIRFrameData for the appropriate frame used to transport this CR. Use the value from PDIRFrameData.

Attribute type: Unsigned32

Allowed Values for RT_CLASS_1, RT_CLASS_2 and RT_CLASS_UDP: BEST_EFFORT (mandatory; as soon as possible)

Allowed Values for RT_CLASS_3: BEST_EFFORT (optional; as soon as possible), 0x00000000 – 0x003D08FF (mandatory; for optimized scheduling)

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. 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 (Time Base: Send Clock Factor * 31,25 µs * Reduction Ratio). The Data Hold Time shall be equal to or less than 1,92 s if the value of the AR attribute RT Class is 0x01, 0x02 or 0x03 and equal to or less than 49,152 s if the value of the AR attribute RT Class is 0x04.

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 IEC 61158-6-10 if the IO CR Type is MULTICAST_PROVIDER_CR or MULTICAST_CONSUMER_CR. The value shall be set to zero for all other IO CR Types.

Attribute type: OctetString[6]

List of IO Data Objects

This attribute contains the IO Data Objects that are placed in the CR.

This list describes where the IO Data Objects of the addressed submodules (indicated by API, Slot and Subslot Number) according to the expected identification are located in the CR.

For a CR having IOCR Type == INPUT_CR or MULTICAST_PROVIDER_CR the IO Data Objects of submodules with SubmoduleProperties.Type == NO_IO and INPUT, and the “input part” of submodules with SubmoduleProperties.Type == IO.

For a CR having IOCR Type == OUTPUT_CR or MULTICAST_CONSUMER_CR the IO Data Objects of submodules with SubmoduleProperties.Type == OUTPUT, and the “output part” of submodules with SubmoduleProperties.Type == IO.

SubmoduleProperties.Type is taken from the Expected Identification of the addressed submodules.

For a dynamically packed frame only the layout of the subframe related to this IO device is described.

A list element consists of the following attributes:

API

This attribute defines to which API the IO Data Object belongs.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Slot Number

This attribute defines to which slot the IO Data Object 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: 0 to 0x9FFF

IO Data Object Frame Offset

This attribute defines the offset of the IO Data Object in the frame.

The attribute is set to 1 440 in the special case of a submodule with SubmoduleProperties.Type == NO_IO and SubmoduleProperties.Discard-IOPS == TRUE as stated in the Expected Identification.

Otherwise, the sum of IO Data Object Frame Offset, Length Data and Length IOPS shall not exceed the C_SDU Length.

Attribute type: Unsigned16

Allowed values: 0 to 1 439 and 1 440

IO Data Object Format

This attribute consists of the following attributes.

Length Data

This attribute defines the number of octets of the IO Data Object without the IOPS. It corresponds to the Submodule Data Length of the addressed submodule within the Expected Identification.

The attribute is 0 in the following cases: SubmoduleProperties.ReduceInputSubmoduleDataLength == TRUE, SubmoduleProperties.ReduceOutputSubmoduleDataLength == TRUE, or SubmoduleProperties.Type == NO_IO as stated for the addressed submodule in the Expected Identification.

Attribute type: Unsigned16

Allowed values: 0 to 1 439

Data Structure

This attribute contains a sequence of data types according to Clause 5.

The sequence is empty if Length Data is 0.

The data structure is derived by local means from the Submodule Ident Number of the addressed submodule within the Expected Identification.

The data structure is adapted from the real configuration's data structure in the following cases: if "Substitute", the application adapts the data structure according to the expected submodule, if "Reduce" was requested, no data is transmitted.

Length IOPS

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

The attribute is set to 0 in the special case of a submodule with SubmoduleProperties.Type == NO_IO and SubmoduleProperties.DiscardIOXS == TRUE as stated in the Expected Identification.

Attribute type: Unsigned8

Allowed values: 0, 1

IOPS

The attribute defines the provider status for the IO Data Object Element.

The attribute is missing if Length IOPS is 0.

Attribute type: Unsigned8

Allowed values:

GOOD – content of IO Data Object is valid,

BAD_BY_SUBSLOT – content of IO Data Object is invalid; it was detected by the submodule,

BAD_BY_SLOT – content of IO Data Object is invalid; it was detected by the module,

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

Special case: BAD_BY_CONTROLLER – content of IO 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.

List of IOCS

This list describes where the IOCS of the submodules (indicated by API, Slot and Subslot Number) according to the expected identification are located within the CR.

For a CR having IOCR Type == INPUT_CR the IOCS of submodules with SubmoduleProperties.Type == OUTPUT, and the IOCS related to the "output part" of submodules with SubmoduleProperties.Type == IO are listed.

For a CR having IOCR Type == OUTPUT_CR the IOCS of submodules with SubmoduleProperties.Type == NO_IO and INPUT, and the IOCS related to the “input part” of submodules with SubmoduleProperties.Type == IO are listed.

For a CR having IOCR Type == MULTICAST_PROVIDER_CR or MULTICAST_CONSUMER_CR the List of IOCS is empty, it is the relationship to the existing endpoints and not to a application relationship.

For a dynamically packed frame only the layout of the subframe related to this IO device is described.

A list element consists of the following attributes:

API

This attribute defines to which API the IOCS belongs.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

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: 0 to 0x9FFF

IOCS Frame Offset

This attribute defines the offset of the IOCS.

The attribute is set to 1440 in the special case of SubmoduleProperties.DiscardIOXS == TRUE as stated in the Expected Identification.

Otherwise, the sum of IOCS Frame Offset and Length IOCS shall not exceed the C_SDU Length.

Attribute type: Unsigned16

Allowed values: 0 to 1 439

Length IOCS

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

The attribute is set to 0 in the special case of SubmoduleProperties.DiscardIOXS == TRUE as stated in the Expected Identification.

Attribute type: Unsigned8

Allowed values: 0, 1

IOCS

The attribute defines the consumer status for the IO Data object.

The attribute is missing if Length IOCS is 0.

Attribute type: Unsigned8

Allowed values:

GOOD – the IO Data Object could be successfully processed by the application process,

BAD_BY_CONTROLLER – the IO Data Object 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.

List of Subframe Data

This attribute shall be present if dynamic frame packing is used. The attribute describes the sub-structure of the frame. Dynamic frame packing is indicated by using a Frame ID within the DFP range (see the attribute Frame ID above).

The subframes described in the list are packed one after another without gap; thus the following equation holds:

$$SubFrameOffset_n = 2 + \sum_{i=0}^{n-1} SubFrameData_i.SubFrameDataLength \quad (22)$$

where

SubFrameOffset is the subframe offset

SubFrameData the subframe data

SubFrameDataLength is the length of the subframe

All IO Data Objects and IOCS defined within the List of APIs shall be located within exactly one subframe. The following subframe local offsets shall not be used for IO Data Objects or IOCS to keep the subframe header and footer intact: 0, 1, 2, 3, Subframe Data Length - 2, Subframe Data Length - 1.

The own Subframe Position of a particular IO device is indicated in the IR Info Block delivered in the connect request (see 7.3.1.2.5.1.2).

A list element consists of the following attributes:

Subframe Position

This attribute defines the position number of the subframe. The position number shall be strictly increasing within the list of Subframe Data.

Attribute type: Unsigned8

Allowed values: 0x01 to 0x3F for packetizing and unpacking of subframes during transport and end to end; 0x40 to 0x7F for packetizing and unpacking of subframes end to end; 0x00 and 0x80 to 0xFF reserved

Subframe Length

This attribute defines the length of the subframe.

Attribute type: Unsigned8

Allowed values: 0x01 to 0xFF

IR Frame Data

This optional attribute shall be present, if the attribute RT Class is set to 0x03. Otherwise the attribute shall be omitted.

It consists of the following attributes:

IOCR Tx Ports

This attribute consists of

Port

This attribute indicates the port, where the frame shall be sent.

Attribute type: Unsigned8

Allowed values: 0x01 to 0xFF

Redundant Port

This attribute indicates the port, where the redundant frame shall be sent if the Media Redundancy with planned duplication is used.

This attribute shall be set to zero if not used, otherwise this attribute shall be set to the number of the port, where the redundant frame shall be sent

Attribute type: Unsigned8

Allowed values: 0x00 (mandatory), 0x01 to 0xFF (optional)

Frame Send Offset

This attribute contains the relative send offset to the start of the related reduction ratio and phase of the redundant frame. The time base is 1 ns. The value shall be less than the duration of one phase.

The value is only valid, if the Media Redundancy with planned duplication is used.

Attribute type: Unsigned32

Allowed Values: BEST_EFFORT – (mandatory) as soon as possible, 0 – 0x003D08FF for optimized scheduling

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.

Protocol

Multicast consumer uses this protocol to resolve the multicast provider source MAC address.

Attribute type: Unsigned8

Allowed values: DNS, DCP

Resolution Factor

Multicast consumer uses this factor to determine the interval for resolving the multicast provider source MAC address.

Attribute type: Unsigned8

Allowed values for interval: 0x0001 to 0xFFFF

Time base: 1 s

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: 0x0000 to 0x0064

Time base: 100 ms

Provider Station Name

This attribute defines the IP address or name of the Multicast Provider.

The syntax of the attribute CM Initiator Station Name shall be applied.

Attribute type: VisibleString[240]

Multicast Provider State

This parameter is the local status for the multicast consumer CR.

Attribute type: Unsigned16

Allowed values: UP_AND_RUNNING, MULTICAST_PROVIDER_NOT_AVAILABLE

7.3.1.3.5 Services

7.3.1.3.5.1 Local Set Input

The Set Input service sets new input data of one submodule together with its IOPS for next transmission for all related application relationships. Input data and IOPS shall be set consistently. The value of IOPS shall only be "GOOD" if the input data is valid. 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 cyclic buffer. The actual transmission of the buffer is triggered in an own cycle defined with the related parameters in CR and is independent from application cycle.

The Set Input service may set the IOPS only if the submodule does not contain input data.

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

Table 131 shows the parameters of the service.

Table 131 – Local Set Input

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	M	
Slot Number	M	
Subslot Number	M	
IOPS	M	
Subslot 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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the subslot number.

IOPS

This parameter shall contain the values according to the attribute in the Communication Relationship class (see 7.3.1.3.4).

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.

7.3.1.3.5.2 Local Set Input IOCS

This service shall be used by a consumer (IO controller) of inputs to pass the value of changed IOCS to the 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.

Table 132 shows the parameters of the service.

Table 132 – Local Set Input IOCS

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the 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.

7.3.1.3.5.3 Local 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 133 shows the parameters of the service.

Table 133 – Local Get Input

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOPS		M
Subslot 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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the 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.

Subslot 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.

7.3.1.3.5.4 Local 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 134 shows the parameters of the service.

Table 134 – Local Get Input IOCS

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the 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.

7.3.1.3.5.5 Local New Input

The New Input service signals to the application of the IO controller an update of the input buffer after reception of a valid APDU. Table 135 shows the parameters of the service.

Table 135 – Local New Input

Parameter name	ind
Argument	M
AREP	M
CREP	M
API	M
Slot Number	M
Subslot Number	M
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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the subslot number.

InData Flag

This parameter indicates that the received buffer is usable or not.

NOTE All consumer IOCRs of the AR have got data first time.

7.3.1.3.5.6 Local 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.

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.

Table 136 shows the parameters of the service.

Table 136 – Local Set Output

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	M	
Slot Number	M	
Subslot Number	M	
IOPS	M	
Subslot 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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the subslot number.

IOPS

This parameter shall contain the values according to the attribute in the Communication Relationship class (see 7.3.1.3.4).

Subslot 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.

7.3.1.3.5.7 Local Set Output IOCS

This service shall be used by a consumer (IO device) of outputs to pass the value of changed IOCS to the 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.

Table 137 shows the parameters of the service.

Table 137 – Local Set Output IOCS

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the 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.

7.3.1.3.5.8 Local 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 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. Table 138 shows the parameters of the service.

Table 138 – Local Get Output

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	M	
Slot Number	M	
Subslot Number	M	
Result(+)		S
AREP		M
IOPS		M
Subslot 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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the 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.

Subslot 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.

7.3.1.3.5.9 Local Get Output IOCS

This service shall be used by an IO device to read the IOCS from the application layer. This service shall only be used in conjunction with the IO AR or Supervisor AR. Table 139 shows the parameters of the service.

Table 139 – Local Get Output IOCS

Parameter name	req	cnf
Argument	M	
AREP	M	
CREP	M	
API	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.

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the 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.

7.3.1.3.5.10 Local New Output

The New Output service signals to the application of the IO device an update of the output buffer after reception of a valid APDU. Table 140 shows the parameters of the service.

Table 140 – Local New Output

Parameter name	ind
Argument	M
AREP	M
CREP	M
API	M
Slot Number	M
Subslot Number	M
InData Flag	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 indication.

CREP

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

API

This parameter contains the number of the application process.

Slot Number

This parameter contains the slot number.

Subslot Number

This parameter contains the subslot number.

InData Flag

This parameter indicates that the received buffer is usable or not.

NOTE All consumer IOCRs of the AR have got data first time.

7.3.1.3.5.11 Local Set Provider State

This service shall be used by the IO controller to set the “Provider State Flag” in the field APDU Data Status of the next conveyed APDU. The flag “Provider State Flag” influences all established ARs of the IO device. Table 141 shows the parameters of the service.

Table 141 – Local Set Provider State

Parameter name	req	cnf
Argument	M	
AREP	U	
Provider State 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.

Provider State Flag

This parameter indicates if the application process is running by the value „run“ or is stopped by the value „stop“. The validity of data is indicated by IOPS only and not by this flag.

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

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

7.3.1.3.5.12 Local Set Redundancy

This service shall be used by the IO device to set the “Redundancy Flag” of the selected AR. Table 142 shows the parameters of the service.

Table 142 – Local Set Redundancy

Parameter name	req	cnf
Argument	M	
AREP	U	
Redundancy 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.

Redundancy Flag

This parameter indicates with the value “primary present” that one AR within the AR set is in state “primary”. The value “no primary present” indicates that none of the AR within the AR set is seen as “primary”.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

7.3.1.3.5.13 Local Set State

This service shall be used by the IO device to set the “State Flag” of the selected AR. Table 143 shows the parameters of the service.

Table 143 – Local Set State

Parameter name	req	cnf
Argument	M	
AREP	U	
State 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.

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“.

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

7.3.1.3.5.14 Local Data State Changed

The service signals a changed of the APDU Data Status value to the application of the IO controller or IO device. This service shall only be conveyed if the APDU Data Status value has been changed by the IO device or the IO controller and shall provide only the changed flags. Table 144 shows the parameters of the service.

Table 144 – Local Data State Changed

Parameter name	ind
Argument	M
AREP	M
CREP	M
Data Valid Flag	C
State Flag	C
Redundancy 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.

The Data Valid 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.

State Flag

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

Redundancy Flag

This parameter indicates with the value “primary present” that one AR within the AR set is “primary”. The value “no primary present” indicates that none of the AR within the AR set is seen as “primary”.

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 1 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 2 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.

7.3.1.3.6 Behavior**7.3.1.3.6.1 Validity of process data****7.3.1.3.6.1.1 IOPS**

The **IOPS** (Input Output Provider Status) describes the validity of data source as being seen by the subscriber of the data.

It belongs to Output Data for an IO controller and to Input Data for an IO device. The IOPS is transmitted in the same IOCR as the submodule’s data itself (i.e. in the output CR for output data, in the input CR for input data).

NOTE 1 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 detecting instance shall set the IOPS accordingly and immediately. The IOPS is set by local means; e.g. on configuration errors, on submodule or module pulled, on insufficient parameterization state of the submodule (application ready pending), on diagnosis...

The IOPS may be set by individual instances participating when transmitting the input data: by the submodule (including the sensor), by the module, by the IO device, and by the IO controller (locally, e.g. data hold timeout). It allows therefore identifying the first instance which detected the invalid content, with the help of the following values:

- GOOD – data is valid
- BAD_BY_SUBSLOT – data is invalid as detected by the submodule
- BAD_BY_SLOT – data is invalid as detected by the module
- BAD_BY_DEVICE – data is invalid as detected by the IO device
- BAD_BY_CONTROLLER – data is invalid as detected by the IO controller

The IOPS is set to GOOD if the submodule’s data is valid.

The IOPS shall set to BAD_BY_XXX

- if the submodule is not available at the AR (pulled, locked by supervisor, superordinated locked, application ready pending, and wrong submodule).

- if the application considers the data as not valid (e.g. power failure at the submodule, ...).

NOTE 2 For submodules without data, the IOPS is expected to be GOOD (preferred solution), even if ApplicationReadyPending is set.

The IOPS of **output data** informs the device about the validity of the data coming from the IO controller (set by the IO controller).

If output data has been set to invalid using IOPS = BAD_BY_CONTROLLER, the device application shall not process this data; it shall apply a substitute value instead, to ensure that the output reaches a safe state in regard to the user's application (e.g. applying "0", a fixed substitute value...).

The default for the substitute value is defined by the device application; this default may be overridden (if provided by the IO device) by parameterization (e.g. holding the last valid value instead). IEC 61158-6-10 provides the standard parameterization data record "Substitute value for one subslot" (index 0x801E); a device application may however use a manufacturer specific parameterization record instead.

The IOPS of **input data** informs the IO controller about the validity of the data coming from the IO device (set by the IO device).

If input data has been set to invalid by the IO device by using IOPS = BAD_BY_SUBSLOT, BAD_BY_SLOT, or BAD_BY_DEVICE, the user application in the IO controller's host shall not process this data. The behavior here is locally defined in the IO controller (such as raising an access error, use local defined substitute values...).

7.3.1.3.6.1.2 IOCS

The **IOCS** (Input Output Consumer Status) indicates whether the application (controller or device) is processing the value.

It belongs to Input Data for an IO controller and to Output Data for an IO device. The IOCS is transmitted in the accompanying IOCR to the submodule's data (i.e. in the input CR for output data, in the output CR for input data).

The IOCS is set by individual instances participating when transmitting the output data: by the submodule (including the actor), by the module, by the IO device, and by the IO controller (for an input module locally, e.g. data hold timeout; for an output module e.g. due to being in operating state Stop).

The detecting instance shall set the IOCS accordingly and immediately. The IOCS is set by local means; e.g. on configuration errors, on submodule or module pulled, on insufficient parameterization state of the submodule (application ready pending), on diagnosis...

The IOCS allows identifying the first instance which does no longer process the content, with the help of the following values:

- GOOD – data is processed by the application
- BAD_BY_SUBSLOT – data is not being processed by the application as detected by the submodule
- BAD_BY_SLOT – data is not being processed by the application as detected by the module
- BAD_BY_DEVICE – data is not being processed by the application as detected by the device
- BAD_BY_CONTROLLER – not being processed by the application as detected by the submodule.

The IOCS is set to GOOD if the data is processed by the application.

The IOCS is set to BAD_BY_XXX

- if the submodule is not available for the AR (pulled, locked by supervisor, superordinated locked, application ready pending, wrong submodule).
- if the device application does not process the data (e.g. hand operation).

The IOCS of **input data** informs the IO device whether its IO controller processes this input data.

The IOCS of **output data** informs the IO controller where the IO device processes this output data.

7.3.1.3.6.1.3 Interdependencies between IOPS and corresponding IOCS

For an IO device applies: The IOCS shall be set independent of the corresponding IOPS; i.e. if the IOPS is set to BAD_BY_CONTROLLER but the IO device is still able to process the data, the IOCS shall be kept on GOOD.

For an IO controller applies: The IOCS may be set dependent or independent of the corresponding IOPS. The possible action of the IO controller is beyond the scope of this service definition.

NOTE The best practice pattern for the handling within the IO controller is: If IO device is not able to process the output data (indicated by setting the IOCS to "BAD_BY_SUBSLOT/SLOT/DEVICE"), the IO controller's application should set the corresponding IOPS to "BAD_BY_CONTROLLER". Thus allows the IO controller's application to apply initial values whenever the IO device is able to process the output data again; indicated by setting the IOCS to "GOOD" and sending a "return of submodule" alarm. During process of the alarm within the IO controller's application the initial value is applied to the output data and in consequence the IOPS is set to "GOOD" again.

The possible action of the IO device is beyond the scope of this service definition.

7.3.1.3.6.1.4 Behavior on Connect, Plug, Prm Begin and Application Ready

To simplify the further description BAD_BY_XXX is used to indicate BAD_BY_SUBSLOT, BAD_BY_SLOT, or BAD_BY_DEVICE.

If the AR Info responded in the Module Diff Block for a particular submodule states APPLICATION_READY_PENDING, the corresponding IOPS/IOCS shall be set to BAD_BY_XXX.

The IOPS/IOCS shall be set to GOOD earliest after finishing the submodule parameterization and if no further circumstances (e.g. application) hinder the validity of the data.

If the AR Info responded in the Module Diff Block for a particular submodule states SUPERORDINATED_LOCKED, LOCKED_BY_IO_CONTROLLER, or LOCKED_BY_IO_SUPERVISOR, the corresponding IOPS/IOCS shall be set to BAD_BY_XXX.

The IOPS/IOCS shall be set to GOOD earliest if the application relation gets the ownership and if no further circumstances (e.g. application) hinder the validity of the data.

If the Ident Info responded in the Module Diff Block for a particular submodule states WRONG or NO, the corresponding IOPS/IOCS shall be set to BAD_BY_XXX.

The IOPS/IOCS shall be set to GOOD earliest after plugging the right submodule, if the application relation gets the ownership, and if no further circumstances (e.g. application) hinder the validity of the data.

If the Module State responded in the Module Diff Block for a particular module states WRONG_MODULE or NO_MODULE, all corresponding IOPS/IOCS of that module's submodules shall be set to BAD_BY_XXX.

The IOPS/IOCS shall be set to GOOD earliest for a particular submodule after plugging the right module with that submodule, if the application relation gets the ownership, and if no further circumstances (e.g. application) hinder the validity of the data.

7.3.1.3.6.1.5 Behavior on transitions

If an IO device changes a particular IOPS/IOCS, it shall indicate a reason to the IO controller.

- On transition from GOOD to BAD_BY_XXX:
 - If the submodule's ownership changed to an IO supervisor or to the application (i.e. superordinated locked), the reason is indicated by a Controller by supervisor alarm.
 - If the submodule has been pulled, the reason is indicated by a Pull alarm.
 - If the application indicates the invalidity of the IO data, the reason is indicated e.g. by a coming diagnosis alarm, by setting ProviderState to "Stop"...
- On transition from BAD_BY_XXX to GOOD:
 - If the submodule's ownership changed back to the IO controller, the reason is indicated by Released alarm.
 - If the submodule has been plugged again, the reason is indicated by a Plug alarm.
 - If the application indicates the validity of the IO data, the reason is indicated e.g. by a going diagnosis alarm, by setting ProviderState to "Run"... Additionally a Return of Submodule alarm shall be indicated.

Note that having diagnosis information for a particular submodule does not imply that its IOPS or IOCS are set to BAD_BY_XXX.

7.3.1.3.6.1.6 Validity of channel data

The validity of submodule data is covered by the IOPS as described above. Since however submodules are sub-structured in channels, the validity of a channel is coverable by the IOPS only, if the submodule contains exactly one channel. If the submodule contains more channels, we need additional information.

A device application may handle the validity of a particular channel using the following approaches:

- by using **"safe" values** for a channel, if channel data is invalid (e.g. 0x7FFF for analogue data or FALSE for digital data)
- by providing **validity information** within the IO data describing the (in)validity of channel data.

The following well-known mechanisms to provide validity information are recommended:

- **qualifier**
A bit allows specifying BAD (0x0) or GOOD (0x1)
- **status**

Two bits allow specifying BAD (0x0), SIMULATION (0x1), UNCERTAIN (0x2), and GOOD (0x3)

– **process value status**

An octet allows defining multiple values to specify the validity; the interpretation of the octet is not specified by this specification

NOTE The following data types include validity information: OctetString2+Unsigned8 (see 5.3.1.6.4), Float32+Unsigned8 (see 5.3.1.6.5), Unsigned8+Unsigned8 (see 5.3.1.6.7), Unsigned16_S (see 5.3.1.6.8), Integer16_S (see 5.3.1.6.9), Unsigned8_S (see 5.3.1.6.10), and OctetString_S (see 5.3.2.2).

The validity information is transported along with the channel data using cyclic services (i.e. RTC). Channel data is a contiguous section of the IO data, the validity information related to this channel (if provided) is also part of the IO data (could be located adjacent to the data or at “some other place”).

If the IOPS is set to BAD_BY_XXX, all channel related validity information is BAD.

If channel data falls off or improves its quality, indicated by a change of the validity information, diagnosis information for this channel shall be added resp. removed. Therefore the information within the diagnosis ASE (i.e. the channel error...) indicates the reason for the change of the quality.

Since transporting data (cyclic, buffered) and alarms (acyclic, queued) is not synchronized, information related to presence or absence of diagnosis within the validity information may stand in conflict to the actual known diagnosis state of the channel. Thus, do not use such information to derive diagnosis information; but use the diagnosis information delivered on the standard way (diagnosis ASE, alarm ASE).

7.3.1.3.6.1.7 Validity of process data

The definitions above define how to determine the validity of process data.

Process data of a particular channel is valid

- if the IOPS of the submodule is GOOD,
- **and** (if provided) the validity information related to this channel is GOOD.

If status or process value status are provided as validity information, the validity information provides beyond GOOD and BAD more values (e.g. UNCERTAIN or SIMULATION). It is left to the programmer, whether to accept a value with this validity information.

7.3.1.3.6.2 Rules for communication relationships

All IOCRs within a particular application relationship shall have

- the same SendClockFactor,
- the same frame ID range (called IOCRProperties.RTClass in the Connect request),
- and the same DataHoldFactor.

The frame ID range shall match the ARTYPE:

- If ARTYPE is set to IOCARSsingle using RT_CLASS_3 (0x0010), IOCRProperties.RTClass shall be set to 0x03 (range 3).
- If ARTYPE is set to IOCARSsingle (0x0001), IOSAR (0x0006), or IOCARSR (0x0020), IOCRProperties.RTClass shall be set to 0x01 (range 7, RTC1), 0x02 (range 6, RTC1), or 0x04 (range 7, RTC_UDP).

NOTE Same SendClockFactor is enforced by IEC 61158-6-10; same frame ID range reduces complexity (meaning an AR is either RTC1 or RTC3); same DataHoldFactor reduces user complexity.

IOCRs may therefore differ in the parameters ReductionRatio, Phase, Sequence, FrameSendOffset and Content.

An application relationship using DFP has exactly one input and one output CR (see IEC 61158-6-10 IRInfoBlock request checking rules). DFP limits due to its subframe definition the input/output data size for a particular pack group member to 255.

There exists at least one input and one output CR.

The number of input CRs and output CRs may differ (e.g. there may be four input and two output CRs).

NOTE This is needed to support the mutual surveillance.

Each IOCR is scheduled and monitored independently according to its parameters (SendClockFactor, RTClass, ReductionRatio, Phase, Sequence, FrameSendOffset, and DataHoldFactor).

IN_DATA on the IO controller is reached, when

- ApplicationReady.req has been received,
- AND**
- Data has been received on each input CR

If the loss of a particular IOCR is detected by the consumer, the application relationship is released / aborted independent whether other IOCRs belonging to the same application relationship are still in data.

The effective DataHoldTime is determined by using the smallest reduction ratio of all consumers, multiplied with the SendClockFactor and with the DataHoldFactor.

7.3.1.3.6.3 Rules for padding

To spare bandwidth and IO device resources the IO controller shall pack the CRs; i.e. data, IOPS and IOCS shall be located without padding.

Paddings are enforced as follows:

- If the CR's data has less than 40 octets when using RT_CLASS_1 or RT_CLASS_3, or less than 42 octets when using RT_CLASS_UDP, the C_SDU is filled up with padding octets to reach the minimum Ethernet packet size.
- If a particular device (IO device or IO controller) expects subframes on word alignment, a subframe having an uneven data length is padded up with one octet to achieve word alignment.

7.3.1.4 Expected Identification class

7.3.1.4.1 Overview

7.3.1.4.1.1 General

The Expected Identification Class identifies the expected identification of the submodules as requested by the IO controller.

7.3.1.4.2 Relations

The services are bound to other services as shown in Table 145:

Table 145 – Binding expected identification services

Service	Binding	Usage
Read Expected Identification Data for one subslot	RPC::Read	Used by IO controller or IO supervisor
Read Expected Identification Data for one slot	RPC::Read	Used by IO controller or IO supervisor
Read Expected Identification Data for one AR	RPC::Read	Used by IO controller or IO supervisor

7.3.1.4.3 Template

This class specifies the object for the parameter data that are related to the IO devices itself and/or to the modules/submodules of the IO device.

The IO device context object is described by the following template:

ASE: AR ASE

CLASS: Expected Identification

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: List of Expected APIs
- 2.2.1 (m) Attribute: API
- 2.2.2 (m) Attribute: Slot Number
- 2.2.3 (m) Attribute: Module Ident Number
- 2.2.4 (m) Attribute: Module Properties
- 2.2.5 (m) Attribute: Module State
- 2.2.6 (m) Attribute: List of Expected Submodules
- 2.2.6.1 (m) Attribute: Subslot Number
- 2.2.6.4 (m) Attribute: Submodule Ident Number
- 2.2.6.5 (m) Attribute: Submodule Properties
- 2.2.6.5.1 (m) Attribute: Type
- 2.2.6.5.2 (m) Attribute: Shared Input
- 2.2.6.5.3 (m) Attribute: Reduce Input Submodule Data Length
- 2.2.6.5.4 (m) Attribute: Reduce Output Submodule Data Length
- 2.2.6.5.5 (m) Attribute: Discard IOXS
- 2.2.6.6 (m) Attribute: List of Data Descriptions
- 2.2.6.6.1 (m) Attribute: Data Direction
- 2.2.6.6.2 (m) Attribute: Submodule Data Length
- 2.2.6.6.3 (m) Attribute: Length IOPS
- 2.2.6.6.4 (m) Attribute: Length IOCS
- 2.2.6.7 (m) Attribute: Submodule State
- 2.2.6.7.1 (m) Attribute: Add Info
- 2.2.6.7.2 (m) Attribute: Qualifier Info
- 2.2.6.7.3 (m) Attribute: Maintenance Required
- 2.2.6.7.4 (m) Attribute: Maintenance Demanded
- 2.2.6.7.5 (m) Attribute: Fault
- 2.2.6.7.6 (m) Attribute: AR Info
- 2.2.6.7.7 (m) Attribute: Ident Info

SERVICES:

1	(m) OpsService:	Read Expected Identification Data for one subslot
2	(m) OpsService:	Read Expected Identification Data for one slot
3	(m) OpsService:	Read Expected Identification Data for one AR
4	(m) OpsService:	Read Module Diff Block for one AR

7.3.1.4.4 Attributes

Implicit

The key attribute Implicit indicates that the Expected Identification object is implicitly addressed by the services.

List of ARs

One AR is composed of the following list elements:

AREP

This attribute identifies the AR this Expected Identification belongs. It is a cross reference to the application relationship class.

List of Expected Submodules

This list contains the expected submodules as given by the IO controller / supervisor in the Connect request.

One Expected Submodule is composed of the following list elements:

API

This attribute defines the expected number of the application process the submodule is assigned to.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Slot Number

This attribute defines the expected slot the submodule is assigned to.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Module Ident Number

This attribute defines the expected identification of the module.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 1 The attribute value is manufacturer specific. See 7.1.4.5.4 for details.

Module Properties

This attribute is reserved for further use.

Attribute type: Unsigned16

Allowed values: see IEC 61158-6-10

Module State

This attribute defines the deviation between expected module and the module in the real identification.

Attribute type: Unsigned16

Allowed values: Shall be set according to Table 146.

Table 146 – Module State

Value	Use
NO_MODULE	The slot exists and the module is not plugged.
WRONG_MODULE	The module ident number is wrong. This coding should be avoided; use “Substitute” for the module and “Wrong (WR)” for each submodule
PROPER_MODULE	The slot exists and the module is okay.
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. This coding should be used in conjunction with “Wrong (WR)” for each submodule

List of Expected Submodules

This list contains the expected submodules as given by the IO controller / supervisor in the Connect request.

One Expected Submodule is composed of the following list elements:

Subslot Number

This attribute defines the expected subslot the submodule is assigned to.

Attribute type: Unsigned16

Allowed values: 0 to 0x9FFF

Submodule Ident Number

This attribute defines the expected identification of the submodule.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 2 The attribute value is manufacturer specific. See 7.1.4.5.4 for details.

Submodule Properties

This attribute consists of the following attributes.

Type

This attribute defines the type of the submodule.

Attribute type: Unsigned16

Allowed values: NO_IO, INPUT, OUTPUT, IO

Shared Input

This attribute defines whether to use the input submodule data as “shared input”. The value SHARED_INPUT shall only be used if the attribute Submodule type contains the value INPUT or IO.

Attribute type: Unsigned8

Allowed values: SHARED_INPUT, NOT_SHARED_INPUT

Reduce Input Submodule Data Length

This attribute defines whether to use the expected input submodule data length for the input CR or to reduce the submodule data length to zero.

The value ZERO shall only be used if the attribute Submodule type contains the value INPUT or IO. This flag may be used for the input CR, if the submodule is

configured as multicast provider, the IO controller is interested in diagnosis/availability of the submodule and does not need access to IO data.

Attribute type: Unsigned8

Allowed values: EXPECTED, ZERO

Reduce Output Submodule Data Length

This attribute defines whether to use the expected output submodule data length for the output CR or to reduce the submodule data length to zero.

The value ZERO shall only be used if the attribute Submodule type contains the value OUTPUT or IO. This flag shall be used for the output CR, if the submodule is configured as multicast consumer, the submodule data is delivered by some multicast provider (other than the IO controller) and the IO controller is interested only in diagnosis/availability of the submodule and does not set the IO data.

Attribute type: Unsigned8

Allowed values: EXPECTED, ZERO

Discard IOXS

This attribute defines whether to transfer the expected IOXS for this submodule in the corresponding CRs or not. The value TRUE shall only be used if the attribute Submodule type contains the value NO_IO.

Attribute type: Unsigned8

Allowed values: FALSE, TRUE

List of Data Descriptions

The following rules apply for the data descriptions.

There shall be one data description with Data Direction = DIRECTION_INPUT for a submodule of Type NO_IO or INPUT.

There shall be one data description with Data Direction = DIRECTION_OUTPUT for a submodule of Type OUTPUT.

There shall be a data descriptions with Data Direction = DIRECTION_INPUT and followed by a data description with Data Direction = DIRECTION_OUTPUT for a submodule of Type IO.

This attribute list consists of the following attributes.

Data Direction

The attribute defines the direction of the data description.

Attribute type: Unsigned16

Allowed values: DIRECTION_INPUT, DIRECTION_OUTPUT

Submodule Data Length

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

Submodule Data Length is independent of the value of Reduce Input/Output Submodule Data Length.

Attribute type: Unsigned16

Allowed values: 0 to 1 439

Length IOPS

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

Length IOPS is independent of the value of Discard IOXS.

Attribute type: Unsigned8

Allowed values: 1

Length IOCS

The attribute Length IOCS defines the number of octets of the IOCS. Length IOPS is independent of the value of Discard IOXS.

Attribute type: Unsigned8

Allowed values: 1

Submodule State

This attribute defines the deviation between expected submodule and the submodule in the real identification.

This attribute consists of the following attributes:

Add Info

This attribute contains additional information for the application relation.

For an application relation with AR Type IOCAR or IOCAR_SR the value is always set to NO_ADD_INFO.

For an application relation with AR Type IOSAR the value TAKEOVER_NOT_ALLOWED indicates that the submodule was configured with AR Properties "Supervisor Takeover allowed" set to "false" and thus the IO supervisor got no ownership for the submodule.

Allowed Values: NO_ADD_INFO, TAKEOVER_NOT_ALLOWED

Advice

This attribute indicates whether advice is present in at least one channel of the submodule.

Advice is derived from the Diagnosis ASE; see 7.3.4.2.6.7 for details.

Allowed Values: NO_ADVICE, ADVICE

NOTE 3 If Shared Input contains the value SHARED_INPUT, this field contains always the value NO_ADVICE (see 7.3.2.2.6.2.3.1 for details).

Maintenance Required

This attribute indicates whether maintenance required is present in at least one channel of the submodule.

Maintenance Required is derived from the Diagnosis ASE; see 7.3.4.2.6.5 for details.

Allowed Values: NO_MAINTENANCE_REQUIRED, MAINTENANCE_REQUIRED

NOTE 4 If Shared Input contains the value SHARED_INPUT, this field contains always the value NO_MAINTENANCE_REQUIRED (see 7.3.2.2.6.2.3.1 for details).

Maintenance Demanded

This attribute indicates whether maintenance demanded is present in at least one channel of the submodule.

Maintenance Demanded is derived from the Diagnosis ASE; see 7.3.4.2.6.6 for details.

Allowed Values: NO_MAINTENANCE_DEMANDED,
MAINTENANCE_DEMANDED

NOTE 5 If Shared Input contains the value SHARED_INPUT, this field contains always the value NO_MAINTENANCE_DEMANDED (see 7.3.2.2.6.2.3.1 for details).

Fault

This attribute indicates whether diagnosis information is present in at least one channel of the submodule.

Fault is derived from the Diagnosis ASE; see 7.3.4.2.6.2 for details.

Allowed Values: NO_FAULT, FAULT

NOTE 6 If Shared Input contains the value SHARED_INPUT, this field contains always the value NO_FAULT (see 7.3.2.2.6.2.3.1 for details).

AR Info

This attribute contains the submodule state in relation to the established AR.

AR Info is derived from the Ownership of the submodule; see 7.3.2.2.6.2.3 for details.

It shall be set according to Table 147.

Table 147 – AR Info

Value	Meaning
OWN	This AR is owner of the submodule
APPLICATION_READY_PENDING	This AR is owner of the submodule but the submodule is blocked by parameter checking pending; meaning that the execution of the parameterization is not ready yet. The submodule will issue a Return of Submodule alarm if parameterization succeeded, or a diagnosis alarm with channel error type "parameterization fault" of "parameter missing" if parameterization failed. Note that some devices may never end with parameter checking.
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 another IOAR
LOCKED_BY_IO_SUPERVISOR	This AR is not owner of the submodule. It is owned by another IOSAR

NOTE 7 If Shared Input contains the value SHARED_INPUT, this field contains always the value OWN (see 7.3.2.2.6.2.3.1 for details).

Ident Info

This attribute contains the submodule state in respect to expected and real identification for an established AR.

Ident Info is derived from the Real submodule; see 7.3.2.2.6.2.2 for details.

It shall be set according to Table 148.

Table 148 – Ident Info

Value	Meaning	Usable in conjunction with Module State "WRONG MODULE"
OK	No difference	No
SUBSTITUTE	Different but compatible	No
WRONG	Incompatible	Yes
NO	Empty	Yes

The field Ident Info contains the value OK, if the real submodule is the expected one.

The value SUBSTITUTE is returned, if the real submodule is able to substitute the expected one; in this case the data length of the real submodule may differ from the expected data length (e.g. a digital input with 8 bits as a substitute for a digital input with 8 bits). Note that the data format of the submodule shall be according to the expected identification also in the substitute case.

The value WRONG is returned, if a real submodule is plugged, but doesn't match or is able to substitute the expected one according to the type (ModuleID, SubmoduleID).

The value NO is returned, if the subslot is empty, i.e. no real submodule is plugged.

Special case: If Shared Input contains the value SHARED_INPUT, this field contains always the value OK (see 7.3.2.2.6.2.3.1 for details). Note that an IO controller using the submodule as shared input will not get pull or plug alarms; the availability of the submodule is indicated for him only via the IOPS.

7.3.1.4.5 Services

7.3.1.4.5.1 General Data Definition for Identification Data

All services to read the identification data (expected or real) have the same parameter layout. To ease the description the parameter is thus described once upfront to the services.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy of the record data object 7.3.1.6.6.1 applies.

The parameter Data has a service specific structure and meaning. Table 149 shows the specific elements of Data for this service.

Table 149 – General Data definition for identification services

Parameter name	service specific elements	rsp	cnf
Data	List of APIs	M	M(=)
	API	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(=)

Data

This parameter contains the values of the object which has been read.

List of APIs

This parameter is composed of the following list elements:

API

This parameter contains the value of the corresponding attribute of the Expected Identification class.

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 Expected Identification class.

Module Ident Number

This parameter contains the value corresponding attribute of the Expected Identification class.

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 Expected Identification class.

Submodule Ident Number

This parameter contains the value corresponding attribute of the Expected Identification class.

7.3.1.4.5.2 Read Expected Identification Data for one subslot

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context is needed for this service; see 7.3.1.6.6.4 for details.

7.3.1.4.5.3 Read Expected Identification Data for one slot

This service allows reading the expected identification data using a record data object with slot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context is needed for this service; see 7.3.1.6.6.4 for details.

7.3.1.4.5.4 Read Expected Identification Data for one AR

This service allows reading the expected identification data using a record data object with AR scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The

parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context is needed for this service; see 7.3.1.6.6.4 for details.

7.3.1.4.5.5 Read Module Diff Block for one AR

This service allows reading the actual state of the module diff block related to the expected configuration using a record data object with AR scope (see 7.3.1.6.1).

If no difference exists the Read Module Diff Block service shall respond no data with Length zero.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 150 for its structure.

The AR context is needed for this service; see 7.3.1.6.6.4 for details

Table 150 – Data elements of Read Module Diff Block

Parameter name	service specific elements	rsp	cnf
Data	Module Diff Block	C	C(=)
	List of APIs	M	M(=)
	API	M	M(=)
	List of Slots	M	M(=)
	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(=)

Data

This parameter contains the values of the object which has been read.

Module Diff Block

The parameter contains the Module Diff Block; see 7.3.1.4.6.2 for details.

7.3.1.4.6 Behavior

7.3.1.4.6.1 General

While the application relation is established to an IO controller or IO supervisor, changes in the real identification, ownership or diagnosis regarding the expected identification are indicated by the IO device using the appropriate alarm associated to the ownership (see 7.3.1.5.1.3) and diagnosis (see 7.3.1.5.1.2).

The services for accessing the real identification, the Module Diff Block and the diagnosis deliver up-to-date information.

7.3.1.4.6.2 Behavior of the Module Diff Block

The Module Diff Block contains submodules deviating from the expected identification. These are

- submodules not matching the expected identification (wrong, missing or substituted),
- submodules not owned by the application relation (superordinated locked, locked by IO controller or locked by supervisor),
- submodules which are not available due to e.g. parameter checking pending (Application Ready Pending),
- and submodules having diagnosis.

The Module Diff Block can be retrieved using the Read Module Diff Block for one AR service (see 7.3.1.4.5.5).

The Module Diff Block is part of Connect.rsp (see 7.3.1.2.5.1.2), Prm Begin.rsp (see 7.3.1.2.5.3) and Application Ready.req (see 7.3.1.2.5.5).

The content of the Module Diff Block depends on the context:

- Read Module Diff Block for one AR
The Module Diff Block is focused on submodules contained in the Expected Submodule Block of the application relationship's Connect.req.
- Connect.rsp / Application Ready.req sequence
The Module Diff Block is focused on submodules contained in the Expected Submodule Block of Connect.req.
- Prm Begin.rsp / Application Ready.req sequence
The Module Diff Block is focused on submodules contained in the Expected Submodule Block of the PrmBegin.req.
- Application Ready.req in the context of a plug/release sequence
The Module Diff Block is focused on the plugged/released submodule.

An empty Module Diff Block (meaning everything is perfect) shall be omitted.

The IO controller/supervisor may use the Module Diff Block returned by the services Connect and Prm Begin to optimize which submodules out of the expected identification to parameterize. These are those submodules, which match the expected identification (SubmoduleState.IdentInfo is OK or SUBSTITUTE) and where ownership was granted (SubmoduleState.ARInfo == OWN).

To allow an IO controller/supervisor to fulfill the startup parameterization correctly,

- any improvement of the ownership from “not granted” (i.e. SubmoduleState.ARInfo is SUPERORDINATED_LOCKED, LOCKED_BY_IO_CONTROLLER, or LOCKED_BY_IO_SUPERVISOR) to “granted” (i.e. SubmoduleState.ARInfo is OWN or APPLICATION_READY_PENDING),
- and any improvement of matching the expected identification from “not matching” (i.e. SubmoduleState.IdentInfo is WRONG or NO) to “matching” (i.e. SubmoduleState.IdentInfo is OK or SUBSTITUTE) in the phase between Connect/Prm Begin and Application Ready

shall be indicated by the IO device after the Application Ready.rsp using the appropriate alarm (i.e. “Release” or “Plug”).

Any degradation of ownership from “granted” to “not granted” or expected identification from OK or SUBSTITUTE to WRONG or MISSING is possible even during the startup parameterization phase.

In this case appropriate errors (e.g. “access / invalid subslot”) will be returned when writing the parameterization records. The Module Diff Block delivered with the Application Ready will contain the degraded status of the submodule.

The IO controller/supervisor uses the Module Diff Block delivered with the service Application Ready.req (see 7.3.1.2.5.5) to determine initial state of the expected identification regarding the availability of the submodule for its application (SubmoduleState.IdentInfo is OK or SUBSTITUTE and SubmoduleState.ARInfo == OWN) and the initial state for the diagnosis of the submodule.

Any further change of the submodules contained in the expected identification is indicated by the IO device using alarms (“Controlled By Supervisor”, “Released”, “Pull”, “Plug”, “Plug Wrong Submodule”, and all kinds of diagnosis related alarms). The alarms are issued earliest after the Application Ready.rsp. Note that the device shall indicate all changes after sending the Module Diff Block with the Application Ready.req; meaning it shall hold alarm requests locally in the alarm queue until the Application Ready.rsp was received.

Special case “shared input”: Since no alarms are allowed on an association beside the Return Of Submodule alarm, an associate has only information about the availability of the input data. Other information like the submodule’s presence (pull, plug, plug wrong), its parameterization state (application ready pending) or its diagnosis of the submodule is not available, since it will not be actualized via some alarm. Thus the Submodule State shall contain in any Module Diff Block for a shared input submodule

- for the Ident Info (indicating its presence) the value OK,
- for Advice (indicating its advice information) the value NO_ADVICE,
- for Maintenance Required (indicating its maintenance required information) the value NO_MAINTENANCE_REQUIRED,
- for Maintenance Demanded (indicating its maintenance demanded information) the value NO_MAINTENANCE_DEMANDED,
- for Fault (indicating its diagnosis information) the value NO_FAULT,
- for ARInfo (indicating ownership problems) the value OWN,

independent of the submodule’s state. In other words this information is only available for the application having the ownership on the submodule. The rules for setting the IOxS to BAD e.g. in case of a pulled or a wrongly plugged submodule (see 7.3.1.3.5) will take care that the submodule’s input data is not useable in such a case.

7.3.1.4.6.3 Behavior of the Module Diff Block regarding diagnosis

If for a particular submodule the SubmoduleState.ARInfo is set to application ready pending the submodule needs further time to finish the parameterization. Depending on the application this may take some time (e.g. compressed air brake, to fill the compressed air reservoir). The IOxS of the submodule is set to BAD in this case. If the submodule afterwards finishes the parameterization without error, it will set its IOxS to GOOD and issue a return of submodule alarm. If the submodule was not able to finish the parameterization without error, it will issue a diagnosis alarm indicating the error. Two standard ChannelErrorTypes are provided for that, ParameterMissing (0x000F) and ParameterizationFault (0x0010).

If the parameterization failed before issuing the ApplicationReady.req, the error is reported using the SubmoduleState.Fault.

The initial diagnosis state for the submodule is delivered with the Module Diff Block contained in the ApplicationReady.req to the IO controller. For this the Module Diff Block contains SubmoduleState.Advice, .MaintenanceRequired, .MaintenanceDemanded and .Fault.

Set SubmoduleState.Advice to true, if at least one diagnosis item exists on this submodule with a severity in the range from Qualifier_3 to Qualifier_6, or MaintenanceRequired.

Set SubmoduleState.MaintenanceRequired to true, if at least one diagnosis item exists on this submodule with a severity in the range from Qualifier_7 to Qualifier_16, or MaintenanceRequired.

Set SubmoduleState.MaintenanceDemanded to true, if at least one diagnosis item exists on this submodule with a severity in the range from Qualifier_17 to Qualifier_26, or MaintenanceDemanded.

Set SubmoduleState.Fault to true, if at least one diagnosis item exists on this submodule with a severity in the range from Qualifier_27 to Qualifier_31, or Fault.

7.3.1.5 Alarm class

7.3.1.5.1 Overview

7.3.1.5.1.1 General

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

For the transmission of an alarm the following conditions have to be fulfilled:

- the source submodule shall be existent (no empty subslot and no wrong submodule)
- a particular application relationship has the ownership of the submodule issuing the alarm; in this case all alarms are transported to the IO controller.
- a particular application relationship is associated to the submodule issuing the alarm; in this case the return of submodule alarm is transported to the IO controller.

An alarm shall always be issued; the sink shall respond with a negative acknowledgment “not supported” in case no handler for the alarm type is installed.

Alarms are used for several purposes: pull, plug, status, update, etc. and especially diagnosis.

Table 151 gives an overview over the defined alarm types, their usage and whether the support of the alarm is mandatory or optional.

The alarm types indicated as “attached to Diagnosis ASE” (see 7.3.4) are issued, whenever entries are added or removed from the Diagnosis ASE. Those alarms thus allow maintaining the actual diagnosis state of an IO device. The services of the diagnosis ASE allow getting the current state of the diagnosis.

The columns “alarm payload” define the items being part of the alarm payload of a particular alarm.

Table 151 – Alarm type

Alarm type	Usage	Use	Attached to Diagnosis ASE	Alarm Payload				
				Maintenance Item	Diagnosis Item	Common Profile Item	Manufacturer Alarm Item	Profile Alarm Item
Diagnosis	Indicate appearing or disappearing diagnosis in standard format for one or more channels. Indicates appearing diagnosis in USI format.	Mandatory	Yes	a	b	—	—	—
Diagnosis Disappears	Indicates all diagnosis disappears for the submodule, when having no Alarm Item (optimization). Indicates disappearing diagnosis in USI format. Indicates disappearing diagnosis information in standard format; not recommended, use diagnosis instead.	Mandatory	Yes	a	b	—	—	—
Media Redundancy mismatch – ring	Indicates a change in the media redundancy status for ring.	Mandatory when supporting media redundancy ring	Yes	a	b	—	—	—
Media Redundancy mismatch – interconnection	Indicates a change in the media redundancy status for interconnection.	Mandatory when supporting media redundancy interconnection	Yes	a	b	—	—	—
Multicast Communication Mismatch Notification	Indicates a multicast CR problem	Mandatory when supporting multicast	Yes	a	b	—	—	—
Port Data Change Notification	Indicates a change of port data e.g. link up or down	Mandatory	Yes	a	b	—	—	—
Sync Data Change Notification	Indicates a change of clock synchronization	Mandatory when supporting clock synchronization	Yes	a	b	—	—	—
Isochronous Mode Problem Notification	Indicates a problem of isochronous applications, e.g. started to late	Mandatory when supporting isochronous mode	Yes	a	b	—	—	—
Network Component Problem Notification	Indicates a problem of network components, e.g. frame dropped no resources	Mandatory when supporting network component diagnosis	Yes	a	b	—	—	—
Time Data Change Notification	Indicates a change of time synchronization	Mandatory when supporting time synchronization	Yes	a	b	—	—	—
Dynamic Frame Packing problem Notification	Indicates a problem during the dynamic frame packing operation	Mandatory when supporting dynamic frame packing	Yes	a	b	—	—	—
MRPD problem Notification	Indicates a problem during MRPD operation	Optional when supporting MRPD	Yes	a	b	—	—	—

Alarm type	Usage	Use	Attached to Diagnosis ASE	Alarm Payload				
				Maintenance Item	Diagnosis Item	Common Profile Item	Manufacturer Alarm Item	Profile Alarm Item
Multiple Interface Mismatch Notification	Indicates an inconsistent setting of parameters affecting more than one interface	Mandatory	Yes	a	b	—	—	—
Pull	Indicates pull of a requested module/submodule	Mandatory	No	—	—	—	c	—
Pull module	Indicates pull of a requested module	Mandatory when supporting the pull module alarm	No	—	—	—	c	—
Plug	Indicates a plug of a requested submodule	Mandatory	No	—	—	—	c	—
Plug Wrong Submodule	Indicates a plug of a wrong submodule	Mandatory	No	—	—	—	c	—
Controlled by supervisor	Indicates that a requested submodule is controlled by an IO supervisor Logical “pull” of a submodule to withdraw ownership	Mandatory	No	—	—	—	c	—
Released	Indicates that a requested submodule has been released by an IO supervisor, IO controller or by IO device local means Logical “plug” of a submodule to return ownership or trigger a reparameterization	Mandatory	No	—	—	—	c	—
Return Of Submodule	Indicates a change of IOCS/IOPS from BAD to GOOD for a requested submodule	Mandatory	No	—	—	—	c	—
Process	Indicates an alarm from the controlled process (usage application and process specific)	Optional	No	—	—	d	—	—
Status	Indicates a status change within a requested submodule	Optional	No	—	—	d	—	—
Update	Indicates a change of parameter for a requested submodule	Optional	No	—	—	d	—	—
Upload and Retrieval Notification	Indicates a request for upload and storage of Record Data objects	Optional	No	—	—	e	—	—
Manufacturer Specific	Values from 0x0020 to 0x007F may be used for manufacturer specific alarm types	Optional	No	—	—	—	f	—
Profile Specific	Shall be used according Profile Guidelines if an IO device complies to a specific device profile	According to supported profile	No	—	—	—	—	g

a The maintenance item is conveyed, if the diagnosis item contains at least one channel diagnosis with ChannelProperties.Maintenance set to MAINTENANCE_REQUIRED or MAINTENANCE_DEMANDED.

b The diagnosis item can be omitted if all diagnosis disappear from the channel.

c The alarm payload may contain manufacturer specific information; however this should not be used.

d The alarm payload may contain common profile specific information or else manufacturer specific information.

e The alarm payload may contain common profile specific information.

f The alarm payload may contain manufacturer specific information.

g The alarm payload may contain profile specific information.

The priority ALARM_HIGH shall be used in the context of Alarm Type “Process” (see Table 151); all other Alarm Types shall use ALARM_LOW as not to interfere with application events.

7.3.1.5.1.2 Alarm types attached to diagnosis ASE

The alarm types stated in Table 152 are exclusively attached to the diagnosis ASE. The alarms are triggered by the diagnosis ASE itself.

Table 152 – Alarm types attached to diagnosis ASE

Alarm type
Diagnosis
Media Redundancy Mismatch – ring
Media Redundancy Mismatch – interconnection
Diagnosis Disappears
Multicast Communication Mismatch Notification
Port Data Change Notification
Sync Data Change Notification
Isochronous Mode Problem Notification
Network Component Problem Notification
Time Data Change Notification
Dynamic Frame Packing problem Notification
MRPD problem Notification
Multiple Interface Mismatch Notification

Alarm types attached to the diagnosis ASE have a relation to channels as stated in the alarm payload.

7.3.1.5.1.3 Alarm types attached to ownership

The alarm types stated in Table 153 are exclusively attached to the OWNSM, ASSSM, and the RSMSM. The alarms are triggered by the state machines itself.

Table 153 – Alarm types attached to ownership

Alarm type
Pull
Pull module
Plug
Plug Wrong Submodule
Controlled by supervisor
Released
Return Of Submodule

Alarm types attached to ownership have no relation to channels; they always state the “whole” submodule in the alarm payload.

7.3.1.5.1.4 Alarm types attached to common profiles, profiles, and application

The alarm types stated in Table 154 are attached to common profiles, profiles, and application.

Table 154 – Alarm types attached to common profiles, profiles, and application

Alarm type
Process
Status
Update
Upload and Retrieval Notification
Manufacturer Specific

The alarm types may have a relation to channels.

The alarm payload may contain optionally common profile specific information:

- for alarm type “Upload and Retrieval alarm” an upload & retrieval item or a iParameter item,
- for alarm type “Status” a PE alarm item or an RS alarm item,
- for alarm type “Process” a process alarm item.

7.3.1.5.2 Relations

The services are bound to other services as shown in Table 155:

Table 155 – Binding Alarm services

Service	Binding	Usage
Alarm Notification	RTA::APMS A Data	Used by IO device

7.3.1.5.3 Template

An Alarm object is described by the following template:

ASE: AR ASE

CLASS: Alarm

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

1	(m) Key Attribute: Implicit
2	(m) Attribute: List of Alarm CRs
2.1	(m) Attribute: AREP
2.2	(m) Attribute: CREP
2.3	(m) Attribute: LT Field
2.4	(m) Attribute: Alarm CR Properties
2.4.1	(m) Attribute: Priority
2.4.2	(m) Attribute: Transport
2.5	(m) Attribute: RTA Timeout Factor
2.6	(m) Attribute: RTA Retries
2.7	(m) Attribute: Local Alarm Reference
2.8	(m) Attribute: Remote Alarm Reference

2.9	(m) Attribute:	Max Alarm Data Length
2.10	(m) Attribute:	Alarm CR Tag Header High
2.10.1	(m) Attribute:	Vlan ID
2.10.2	(m) Attribute:	Alarm User Priority
2.11	(m) Attribute:	Alarm CR Tag Header Low
2.11.1	(m) Attribute:	Vlan ID
2.11.2	(m) Attribute:	Alarm User Priority
2.12	(m) Attribute:	Alarm Priority High Sequence Number
2.13	(m) Attribute:	Alarm Priority Low Sequence Number

SERVICES:

1 (m) OpsService: Alarm Notification

7.3.1.5.4 Attributes**Implicit**

The key attribute Implicit indicates that the Alarm object is implicitly addressed by the services.

List of Alarm CRs

This attribute is composed of the following list elements:

AREP

This attribute identifies the AR this CR belongs. It is a cross reference to the application relationship class.

CREP

This attribute contains the local identifier of this CR. The attribute is used to uniquely identify a particular CR. It is a cross reference to the communication relationship class.

LT Field

This attribute contains the length type field of the Physical Layer Frame (see 7.3.1.3.4).

Alarm CR Properties

This attribute consists of the following attributes:

Attribute type: Unsigned16

Priority

This attribute defines whether the Alarm class uses one or two alarm queues.

An IO device shall always support two alarm queues. The IO controller as alarm sink may choose whether it limits itself to one resource or not.

Allowed values: USER_PRIORITY - the alarm priority as given by the user is used and Alarm notifications are enqueued in the alarm high or alarm low queue accordingly, FIXED_PRIORITY - user priority is ignored and Alarm notifications are always enqueued in the alarm low queue accordingly only the alarm low queue is used.

Transport

This attribute contains the transport layer (layer 2 resp. UDP) of the alarm.

RTA_CLASS_1 means, IEEE 802.3 and IEEE 802.1Q are applicable without any extension.

RTA_CLASS_UDP means, IEEE 802.3, IEEE 802.1Q, and IP-Suite (see 6.3.15) are applicable without any extension.

Attribute type: Unsigned8

Allowed values: RTA_CLASS_1, RTA_CLASS_UDP

RTA Timeout Factor

This attribute indicates the timeout value of the alarm source.

Attribute type: Unsigned16

Time Base: 100 ms

Allowed Values: 1 up to 100 mandatory, 101 to $2^{16}-1$ optional

Default value: 100 ms

RTA Retries

This attribute contains the number of retries.

Attribute type: Unsigned16

Allowed Values: 3 to 15.

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 maximum length of alarm data accepted by the alarm sink (IO controller, and IO supervisor).

Attribute type: Unsigned16

Allowed value: 200 to 1 432

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 CR 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

Alarm Priority High Sequence Number

This attribute stores the sequence number of the last Alarm notification being send to the alarm sink with alarm priority high. It is initialized with an arbitrary value and shall be incremented with every Alarm notification.

Whenever an Alarm is repeated over different ARs of an ARSet the same Sequence Number shall be used.

Attribute type: Unsigned16

Allowed values: 0x0000 to 0x7FFF

Alarm Priority Low Sequence Number

This attribute stores the sequence number of the last Alarm notification being sent to the alarm sink with alarm priority low. It is initialized with an arbitrary value and shall be incremented with every Alarm notification.

Whenever an Alarm is repeated over different ARs of an ARSet the same Sequence Number shall be used.

Attribute type: Unsigned16

Allowed values: 0x0000 to 0x7FFF

7.3.1.5.5 Services**7.3.1.5.5.1 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 156 shows the parameters of the service.

Table 156 – Alarm Notification

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
AREP	M	M(=)		
Alarm Priority	M	M(=)		
Alarm type	M	M(=)		
API	M	M(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Module Ident Number	M	M(=)		
Submodule Ident Number	M	M(=)		
Alarm Specifier	M	M(=)		
Sequence Number	M	M(=)		
Channel Diagnosis	U	U(=)		
Manufacturer Specific Diagnosis	U	U(=)		
Submodule Diagnosis State	U	U(=)		
AR Diagnosis State	U	U(=)		
Alarm Payload	U	U(=)		
Maintenance Item	U	U(=)		
User Structure Identifier	M	M(=)		
Maintenance Status	M	M(=)		
Alarm Item	M	M(=)		
User Structure Identifier	M	M(=)		
Diagnosis Item	C	C(=)		
Channel Diagnosis	C	C(=)		
List of Channel Diagnosis Data	M	M(=)		
Channel Number	M	M(=)		
Channel Properties	M	M(=)		
Channel Error type	M	M(=)		
Ext Channel Diagnosis	C	C(=)		
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(=)		
Qualified Channel Diagnosis	C	C(=)		
List of Qualified 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(=)		
Qualified Channel Qualifier	M	M(=)		
Manufacturer Specific Diagnosis	C	C(=)		
List of Data	M	M(=)		

Parameter name	req	ind	rsp	cnf
Data	M	M(=)		
Upload & Retrieval Item	C	C(=)		
List of Records	M	M(=)		
Record Number	M	M(=)		
Record Length	M	M(=)		
iParameter Item	C	C(=)		
iPar Req Header	M	M(=)		
Max Segm Size	M	M(=)		
Transfer Index	M	M(=)		
Total iPar Size	M	M(=)		
Manufacturer Alarm Item	C	C(=)		
List of Data	M	M(=)		
Data	M	M(=)		
Profile Alarm Item	C	C(=)		
List of Data	M	M(=)		
Data	M	M(=)		
Result(+)			S	S
AREP			M	M
Alarm type			M	M
API			M	M
Slot Number			M	M
Subslot Number			M	M
Alarm Specifier			M	M
PNIO Status			M	M
Result(-)			S	S
AREP			M	M
PNIO Status			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.

Alarm Priority

This parameter contains the priority of the alarm.

Allowed Values: ALARM_HIGH, ALARM_LOW

Alarm type

Attribute type: Unsigned16

Allowed values: see Table 151

API

This parameter defines the API the submodule issuing the alarm belongs to.

Slot Number

This parameter defines the slot number as a local identifier. It shall contain the physical or logical slot which issues the alarm.

Subslot Number

This parameter defines the subslot number as a local identifier. It shall contain the physical or logical subslot which issues the alarm.

Module Ident Number

This parameter contains the module identification.

Submodule Ident Number

This parameter contains the submodule identification.

Alarm Specifier

This parameter consists of the following parameters.

Sequence Number

This parameter defines the sequence number of the alarm. By means of the Sequence Number the receiver detects duplications and reflects the value of the field in the Alarm.

Allowed values: 0 to 2 047

Channel Diagnosis

This parameter shall only be used for Alarm types attached to the Diagnosis ASE (see Table 151). For all other Alarm Types this parameter is not valid.

Table 157 shows the allowed values.

Table 157 – Channel Diagnosis

Value	Meaning
NO_CHANNEL_DIAG	Means that no diagnosis item in standard format exists on this submodule.
CHANNEL_DIAG	Means that diagnosis in standard format exists on this submodule.

Manufacturer Specific Diagnosis

This parameter shall only be used for Alarm types attached to the Diagnosis ASE (see Table 151). For all other Alarm Types this parameter is not valid.

Table 158 shows the allowed values.

Table 158 – Manufacturer Specific Diagnosis

Value	Meaning
NO_MAN_DIAG	Means that no diagnosis item in USI format exists on this submodule.
MAN_DIAG	Means that diagnosis in USI format exists on this submodule.

Submodule Diagnosis State

This parameter shall only be used for Alarm types attached to the Diagnosis ASE (see Table 151). For all other Alarm Types this parameter is not valid.

Table 159 shows the allowed values.

Table 159 – Submodule Diagnosis State

Value	Meaning
NO_FAULT	Fault free No diagnosis item exists in standard or USI format on this submodule with a severity in the range from Qualifier_27 to Qualifier_31, or Fault
FAULT	At least one diagnosis item exists in standard or USI format on this submodule with a severity in the range from Qualifier_27 to Qualifier_31, or Fault

AR Diagnosis State

This parameter shall only be used for Alarm types attached to the Diagnosis ASE (see Table 151). For all other Alarm Types this parameter is not valid.

The AR Diagnosis State shall be corresponding to the current state of the Communication Relationship class parameter Problem Indicator Flag, which is related to all modules/submodules that belong to the AR.

Table 160 shows the allowed values.

Table 160 – AR Diagnosis State

Value	Meaning
NO_FAULT	Fault free No diagnosis item exists in standard or USI format in this AR with a severity in the range from Qualifier_27 to Qualifier_31, or Fault.
FAULT	At least one diagnosis item exists in standard or USI format in this AR with a severity in the range from Qualifier_27 to Qualifier_31, or Fault

Alarm Payload

This optional parameter contains the alarm payload; structured according to the rules shown in Table 151. It consists of the following parameters:

Maintenance Item

This conditional parameter contains the maintenance status. It is provided if at least one diagnosis item exists in standard format on this submodule with a severity in the range Qualifier_3 to Qualifier_26, MaintenanceRequired, or MaintenanceDemanded. Otherwise the MaintenanceItem shall be omitted.

It consists of the following parameters:

User Structure Identifier

This parameter identifies the format of the item.

Attribute type: Unsigned16

Allowed values: MAINTENANCE_USI

Maintenance Status

This conditional parameter contains the maintenance status of the submodule.

It consists of the following parameters:

Maintenance Required

This parameter contains the value MAINTENANCE_REQUIRED if at least one diagnosis item exists with a severity in the range from Qualifier_7 to Qualifier_16, or MaintenanceRequired on this submodule.

Allowed values: MAINTENANCE_REQUIRED, NO_MAINTENANCE_REQUIRED

Maintenance Demanded

This parameter contains the value MAINTENANCE_DEMANDED if at least one diagnosis item exists with a severity in the range from Qualifier_17 to Qualifier_26, or MaintenanceDemanded on this submodule.

Allowed values: MAINTENANCE_DEMANDED,
NO_MAINTENANCE_DEMANDED

Qualifier 3

This parameter contains the value QUALIFIER_3_SET if at least one diagnosis item exists with a severity Qualifier_3 on this submodule.

Allowed Values: QUALIFIER_3_SET, QUALIFIER_3_NOT_SET

Qualifier 31

This parameter contains the value QUALIFIER_31_SET if at least one diagnosis item exists with a severity Qualifier_31 on this submodule.

Allowed Values: QUALIFIER_31_SET, QUALIFIER_31_NOT_SET

Alarm Item

This optional parameter contains the accompanying information for the alarm. It is present, if further information is conveyed. The rules which item (i.e. diagnosis item, upload and retrieval item, iParameter Item, or common alarm item) is conveyed are described in Table 151.

This parameter is composed of the following parameters.

User Structure Identifier

This parameter identifies the format of the item.

Attribute type: Unsigned16

Allowed values: shown in Table 161

Table 161 – User Structure Identifier

Value	Meaning / Usage
CHANNEL_DIAGNOSIS_USI	Shall only be used in conjunction with Channel Diagnosis in Alarm Notification and Diagnosis Data.
EXT_CHANNEL_DIAGNOSIS_USI	Shall only be used in conjunction with Extended Channel Diagnosis in Alarm Notification and Diagnosis Data.
QUALIFIED_CHANNEL_DIAGNOSIS_USI	Shall only be used in conjunction with Qualified Channel Diagnosis in Alarm Notification and Diagnosis Data.
MANUFACTURER_SPECIFIC_USI_0x0000 up to MANUFACTURER_SPECIFIC_USI_0x7FFF	Shall be used in conjunction with diagnosis in USI format in Alarm Notification and Diagnosis Data.
PROFILE_SPECIFIC_USI_0x9000 up to PROFILE_SPECIFIC_USI_0x9FFF	The meaning is defined in profile specifications.
UPLOAD_AND_RETRIEVAL_USI	Shall be used in conjunction with Upload and Retrieval notification.
IPARAMETER_USI	Shall be used in conjunction with iParameter notification.

Diagnosis Item

This optional parameter is present only, if the User Structure Identifier is set to CHANNEL_DIAGNOSIS_USI, EXT_CHANNEL_DIAGNOSIS_USI, QUALIFIED_CHANNEL_DIAGNOSIS_USI, MANUFACTURER_SPECIFIC_USI_0x0000 up to MANUFACTURER_SPECIFIC_USI_0x7FFF.

A list element consists of the following parameters:

Channel Diagnosis

This optional parameter is present if the User Structure Identifier is set to CHANNEL_DIAGNOSIS_USI. It consists of the following parameters:

List of Channel Diagnosis Data

This parameter contains a list of channel diagnosis data. A list element consists of the following parameters:

Channel Number

This parameter defines the channel number as a local identifier (see 7.3.4). 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 to 0x7FFF user specific channel number, 0x8000 submodule

Channel Properties

This parameter consists of the following parameters:

Type

This parameter has the same definition as specified in the Diagnosis ASE.

Accumulative

This parameter has the same definition as specified in the Diagnosis ASE.

Maintenance

This parameter has the same definition as specified in the Diagnosis ASE.

Specifier

This parameter indicates that diagnosis/maintenance appears or disappears. The semantic is shown in Table 162.

Allowed values: APPEARS, DISAPPEARS, DISAPPEARS_BUT_OTHER_REMAIN, ALL_DISAPPEARS

Table 162 – Semantics of Specifier

Maintenance	Specifier	Meaning
FAULT	APPEARS	Diagnosis appears The diagnosis ASE contains this and possible other entries with ChannelProperties.Maintenance = FAULT for this channel.
FAULT	DISAPPEARS	Diagnosis disappears The diagnosis ASE contains no further entry with ChannelProperties.Maintenance = FAULT for this channel.
FAULT	DISAPPEARS_BUT_OTHER_REMAIN	Diagnosis disappears The diagnosis ASE still contains other entries with ChannelProperties.Maintenance = FAULT for this channel.
FAULT	ALL_DISAPPEARS	Diagnosis disappears The diagnosis ASE contains no longer any entries for this channel, all subsequent ^a Diagnosis, Maintenance (required and demanded) and Qualified disappears.
MAINTENANCE_REQUIRED	APPEARS	Maintenance Required appears The diagnosis ASE contains this and possible other entries with ChannelProperties.Maintenance = MAINTENANCE_REQUIRED for this channel.
MAINTENANCE_REQUIRED	DISAPPEARS	Maintenance Required disappears The diagnosis ASE contains no further entry with ChannelProperties.Maintenance = MAINTENANCE_REQUIRED for this channel.
MAINTENANCE_REQUIRED	DISAPPEARS_BUT_OTHER_REMAIN	Maintenance Required disappears The diagnosis ASE still contains other entries with ChannelProperties.Maintenance = MAINTENANCE_REQUIRED for this channel.
MAINTENANCE_DEMANDED	APPEARS	Maintenance Demanded appears The diagnosis ASE contains this and possible other entries with ChannelProperties.Maintenance = MAINTENANCE_DEMANDED for this channel.
MAINTENANCE_DEMANDED	DISAPPEARS	Maintenance Demanded disappears The diagnosis ASE contains no further entry with ChannelProperties.Maintenance = MAINTENANCE_DEMANDED for this channel.
MAINTENANCE_DEMANDED	DISAPPEARS_BUT_OTHER_REMAIN	Maintenance Demanded disappears The diagnosis ASE still contains other entries with ChannelProperties.Maintenance = MAINTENANCE_DEMANDED for this channel.

Maintenance	Specifier	Meaning
QUALIFIED_DIAGNOSIS	APPEARS	Qualified Diagnosis appears The diagnosis ASE contains this and possible other entries with ChannelProperties.Maintenance = QUALIFIED_DIAGNOSIS for this channel.
QUALIFIED_DIAGNOSIS	DISAPPEARS	Qualified Diagnosis disappears The diagnosis ASE contains no further entry with ChannelProperties.Maintenance = QUALIFIED_DIAGNOSIS for this channel.
QUALIFIED_DIAGNOSIS	DISAPPEARS_BUT_OTHER_REMAIN	Qualified Diagnosis disappears The diagnosis ASE still contains other entries with ChannelProperties.Maintenance = QUALIFIED_DIAGNOSIS for this channel.
^a Subsequent means, that for this particular ChannelErrorType all ExtChannelErrorTypes (and the ChannelErrorType itself) with diagnosis, maintenance required, maintenance demanded, and qualified changes to OK.		

Direction

This parameter has the same definition as specified in the Diagnosis ASE.

Channel Error type

This parameter has the same definition as specified in the Diagnosis ASE.

Ext Channel Diagnosis

This optional parameter is present if the User Structure Identifier is set to EXT_CHANNEL_DIAGNOSIS_USI. It consists of the following parameters:

List of Ext Channel Diagnostic Data

This parameter contains a list of extended channel diagnosis data. A list element consists of the following parameters:

Channel Number

This parameter has the same definition as specified in the Diagnosis ASE.

Channel Properties

This parameter has the same definition as specified in the Channel Diagnosis above.

Channel Error type

This parameter has the same definition as specified in the Diagnosis ASE.

Ext Channel Error type

This parameter has the same definition as specified in the Diagnosis ASE.

Ext Channel Add Value

This parameter has the same definition as specified in the Diagnosis ASE.

Qualified Channel Diagnosis

This optional parameter is present if the User Structure Identifier is set to QUALIFIED_CHANNEL_DIAGNOSIS_USI. It consists of the following parameters:

List of Qualified Channel Diagnostic Data

This parameter contains a list of extended channel diagnosis data. A list element consists of the following parameters:

Channel Number

This parameter has the same definition as specified in the Diagnosis ASE.

Channel Properties

This parameter has the same definition as specified in the Channel Diagnosis above.

Channel Error type

This parameter has the same definition as specified in the Diagnosis ASE.

Ext Channel Error type

This parameter has the same definition as specified in the Diagnosis ASE.

Ext Channel Add Value

This parameter has the same definition as specified in the Diagnosis ASE.

Qualified Channel Qualifier

This parameter has the same definition as specified in the Diagnosis ASE.

Manufacturer Specific Diagnosis

This optional parameter is present if the User Structure Identifier is set to a value in the range from MANUFACTURER_SPECIFIC_USI_0x0000 up to MANUFACTURER_SPECIFIC_USI_0x7FFF. It consists of the following parameters:

Diagnosis in USI format shall only be used if Channel Diagnosis, or Extended Channel Diagnosis or Qualified Extended Channel Diagnosis does not cover the diagnosis information.

List of Data

This parameter contains a list of diagnosis data in USI formats. A list element consists of the following parameters.

Data

This parameter contains diagnosis data in USI format.

Attribute type: see Clause 5

Upload and Retrieval Item

This optional parameter is present only if the Alarm Type is set to "Upload and Retrieval Notification" and the User Structure Identifier is set to UPLOAD_AND_RETRIEVAL_USI.

With this parameter a submodule indicates the existence of Record Data objects which shall be stored at the IO controller. The IO controller shall upload and permanently store the Record Data objects. At every following connect sequence and for every following plug and released sequence the IO controller shall convey this additional Record Data objects to the source of the stored Record Data objects. Record Data objects which are part of the GSD description and conveyed from the engineering system to the IO controller shall not be part of the List of Records.

It consists of the following parameters:

List of Records

This parameter contains a list of Record Data objects. A list element consists of the following parameters:

Record Number

This parameter contains the Index of a Record Data object as shown in the Record Data Class in AR ASE.

Allowed Values: 0 to 0xFFFF

Record Length

This parameter contains the Length of a Record Data object as shown in the Record Data Class in AR ASE.

Allowed Values: 1 to $2^{32}-256$

iParameter Item

This optional parameter is present only, if the Alarm Type is set to "Upload and Retrieval Notification" and the User Structure Identifier is set to IPARAMETER_USI.

With this parameter a submodule indicates to upload iParameters or requests to download previously stored iParameters.

The difference between this iParameter Item and the Upload and Retrieval Item defined above is that the iParameter Item uses one record data element, allows segmented transfer of the record data element and transfers the record data on request only.

If the submodule requests an upload, the IO controller shall upload and permanently store the record data object indicated in the alarm.

If the submodule requests a download, the previously stored record data object (result of a previous upload) is transferred to the submodule.

Record Data objects which are part of the GSD description and conveyed from the engineering system to the IO controller shall not be part of this iParameter record data object.

It consists of the following parameters:

iPar Req Header

This parameter contains the request header used for the up-/download of the iParameters. It consists of the following parameters:

SR_Version

This parameter indicates the version of the iParameter protocol.

Attribute type: Unsigned8

Allowed values: 0x01

N_Count

This parameter indicates the notification count of the iParameter protocol.

Attribute type: Unsigned8

Allowed values: 0x01

SR_Type

This parameter indicates the subtype of the iParameter notification.

Attribute type: Unsigned8

Allowed values: 0x01 for requesting an upload, 0x03 for requesting a download

Max Segm Size

This parameter contains the maximum segment size to use for an individual record data read. It thus allows fragmentation of huge iParameters.

Attribute type: Unsigned32

Allowed Values: 1 to $2^{22}-1$

Transfer Index

This parameter contains the record data index to use for the up-/download of the iParameters.

Attribute type: Unsigned32

Allowed values: 0 to 0x7FFF

Total iPar Size

This parameter contains the total size of the iParameters. The parameter shall be set to the size of the iParameters when requesting an upload. On requesting a download the value “0” is allowed; indicating to download the whole parameter set as stored in the IO controller.

Attribute type: Unsigned32

Allowed Values: 1 to $2^{32} - 256$

Manufacturer Alarm Item

This optional parameter is present only, if the Alarm Type is set to “Manufacturer Specific”. In this case the User Structure Identifier shall be set to MANUFACTURER_SPECIFIC_USI_0x0000 up to MANUFACTURER_SPECIFIC_USI_0x7FFF.

List of Data

This parameter contains a list of manufacturer specific alarm data.

The length of the user data shall not exceed 1 408 octets.

A list element is composed of the following parameters:

Data

This parameter contains user specific alarm data.

Attribute type: see Clause 5

Profile Alarm Item

This optional parameter is present only, if the Alarm Type is set to “Profile Specific”. In this case the User Structure Identifier shall be set to PROFILE_SPECIFIC_USI_0x9000 up to PROFILE_SPECIFIC_USI_0x9FFF.

List of Data

This parameter contains a list of profile specific alarm data.

The length of the user data shall not exceed 1 408 octets.

A list element is composed of the following parameters:

Data

This parameter contains user specific alarm data.

Attribute type: see Clause 5

Result(+)

This parameter indicates that the service request succeeded.

Alarm type

This parameter reflects the value from the request.

API

This parameter reflects the value from the request.

Slot Number

This parameter reflects the value from the request.

Subslot Number

This parameter reflects the value from the request.

Alarm Specifier

This parameter reflects the value from the request.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

7.3.1.5.6 Behavior**7.3.1.5.6.1 General behavior of the alarm sink**

Each application process shall maintain a receiving queue for each Alarm type. If used 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.

Alarm queues shall be cleared in case of AR termination and pull/plug/takeover alarms.

7.3.1.5.6.2 General behavior of the alarm source

An alarm is never sent for an associate (exception is the return of submodule alarm) or over an AR in state backup. If there is no owner of a submodule, no alarm is sent.

A new alarm of the same priority remains in the alarm queue on the IO device as long as the acknowledgement from the IO controller regarding the previous alarm is outstanding. This may queue up alarms in the IO device's alarm queue. This queue may contain entries which are obsolete due to other (later) entries (e.g. diagnosis "short circuit" coming and going when having a loose cable). In such a case the IO device may delete entries as an optimization if this deletion doesn't affect the user.

Alarm queues shall be cleared in case of AR termination.

7.3.1.5.6.3 Behavior during (Re-)Parameterization Sequence

Between the starting point of a submodule (re-)parameterization sequence, i.e. one of the services Connect.req (initiated by the IO controller), PrmBegin.req (initiated by the IO controller), Plug.req (initiated by the IO device), or Release.req (initiated by the IO device) and the end point of the sequence, i.e. ApplicationReady.rsp (initiated by the IO controller),

the IO device shall not issue alarms to the IO controller which are related to the submodules being part of the (re-)parameterization sequence.

If at the starting point of the sequence alarm queues contain alarms related to the submodules contained in the sequence, these alarms are deleted from the alarm queues.

The end of the (re-)parameterization sequence is indicated by the IO controller using PrmEnd.req, committed by the IO device using ApplicationReady.req and committed by the IO controller using ApplicationReady.rsp.

However if all application relations are in state backup after ApplicationReady.req, the IO device is not allowed to send diagnosis alarms. If afterwards an application relation turns to state primary, the IO device shall issue diagnosis alarms to match the difference between the diagnosis state as handed out in the ApplicationReady.req's Module Diff Block for this application relation and the current diagnosis state present in the diagnosis ASE.

7.3.1.5.6.4 Behavior of the attribute Submodule State

The attribute behavior of an IO device application is described in 7.3.2.2.6.

7.3.1.6 Record Data class

7.3.1.6.1 Overview

In the Application Layer environment, application processes contain data that remote applications are able to read and write. The Record Data class defines attributes of general purpose Record Data objects and provides a set of services used to read, and write their values.

A particular Record Data object has at least one of the access rights read or write, or even both.

The access to a record data object is entirely ("complete" record) and if supported by the application partially.

The local connection between a Record Data object and a real world or physical object is not in the scope of this definition.

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 Record Data Class allows addressing Record Data objects related to specific layers of the addressing model. The scope of a particular record data object is derived from the Index as described in 7.1.6.2.

The scope of a particular Record Data object as derived from the index is

- a particular submodule within an AP; addressed by API, Slot Number, and Subslot Number.
- a particular module within an AP; addressed by API, and Slot Number.
- a particular AP; addressed by API.
- a particular application relation; addressed by ARUUID or if using an implicit AR by Target AR UUID.
- a device.

Record data may contain parameters or information for channels; whether this is the case is derivable from the record data structure only; it is not expressed in the addressing parameters.

Both IO controller and IO supervisor are able to read or write the value of some Record Data object from an IO device and optional from an IO controller or IO supervisor. The client application process may use the Application Relationship class to establish an application relation for access to Record Data objects of a server.

Write access is permitted to the owner of a submodule; i.e. IO controller or IO supervisor. By using the attribute Device Access an IO supervisor may write Record Data objects if permitted by the application process of the IO device. Read access is possible for all AR types including the implicit AR.

A NULL response for a read record request shall be created if

- the defined filter leads to no matching items (e.g. DiagnosisData)
- the addressed record is supported but was not written (e.g. Adjust or Check)
- the addressed object contains no information (e.g. LogBookData without entry or RecordInputDataObjectElement for inputs without data)

The NULL response is indicated by a positive response of the read, read implicit or read query service having Length set to 0 (i.e. no Data returned).

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 Class 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.

The objects of an IO device can be read using the read record service and the predefined filters addressed by the record indices. The IO device builds the intersection and search for available objects.

The size of the intersection depends on the AR type, the record index and the IO device. For the implicit AR and the IOS AR for DeviceAccess the intersection is only depending on the record index and the IO device. For an IO AR it also depends on the selected submodules of the AR.

EXAMPLE Figure 30 shows the intersection of objects for a read record service with a slot specific index from an IO controller using an IO AR. In this case the IO device should use the "green" intersection to search for e.g. "Diagnosis in channel coding" and send the available data in the read record response.

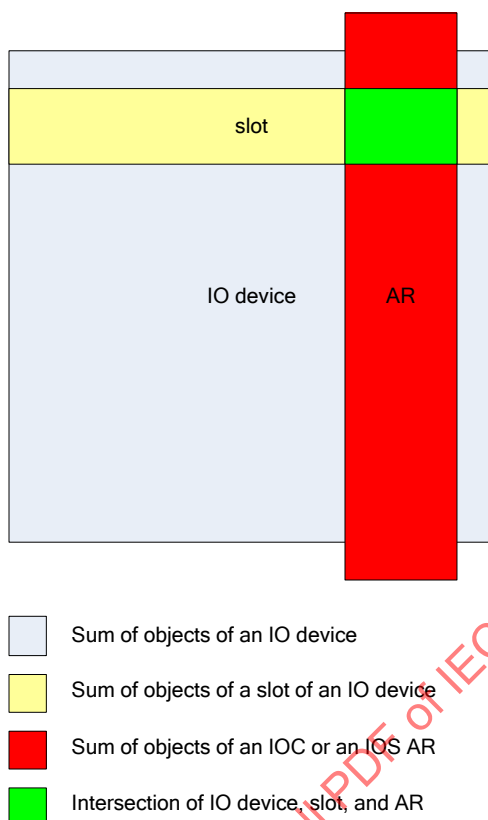


Figure 30 – Example for an intersection of IO device, slot, and AR

7.3.1.6.2 Relations

The RecordData Classes services are bound to other services as shown in Table 163:

Table 163 – Binding Record Data services

Service	Binding	Usage
Read	RPC::Read	used by IO controller or IO supervisor
Read Query	RPC::Read	used by IO controller or IO supervisor
Write	RPC::Write	used by IO controller or IO supervisor
Write Combined Object Container	RPC::Write	used by IO controller or IO supervisor
Local Write Multiple	Local	used by FAL
Local New Write Multiple	Local	used by FAL

7.3.1.6.3 Template

A Record Data object is described by the following template:

ASE: **AR ASE**
CLASS: **Record Data**
CLASS ID: **not used**
PARENT CLASS: **TOP**

ATTRIBUTES:

1	(m)	Key Attribute:	Implicit
2	(m)	Attribute:	List of supported data records
2.1	(m)	Attribute:	Index
2.2	(m)	Attribute:	Read Record Data Description
2.3	(m)	Attribute:	Write Record Data Description

SERVICES:

1	(m)	OpsService:	Read
2	(m)	OpsService:	Read Implicit
3	(o)	OpsService:	Read Query
4	(m)	OpsService:	Write
5	(m)	OpsService:	Write Combined Object Container
6	(m)	OpsService:	Local Write Multiple
7	(m)	OpsService:	Local New Write Multiple

7.3.1.6.4 Attributes**Implicit**

The key attribute Implicit indicates that the User Parameter object is implicitly addressed by the services.

Max Supported Record Size

This attribute indicates the maximum supported record size, when writing a data record.

This attribute list contains the following attributes:

Attribute type: Unsigned32

Default value: 4068

List of supported data records

This attribute list contains the following attributes:

Index

This attribute defines the index of the Record Data object.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF

Read Record Data Description

This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description.

Attribute type: see Clause 5

Write Record Data Description

This attribute may contain either a Simple Data Description or an Array Data Description or a Record Data Description.

Attribute type: see Clause 5

7.3.1.6.5 Services

7.3.1.6.5.1 Read Services

7.3.1.6.5.1.1 Read

This confirmed service may be used to read the value of a Record Data object. Table 164 shows the parameters of the services.

Table 164 – Read

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(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
PNIO Status			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

If the scope of the record data object is subslot, slot or API, this parameter shall be used to address the desired AP.

Slot Number

If the scope of the record data object is subslot or slot, this parameter shall be used to address the desired slot (typically a module).

Subslot Number

If the scope of the record data object is subslot, this parameter shall be used to address the desired subslot (typically a submodule).

Index

The parameter Index shall be used to address the desired Record Data object. It is additionally used to derive the scope of the Record Data object (see IEC 61158-6-10).

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 parameter Seq Number shall be incremented for each service request during a session.

If for example a write service request had the Seq Number set to “17”, a following read service request shall have the Seq Number set to “18”.

When starting a new session the sequence number shall start with 0. A confirmation with a Seq Number that is not outstanding will be ignored. An indication 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 maximum number of octets to be read, as provided by the client. It describes the size of the buffer provided to hold the read result.

If the requested length is greater or equal actual length then the actual length with data shall be responded. If the requested length is shorter than the actual length then a parameter error shall be the result.

Note that the parameter Length may exceed the Max Supported Record Size as provided by the IO device. Max Supported Record Size is used for Write requests only.

Attribute type: Unsigned32

Allowed values: 1 to $2^{32} - 256$

Result(+)

This parameter indicates that the service request succeeded.

Length

The parameter Length indicates the number of octets of the Data object that has been read.

Data

The parameter Data contains the value of the specific object which has been read and consists of the number of octets indicated in the Length. This parameter has to be composed of the data types defined in Clause 5. A specific meaning for the different services of this data exists, which is described in the special service parts. Table 165 shows the list of special read services.

Table 165 – Read Services

Service Name	Subclause
Read PD Port Statistic	7.3.3.2.5.22
Read AR Data for one API	7.3.1.2.5.10.2
Read AR Data for device	7.3.1.2.5.10.3
Read Module Diff Block for one AR	7.3.1.4.5.5
Read Expected Fast Startup Data	7.3.1.2.5.11
Read Expected Identification Data for one subslot	7.3.1.4.5.2
Read Expected Identification Data for one slot	7.3.1.4.5.3
Read Expected Identification Data for one AR	7.3.1.4.5.4
Read Record Input Data Object Element	7.3.2.2.5.10
Read Record Output Data Object Element	7.3.2.2.5.11
Read Substitute Data	7.3.2.2.5.12
Read GSD Data	7.3.2.2.5.14
Read Real Identification Data for one subslot	7.3.2.2.5.4
Read Real Identification Data for one slot	7.3.2.2.5.6
Read Real Identification Data for one AR	7.3.2.2.5.7
Read Real Identification Data for one API	7.3.2.2.5.8
Read API Data	7.3.2.2.5.9
Read I&M0 Filter Data	7.3.2.3.5.1
Read I&M4 Data	7.3.2.3.5.10
Read I&M5 Data	7.3.2.3.5.11
Read I&M0 Data	7.3.2.3.5.2
Read I&M1 Data	7.3.2.3.5.4
Read I&M2 Data	7.3.2.3.5.6
Read I&M3 Data	7.3.2.3.5.8
Read PDev Data	7.3.3.2.5.1
Read PD NC Data Check	7.3.3.2.5.10
Read PD Port Data Real	7.3.3.2.5.11
Read PD Port Data Check	7.3.3.2.5.14
Read PD Port Data Adjust	7.3.3.2.5.16
Read PD Port FO Data Real	7.3.3.2.5.17
Read PD Port FO Data Check	7.3.3.2.5.19
Read PD Real Data	7.3.3.2.5.2
Read PD Port FO Data Adjust	7.3.3.2.5.21
Read Autoconfiguration Data	7.3.3.8.5.1
Read PD Expected Data	7.3.3.2.5.3
Read PD Interface Data Real	7.3.3.2.5.4
Read PD Interface Adjust	7.3.3.2.5.5
Read PD Interface FSU Data Adjust	7.3.3.2.5.8
Read PD Sync Data	7.3.3.3.5.2
Read PD IR Data	7.3.3.4.5.2
Read PD IR Subframe Data	7.3.3.4.5.4
Read PD Interface MRP Data Real	7.3.3.5.5.1

Service Name	Subclause
Read PD Interface MRP Data Check	7.3.3.5.5.3
Read PD Interface MRP Data Adjust	7.3.3.5.5.5
Read PD Port MRP Data Real	7.3.3.5.5.6
Read PD Port MRP Data Adjust	7.3.3.5.5.8
Read PD Port MRPIC Data Real	7.3.3.5.5.9
Read PD Port MRPIC Data Check	7.3.3.5.5.11
Read PD Port MRPIC Data Adjust	7.3.3.5.5.13
Read PD Time Data	7.3.3.6.5.2
Read Diagnosis in channel coding for one subslot	7.3.4.2.5.5
Read Diagnosis in all codings for one subslot	7.3.4.2.5.5.2
Read Maintenance required in channel coding for one subslot	7.3.4.2.5.5.3
Read Maintenance required in all codings for one subslot	7.3.4.2.5.5.4
Read Maintenance demanded in channel coding for one subslot	7.3.4.2.5.5.5
Read Maintenance demanded in all codings for one subslot	7.3.4.2.5.5.6
Read Diagnosis, Maintenance, Qualified and Status for one subslot	7.3.4.2.5.5.7
Read Diagnosis in channel coding for one slot	7.3.4.2.5.6
Read Diagnosis in all codings for one slot	7.3.4.2.5.6.2
Read Maintenance required in channel coding for one slot	7.3.4.2.5.6.3
Read Maintenance required in all codings for one slot	7.3.4.2.5.6.4
Read Maintenance demanded in channel coding for one slot	7.3.4.2.5.6.5
Read Maintenance demanded in all codings for one slot	7.3.4.2.5.6.6
Read Diagnosis, Maintenance, Qualified and Status for one slot	7.3.4.2.5.6.7
Read Diagnosis in channel coding for one AR	7.3.4.2.5.7
Read Diagnosis in all codings for one AR	7.3.4.2.5.7.2
Read Maintenance required in channel coding for one AR	7.3.4.2.5.7.3
Read Maintenance required in all codings for one AR	7.3.4.2.5.7.4
Read Maintenance demanded in channel coding for one AR	7.3.4.2.5.7.5
Read Maintenance demanded in all codings for one AR	7.3.4.2.5.7.6
Read Diagnosis, Maintenance, Qualified and Status for one AR	7.3.4.2.5.7.7
Read Diagnosis in channel coding for one API	7.3.4.2.5.8
Read Diagnosis in all codings for one API	7.3.4.2.5.8.2
Read Maintenance required in channel coding for one API	7.3.4.2.5.8.3
Read Maintenance required in all codings for one API	7.3.4.2.5.8.4
Read Maintenance demanded in channel coding for one API	7.3.4.2.5.8.5
Read Maintenance demanded in all codings for one API	7.3.4.2.5.8.6
Read Diagnosis, Maintenance, Qualified and Status for one API	7.3.4.2.5.8.7
Read Diagnosis, Maintenance, Qualified and Status for device	7.3.4.2.5.9
Read PE Entity Filter Data for one AR	7.3.5.2.5.4.2
Read PE Entity Filter Data for device	7.3.5.2.5.4.3
Read PE Entity Status Data for one subslot	7.3.5.2.5.5.2
Read PE Entity Status Data for one AR	7.3.5.2.5.5.3
Read PE Entity Status Data for device	7.3.5.2.5.5.4
Read LogBook Data	7.3.6.2.5.1
Read Isochronous Mode Data	7.3.2.4.5.2

Service Name	Subclause
RS Get Event	7.3.7.2.5.2
RS Adjust Observer	7.3.7.2.5.4

Result(-)

This parameter indicates that the service request failed.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.6.5.1.2 Read Implicit

This confirmed service may be used to read the value of a Record Data object using an implicit AR.

If read implicit is rejected by RPC error NCA_server_too_busy (indicating that the resources for implicit reads are used up currently for other clients), the client should retry the service after waiting a random time between 1 s and 10 s.

Table 166 shows the parameters of the service.

Table 166 – Read Implicit

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
API	M	M(=)		
Target AR UUID	C	C(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Length	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Length			M	M(=)
Data			M	M(=)

Parameter name	req	ind	rsp	cnf
Result(-)			S	S(=)
AREP			M	M(=)
PNIO Status			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. It contains the parameters similar to the Read Service (see 7.3.1.6.5.1.1) and an additional parameter:

Target AR UUID

For a record data object with scope application relation, this parameter indicates the target AR.

Result(+)

This parameter indicates that the service request succeeded. It contains the parameters of the Read Service Result (+) (see 7.3.1.6.5.1.1).

Result(-)

This parameter indicates that the service request failed. It contains the parameters of the Read Service Result (-) (see 7.3.1.6.5.1.1).

7.3.1.6.5.1.3 Read Query

This optional confirmed service may be used to read the value of a Record Data object.

To optimize the access to many small data objects a server may offer different combinations of this data behind one Index. In this case, the content of the Record Data object is defined by the Selector. Table 167 shows the parameters of the service.

Table 167 – Read Query

Parameter name	req	ind	rsp	cnf
Argument	M	M(=)		
AREP	M	M(=)		
API	M	M(=)		
Target AR UUID	C	C(=)		
Slot Number	M	M(=)		
Subslot Number	M	M(=)		
Index	M	M(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Selector	M	M(=)		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Length			M	M(=)
Data			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)

Parameter name	req	ind	rsp	cnf
Seq Number			M	M(=)
PNIO Status			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. It contains the parameters of the Read Service (see 7.3.1.6.5.1.1) and an additional parameter:

Selector

The parameter Selector is used by the server to identify the requested data. To optimize the access to many small data objects a server may offer different combinations of this data behind one Index and select the requested collection with the Selector.

Result(+)

This parameter indicates that the service request succeeded. It contains the parameters of the Read Service Result (+) (see 7.3.1.6.5.1.1).

Result(-)

This parameter indicates that the service request failed. It contains the parameters of the Read Service Result (-) (see 7.3.1.6.5.1.1).

7.3.1.6.5.2 Write Services

7.3.1.6.5.2.1 Write

This confirmed service may be used to write the value of a Record Data object.

In addition to this common write service, additional services are defined for write specific application data. All this write services use the same parameters, only the data field is different. Table 168 shows the parameters of the services.

Table 168 – 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(=)		
Seq Number	M	M(=)		
Length	M	M(=)		
Data	M	M(=)		
Prm Flag		M		
Result(+)			S	S(=)
AREP			M	M(=)
Seq Number			M	M(=)
Result(-)			S	S(=)
AREP			M	M(=)

Parameter name	req	ind	rsp	cnf
Seq Number			M	M(=)
PNIO Status			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. It contains the parameters of the Read Service (see 7.3.1.6.5.1.1) and additional parameters:

Length

The parameter Length indicates the number of octets to be written.

The parameter Length shall not exceed the Max Supported Record Size as provided by the IO device.

If the requested length does not match the length of the actual data record an error shall be the result.

Attribute type: Unsigned32

Allowed values: 1 to 2^{32} - 256

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 Clause 5. A specific meaning for the different services of this data exists, which is described in the special service parts. Table 169 shows the list of special write services.

Table 169 – Write Services

Service Name	Subclause
Write Expected Fast Startup Data	7.3.1.2.5.12
Write Substitute Value	7.3.2.2.5.13
Write I&M1 Data	7.3.2.3.5.3
Write I&M2 Data	7.3.2.3.5.5
Write I&M3 Data	7.3.2.3.5.7
Write I&M4 Data	7.3.2.3.5.9
Write PD Interface Adjust	7.3.3.2.5.5
Write PD Interface FSU Data Adjust	7.3.3.2.5.7
Write PD NC Data Check	7.3.3.2.5.9
Write PD Port Data Check	7.3.3.2.5.12
Write PD Port Data Adjust	7.3.3.2.5.15
Write PD Port FO Data Check	7.3.3.2.5.18
Write PD Port FO Data Adjust	7.3.3.2.5.20
Write PD Sync Data	7.3.3.3.5.1
Write PD IR Data	7.3.3.4.5.1
Write PD IR Subframe Data	7.3.3.4.5.3
Write PD Interface MRP Data Check	7.3.3.5.5.2
Write PD Interface MRP Data Adjust	7.3.3.5.5.4
Write PD Port MRP Data Adjust	7.3.3.5.5.7

Service Name	Subclause
Write PD Port MRPIC Data Check	7.3.3.5.5.10
Write PD Port MRPIC Data Adjust	7.3.3.5.5.12
Write PD Time Data	7.3.3.6.5.1
Write Isochronous Mode Data	7.3.2.4.5.1
RS Ack Event	7.3.7.2.5.3
RS Adjust Observer	7.3.7.2.5.4

Prm Flag

The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed. It contains the parameters of the Read Service Result (-) (see 7.3.1.6.5.1.1).

7.3.1.6.5.2.2 Write Combined Object Container

This confirmed service allows writing the combined object container to the combined object access point (see 7.1.4.5.5 for details).

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 170 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 170 – Data elements of Write Combined Object Container

Parameter name	service specific elements	req	ind
Data	List of Container	M	M(=)
	API	M	M(=)
	Slot Number	M	M(=)
	Subslot Number	M	M(=)
	Index	M	M(=)
	Length	M	M(=)
	Data	M	M(=)

Data

This parameter contains the value of the object to write.

List of Container

The service transmits a number of data records with this list to write to the access point. Each list element consists of the following elements:

API

If the scope of the record data object is subslot, slot or API, this parameter shall be used to address the desired AP.

Slot Number

If the scope of the record data object is subslot or slot, this parameter shall be used to address the desired slot (typically a module).

Subslot Number

If the scope of the record data object is subslot, this parameter shall be used to address the desired subslot (typically a submodule).

Index

The parameter Index shall be used to address the desired Record Data object. It is additionally used to derive the scope of the Record Data object.

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 $2^{32} - 256$.

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 Clause 5. A specific meaning for the different services of this data exists, which is described in the special service parts. Table 169 shows the list of special write services.

7.3.1.6.5.2.3 Local Write Multiple

This local service may be used to convey multiple Record Data objects within one APDU. This service shall only be used in conjunction with the IO AR or Supervisor AR. Table 171 shows the parameters of the service.

It is possible to convey Record Data objects and attributes from multiple ASEs (e.g. Communication Interface Management ASE) in one APDU.

This local service is mapped by building a surrounding data record from the individual address information (API, slot, subslot, index), sequence number and data of the particular data records. This data record is then transmitted to the device by using the service Write.req (see 7.3.1.6.5.2.1).

Within the device such a surrounding data record is recognized by its special index (see IEC 61158-6-10 for its definition; in particular the APDU substitution IODWriteMultipleReq). Within the device a receipt of such a data record (indicated via Write.ind) is then mapped to the local service Local New Write Multiple.

Table 171 – Local Write Multiple

Parameter name	req	cnf
Argument	M	
AREP	M	
Seq Number	M	
List of Data Records	M	
API	M	
Slot Number	M	
Subslot Number	M	
Index	M	
Seq Number	M	
Length	M	
Data	M	

Parameter name	req	cnf
Prm Flag		
Result(+)		S
AREP		M
Seq Number		M
List of Results		M
Seq Number		M
PNIO Status		M
Add Data 1		M
Add Data 2		M
Result(-)		S
AREP		M
Seq Number		M
PNIO Status		M
Add Data 1		M
Add Data 2		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

See Definition in 7.3.1.6.5.1.1.

The Seq Number for the Write Multiple data record is the lowest one.

NOTE 1 This seq number is used for the write multiple service.

List of Data Records

The service transmits a number of data records with this list. Each list element is contained of the following elements:

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.

Seq Number

See Definition in 7.3.1.6.5.1.1.

The Seq Number for each data record in the list is in ascending order.

NOTE 2 This seq number is used for the individual data record.

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 Clause 5.

Prm Flag

The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

Result(+)

This parameter indicates that the service request succeeded.

Result(-)

This parameter indicates that the service request failed.

7.3.1.6.5.2.4 Local New Write Multiple

This local service indicates the receipt of multiple Record Data objects within one APDU. This service shall only be used in conjunction with the IO AR or Supervisor AR. Table 172 shows the parameters of the service.

It is possible to convey Record Data objects and attributes from multiple ASEs (e.g. Communication Interface Management ASE) in one APDU.

This local service is mapped by receipt of a surrounding data record indicated by receiving the special index (see IEC 61158-6-10 for its definition; in particular the APDU substitution IODWriteMultipleReq) on the service indication Write.ind (see 7.3.1.6.5.2.1).

The individual address information (API, slot, subslot, index), sequence number and data of the particular data records are extracted from the surrounding data record by the application and presented in this service's indication.

If the surrounding data record is not decodable (due to its internal structure) the AR is aborted with protocol error. See IEC 61158-6-10 for details.

Otherwise the particular data records are applied one by one onto the application by using the address information and the data. The results of the particular data record writes are collected together (see IEC 61158-6-10 for its definition; in particular the APDU substitution IODWriteMultipleReq) and returned via Write.rsp(+).

The local service Write Multiple.cnf then extracts the particular results from the returned information and returns it to the application.

Table 172 – Local New Write Multiple

Parameter name	ind	rsp
Argument	M(=)	
AREP	M(=)	
Seq Number	M(=)	
Slot Number	M(=)	
Subslot Number	M(=)	
Index	M(=)	
Seq Number	M(=)	
Length	M(=)	
Data	M(=)	
Prm Flag	M	
Result(+)		S
AREP		M
Seq Number		M
List of Results		M
Seq Number		M
PNIO Status		M
Add Data 1		M
Add Data 2		M
Result(-)		S
AREP		M
Seq Number		M
PNIO Status		M
Add Data 1		M
Add Data 2		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

See Definition in 7.3.1.6.5.1.1.

NOTE 1 This seq number is used for the write multiple service.

List of Data Records

The service transmits a number of data records with this list. Each list element is contained of the following elements:

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.

Seq Number

See Definition in 7.3.1.6.5.1.1.

NOTE 2 This seq number is used for the individual data record.

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 Clause 5.

Prm Flag

The local indication parameter shall contain the value TRUE while parameterization during the connection establishment phase. Otherwise, the value shall be set to FALSE.

Attribute type: Boolean

Result(+)

This parameter indicates that the service request itself succeeded; note however that the PNIO Status of some or even all write results may indicate an error.

List of Results

This list contains the result of each data record access. Each list element is contained of the following elements:

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

Result(-)

This parameter indicates that the service request failed totally and none of the data records has been written.

PNIO Status

This parameter shall contain the actual error code according to the protocol definition.

Attribute type: Unsigned32

Add Data 1

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

Attribute type: Unsigned 16

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.

Attribute type: Unsigned 16

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

7.3.1.6.6 Behavior

7.3.1.6.6.1 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 $2^{32} - 65$ octets.
- The attribute Record Data description has to be set accordingly.

7.3.1.6.6.2 Length of Record Data Objects

A Record Data object shall not exceed the total length of $4\,294\,967\,231 (2^{32} - 65)$ octets.

If a record data object is sub-structured into blocks, the length of a particular block (due to the definition of the block header in IEC 61158-6-10 “APDU abstract syntax”) shall not exceed 65 533 octets.

7.3.1.6.6.3 Error code hierarchy of the Record Data object

The PNIO Status returned from the record data services shall be set according to the current error and shall consist of the elements Error code, Error Decode, Error code 1, and Error code 2.

If PNIO Status is set to (0x00, 0x00, 0x00, 0x00) the special case “No error” applies.

It is recommended to apply the error checks when writing a Record Data object in the following order:

- a) If service violates the IEC 61158-6-10 “PDU checking rules”, PNIO status is set according to IEC 61158-6-10 “PDU checking rules”.
- b) If AR state is set to BACKUP:
- c) PNIO Status = { IODWriteRes, PNIORW, “access / backup”, user specific value }
- d) If AR is not owner of the submodule or is a device access AR and application does not grant write access:
- e) PNIO Status = { IODWriteRes, PNIORW, “access / access denied”, user specific value }
- f) See 7.3.2.2.6.2.3 for details.
- g) If index is not supported:
- h) PNIO Status = { IODWriteRes, PNIORW, “access / invalid index”, ec2 }
- i) ec2 is set to 10 (recommended) or a user specific value.

- j) If write-access is not supported:
- k) PNIO Status = { IODWriteRes, PNIORW, ec1, user specific value }
- l) ec1 is set to “application / write error”, “access / type conflict”, “access / state conflict”, or “access / access denied”.
- m) Content errors in the structure of data record object to be written, or partial write onto a data record not supporting partial access:
- n) PNIO Status = { IODWriteRes, PNIORW, ec1, user specific value }
- o) ec1 is set to “access / write length error”, “access / invalid range”, “access / invalid parameter” or “access / invalid type”.
- p) Resource errors during access:
- q) PNIO Status = { IODWriteRes, PNIORW, ec1, user specific value }
- r) ec1 is set to “resource / read constrain conflict”, “resource / write constrain conflict”, “resource / resource busy”, “resource / resource unavailable”, or “resource / user specific 12” to “resource / user specific 19”.
- s) This indicates a temporary error and may lead to a successful access when the clients retries.
- t) Application errors during access:
- u) PNIO Status = { IODWriteRes, PNIORW, ec1, user specific value }
- v) Error code 1 is set to “application / module failure”, “application / busy”, “application / version conflict”, “application / feature not supported”, or “application / user specific 1” to “application / user specific 6”.
- w) User specific errors during access:
- x) PNIO Status = { IODWriteRes, PNIORW, “user specific / user specific”, user specific value }

It is recommended to apply the error checks when reading a Record Data object in the following order:

- y) If service violates the IEC 61158-6-10 “PDU checking rules”, PNIO status is set according to IEC 61158-6-10 “PDU checking rules”.
- z) If index is not supported:
- aa) PNIO Status = { IODReadRes, PNIORW, “access / invalid index”, ec2 }
- bb) ec2 is set to 10 (recommended) or a user specific value.
- cc) If read-access is not supported:
- dd) PNIO Status = { IODReadRes, PNIORW, ec1, user specific value }
- ee) ec1 is set to “application / read error”, “access / type conflict”, “access / state conflict”, or “access / access denied”.
- ff) Partial read onto a data record not supporting partial access:
- gg) PNIO Status = { IODReadRes, PNIORW, ec1, user specific value }
- hh) Error code 1 is set to “access / invalid range”, or “access / invalid type”.
- ii) Resource errors during the access onto the data record object:
- jj) PNIO Status = { IODReadRes, PNIORW, ec1, user specific value }
- kk) ec1 is set to “resource / read constrain conflict”, “resource / write constrain conflict”, “resource / resource busy”, “resource / resource unavailable”, or “resource / user specific 12” to “resource / user specific 19”.
- ll) This indicates a temporary error and may lead to a successful access when the clients retries.
- mm) Application errors during the access onto the data record object:
- nn) PNIO Status = { IODReadRes, PNIORW, ec1, user specific value }

- oo) Error code 1 is set to “application / module failure”, “application / busy”, “application / version conflict”, “application / feature not supported”, or “application / user specific 1” to “application / user specific 6”.
- pp) User specific errors during the access onto the data record object:
- qq) PNIO Status = { IODReadRes, PNIORead, “user specific / user specific”, user specific value }

7.3.1.6.6.4 Focus of the record data object

A particular record data object is associated to an AR context in one of the following modes

- The AR context is needed to access the record data object.
- The AR context is used to filter the information of the record data object.
- The AR context has no meaning for accessing the record data object.

If the AR context is needed for the record data object, the information is always related to the expected identification data (see Expected Identification class).

The information of a record data object is filtered according to the following rules:

- If the record data object is addressed over the established AR or over implicit AR with Target AR UUID != NIL, the information
 - is filtered according to the expected identification data (see Expected Identification class), if the addressed object is part of the addressed application relation.

NOTE 1 e.g. by getting diagnosis information from a slot, where at least one submodule is part of the expected identification data, only diagnosis related to submodules within the addressed slots, which are part of the expected identification data is returned.

- is filtered according to the real identification data (see Real Identification class) if the addressed object is not part of the addressed application relation.

NOTE 2 e.g. by getting diagnosis information from a slot, where no submodule is part of the expected identification data, diagnosis related to all submodules within the addressed slots, which are part of the real identification data is returned.

- If the record data object is addressed over implicit AR with Target AR UUID == NIL, the information is always filtered according to the real identification data (see Real Identification class).

The reason for these filter rules is that the user wants to see only the objects he is interested in.

If the AR context has no meaning for the record data object, the information is always related to the real identification data (see Real Identification class).

A Device Access AR does not provide an expected identification. Thus, a DeviceAccess AR belongs to NO API. Thus, the API-filtered records are empty for a DeviceAccess.

7.3.1.6.6.5 Parameterization behavior

The parameterization of a submodule is done by using the Record Data Class Write service.

The error code hierarchy stated in 7.3.1.6.6.1 shall be applied as fully as possible.

If the data record write service request succeeded, but the data record content evaluation leads to a parameterization error (for example due to interdependencies between several data records needed for the complete parametrization, to a delayed parameter evaluation, combined object container, etc.), a parameter missing or parameter fault diagnosis shall be added to the diagnosis ASE for the addressed submodule (see 7.3.4.1.4). The diagnosis' ExtChannelAddValue contains an index (if known) of the data record and the offset (if known)

of the faulty parameter. If multiple errors are detected one arbitrary error (for example the first detected) is stated.

For the parameterization within a parameterization sequence, initiated with Connect.req, Prm Begin.req, or Plug.req and ended with Prm End.req, the submodule shall act as if it was reset to its default and applied just those parameters written during the parameterization sequence.

This may be implemented in different ways:

- With begin of the parameterization sequence reset the submodule to its default and apply the parameters written in the parameterization sequence.
- Or optimized: At the end of the parametrization sequence determine the delta between the current parameterization of the submodule and the one to apply. Apply smart the parameterization delta to reduce the impact onto the periphery.

7.3.1.6.6.6 Internal sorting

If a data record defined within this specification contains indexable elements,

- the elements should be returned on a read request in ascending order (for example API, slot number, subslot number)
- the elements should be provided on a write request in ascending order.

7.3.1.6.6.7 Persistency

Whether a write to a record data object is stored non-volatile in the device's persistency is defined in IEC 61158-6-10.

The persistency of the device is attached to the Configuration Interface Management ASE; see 7.3.3.2.6.2 for details.

7.3.2 Real Identification ASE

7.3.2.1 Overview

The Real Identification ASE represents physically available APOs. It represents all APOs and conveys all events (for example pull and plug, parameterization) from and to the APOs.

The APO represents the submodule with its channels and therefor is an access point to the application. Behind them is the physical module as a real object. The Real Identification ASE works on submodule granularity.

The Real Identification ASE is organized in 3 classes:

- Real Identification class: managing the APOs and the ownership
- I&M Data class: managing the I&M data of the APOs
- Isochronous Mode Application class: managing necessary parameter for isochronous mode

There are two places where IO data exist: In the Real Identification ASE, the IO data are stored in the Real Identification Format. In the Communication Relationship class the IO data are stored in the Expected Identification Format.

In Isochronous Mode, the application has to use the FAL services for reading and writing of data. These need to be delivered or collected at the correct timepoint.

The record data for the Isochronous Mode are stored in the Expected Identification.

7.3.2.2 Real Identification class

7.3.2.2.1 Overview

See 7.3.2.1.

7.3.2.2.2 Relations

The services are bound to other services as shown in Table 173:

Table 173 – Binding real identification services

Service	Binding	Usage
Local Add Submodule	Local	used by FAL
Local Update Submodule	Local	used by FAL
Local Remove Submodule	Local	used by FAL
Read Real Identification Data for one subslot	AR:Record Data:Read	used by IO controller or IO supervisor
Read Real Identification Data for one slot	AR:Record Data:Read	used by IO controller or IO supervisor
Read Real Identification Data for one AR	AR:Record Data:Read	used by IO controller or IO supervisor
Read Real Identification Data for one API	AR:Record Data:Read	used by IO controller or IO supervisor
Read API Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read Record Input Data Object Element	AR:Record Data:Read	used by IO controller or IO supervisor
Read Record Output Data Object Element	AR:Record Data:Read	used by IO controller or IO supervisor
Read Substitute Value	AR:Record Data:Read	used by IO controller or IO supervisor
Write Substitute Value	AR:Record Data:Write	used by IO controller or IO supervisor
Read GSD Data	AR:Record Data:Read	used by IO controller or IO supervisor
Local	AR:Alarm:Alarm Notification	used by RI class

7.3.2.2.3 Template

This class specifies the real identification; i.e. the submodules of the IO device.

The real identification object is described by the following template:

ASE: Real Identification ASE

CLASS: Real Identification

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: List of Real Submodules
- 2.1 (m) Attribute: API
- 2.2 (m) Attribute: Slot Number
- 2.3 (m) Attribute: Subslot Number
- 2.4 (m) Attribute: Module Ident Number
- 2.5 (m) Attribute: Submodule Ident Number
- 2.6 (m) Attribute: Owner

2.7	(m) Attribute:	Submodule Properties
2.7.1	(m) Attribute:	Type
2.8	(m) Attribute:	List of Data Descriptions
2.8.1	(m) Attribute:	Data Direction
2.8.2	(m) Attribute:	Submodule Data Length
2.8.3	(m) Attribute:	Length IOPS
2.8.4	(m) Attribute:	Length IOCS
2.9	(m) Attribute:	Substitution Mode
2.10	(c) Constraint:	Substitution Mode = REPLACEMENT_VALUE or
PROFILE_SPECIFIC		
2.10.1	(m) Attribute:	Substitution Value
3	(o) Attribute:	GSD Data
SERVICES:		
1	(m) OpsService:	Local Add Submodule
2	(m) OpsService:	Local Update Submodule
3	(m) OpsService:	Local Remove Submodule
4	(m) OpsService:	Read Real Identification Data for one subslot
5	(m) OpsService:	Read Real Identification Data for one slot
6	(m) OpsService:	Read Real Identification Data for one AR
7	(m) OpsService:	Read Real Identification Data for one API
8	(m) OpsService:	Read API Data
9	(m) OpsService:	Read Record Input Data Object Element
10	(m) OpsService:	Read Record Output Data Object Element
11	(o) OpsService:	Read Substitute Value
12	(o) OpsService:	Write Substitute Value
13	(o) OpsService:	Read GSD Data

7.3.2.2.4 Attributes

Implicit

The key attribute Implicit indicates that the real identification object is implicitly addressed by the services.

List of Real Submodules

This list contains the list of real submodules:

One real submodule is composed of the following list elements:

API

This attribute defines the number of the application process the submodule is assigned to.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Slot Number

This attribute defines the slot the submodule is assigned to.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines the subslot the submodule is assigned to.

Attribute type: Unsigned16

Allowed values: 0 to 0x9FFF

Module Ident Number

This attribute defines the identification of the module.

Attribute type: Unsigned32

Allowed values: 1 to 0xFFFFFFFF

NOTE 1 The attribute value is manufacturer specific. See 7.1.4.5.4 for details.

Submodule Ident Number

This attribute defines the identification of the submodule.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

NOTE 2 The attribute value is manufacturer specific. See 7.1.4.5.4 for details.

Owner

This attribute defines the current owner of the submodule. It holds the AREP of the AR which has the ownership of the submodule.

Special cases: The value NONE indicates that the submodule is currently not owned, the value SOL indicates that the submodule is superordinated locked by the application.

Allowed values: AREP, SOL, NONE

Submodule Properties

This attribute consists of the following attributes.

Type

This attribute defines the type of the submodule.

Attribute type: Unsigned16

Allowed values: NO_IO, INPUT, OUTPUT, IO

List of Data Descriptions

The following rules apply for the data descriptions.

There shall be one data description with Data Direction = DIRECTION_INPUT for a submodule of Type NO_IO or INPUT.

There shall be one data description with Data Direction = DIRECTION_OUTPUT for a submodule of Type OUTPUT.

There shall be a data descriptions with Data Direction = DIRECTION_INPUT and followed by a data description with Data Direction = DIRECTION_OUTPUT for a submodule of Type IO.

This attribute list consists of the following attributes.

Data Direction

The attribute defines the direction of the data description.

Attribute type: Unsigned16

Allowed values: DIRECTION_INPUT, DIRECTION_OUTPUT

Submodule Data Length

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

Submodule Data Length is independent of the value of Reduce Input/Output Submodule Data Length.

Attribute type: Unsigned16

Allowed values: 0 to 1 439

Length IOPS

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

Length IOPS is independent of the value of Discard IOXS.

Attribute type: Unsigned8

Allowed values: 1

Length IOCS

The attribute Length IOCS defines the number of octets of the IOCS. Length IOPS is independent of the value of Discard IOXS.

Attribute type: Unsigned8

Allowed values: 1

Substitution Mode

This attribute defines the substitution mode of the submodule for the output data object element.

Attribute type: Unsigned16

Allowed values: ZERO, LAST_VALUE, REPLACEMENT_VALUE, PROFILE_SPECIFIC

Substitution Value

This conditional attribute is present when the value of the Substitution Mode attribute is "REPLACEMENT_VALUE" or "PROFILE_SPECIFIC". It contains the value to be substituted.

Attribute type: according to data description

GSD Data

This optional attribute hosts the GSD for this IO device.

Attribute type: OctetString

7.3.2.2.5 Services

7.3.2.2.5.1 General Data Definition for Identification Data

See 7.3.1.4.5.1.

7.3.2.2.5.2 Local Add Submodule

This local service shall be used to add (plug) a submodule into the real identification data.

Table 174 – Local Add Submodule

Parameter name	req	cnf
Argument	M(=)	M(=)
API	M(=)	M(=)
Slot Number	M(=)	M(=)
Subslot Number	M(=)	M(=)
Module Ident Number	M(=)	M(=)
Submodule Ident Number	M(=)	M(=)

Argument

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

API

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

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.

Module Ident Number

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

Submodule Ident Number

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

7.3.2.2.5.3 Local Remove Submodule

This local service shall be used to remove (pull) a submodule from the real identification data.

Table 175 – Local Remove Submodule

Parameter name	req	cnf
Argument	M(=)	M(=)
API	M(=)	M(=)
Slot Number	M(=)	M(=)
Subslot Number	M(=)	M(=)

Argument

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

API

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

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.

7.3.2.2.5.4 Local Update Submodule

This local service shall be used for combined remove (pull) and add (plug) of a submodule in the real identification data.

Table 176 – Local Update Submodule

Parameter name	req	cnf
Argument	M(=)	M(=)
API	M(=)	M(=)
Slot Number	M(=)	M(=)
Subslot Number	M(=)	M(=)
Module Ident Number	M(=)	M(=)
Submodule Ident Number	M(=)	M(=)

Argument

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

API

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

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.

Module Ident Number

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

Submodule Ident Number

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

7.3.2.2.5.5 Read Real Identification Data for one subslot

This service allows reading the real identification data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

7.3.2.2.5.6 Read Real Identification Data for one slot

This service allows reading the real identification data using a record data object with slot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context may be used for this service; see 7.3.1.6.6.4 for details.

The response may vary depending on the vendor of the device:

- The real identification data for this slot not filtered by the AR, or
- If the slot is contained in the expected identification data of the AR, use the expected identification to filter the real identification and return the intersecting set; otherwise, return the real identification of the slot without filter.

7.3.2.2.5.7 Read Real Identification Data for one AR

This service allows reading the real identification data using a record data object with AR scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context is needed for this service; see 7.3.1.6.6.4 for details.

7.3.2.2.5.8 Read Real Identification Data for one API

This service allows reading the real identification data using a record data object with API scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.1.4.5.1 for its general description.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

7.3.2.2.5.9 Read API Data

This service allows reading the supported application profiles using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 177 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 177 – Data elements of Read API Data

Parameter name	service specific elements	rsp	cnf
Data	List of APs	M	M(=)
	API	M	M(=)

Data

This parameter contains the values of the object which has been read.

List of APs

This parameter is composed of the following list elements:

API

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

7.3.2.2.5.10 Read Record Input Data Object Element

This confirmed service allows reading the value of an Input Data object element.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1). But the parameter Data has a service specific structure and meaning. Table 178 shows the specific elements of Data for this service.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

This service returns “access / invalid index” for submodules without input data; i.e. for submodules having output data only, or having no IO data.

If the submodule is owned by the AR specified in the service (over the AREP resp. the Target AR UUID), the input data object element as seen/provided by the owner of the submodule is returned. If the owner sees OK or SUBSTITUTE, the input data is returned accordingly (i.e. input data, IOPS from the device, IOCS from the IO controller owning the submodule), if the owner sees WRONG or NO_SUBMODULE, the service returns “access / invalid index”.

If the submodule is not owned by the AR, the service returns the real data of the submodule. If the real data cannot be determined by the application, the service returns “access / invalid index”.

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

If the submodule is not owned by an AR, the service returns the real data of the submodule. If the real data cannot be determined by the application, the service returns “access / invalid index”.

Table 178 – Data elements of Read Record Input Data Object Element

Parameter name	service specific elements	rsp	cnf
Data	Length IOCS	M	M(=)
	IOCS	M	M(=)
	Length IOPS	M	M(=)
	IOPS	M	M(=)
	Length Data	M	M(=)
	Data	M	M(=)

Data

This parameter contains the values of the object which has been read.

Length IOCS

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

IOCS

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

Length IOPS

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.

Length Data

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

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 Data. Its content is composed according to the attribute Input Data Structure of the Input Data object. Data contains the input data as given by the IO device.

7.3.2.2.5.11 Read Record Output Data Object Element

This confirmed service allows reading the value of an Output Data object element with the substitute value.

The output data object allows applying a substitute value as seen in Figure 31.

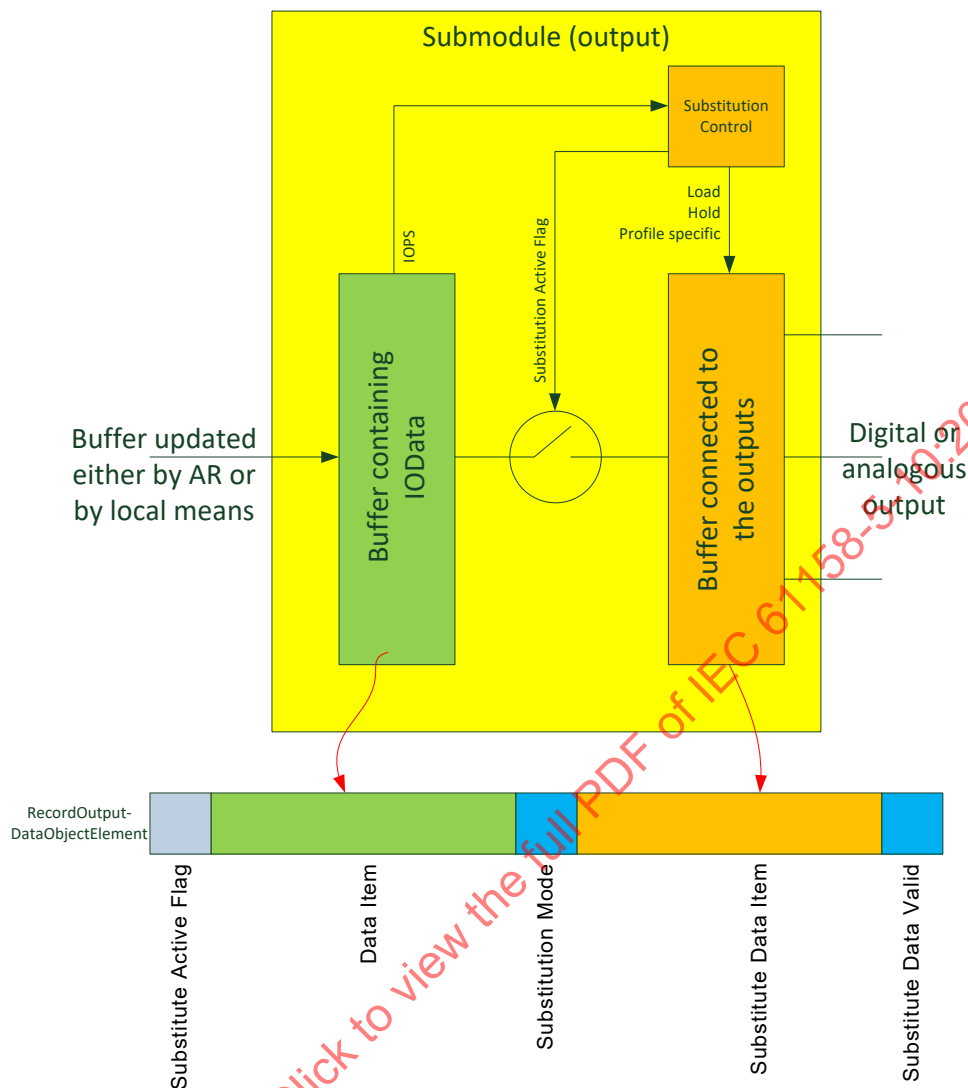


Figure 31 – Substitute Value

The substitute value is defined by one of the following substitution modes:

- ZERO – is a submodule specific function to set all output values of the Output Data element to inactive (e.g. free from voltage or current),
- 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 Substitution Mode (e.g. “0”),
- PROFILE_SPECIFIC – all output values of the Output Data element shall be set according to a specific profile.

The substitution mode is defined using the service Write Substitute Value (see 7.3.2.2.5.13), by applying a vendor specific data record or by local means.

The substitution control controls the buffer content connected to the outputs depending on the IOPS of the output data object.

If IOPS is set to GOOD

- the Substitute Active Flag is set to OPERATION, meaning the outputs are set according to normal control process (green drawn buffer).
- the output data buffer updated by the AR being the owner of the submodule is applied to the buffer connected to the outputs.
- the service Read Substitute Value returns
 - Substitute Active Flag = OPERATION
 - Data Item contains the output data buffer updated by the AR
 - Substitution Mode contains the configured substitution mode
 - If Substitute Data Valid == GOOD
 - Substitute Data Item contains the buffer connected to the outputs (preferred solution)
 - If Substitute Data Valid == BAD_BY_SUBSLOT
 - Substitute Data Item is invalid

NOTE 1 The buffer connected to the outputs is already delivered with the Data Item.

If IOPS is set to BAD_BY_XXX

- the Substitute Active Flag is set to SUBSTITUTE, meaning the outputs are set according to the substitution mode.
- the output data buffer updated by the AR being the owner of the submodule is disconnected from the buffer connected to the outputs; the buffer to the outputs is filled by the substitute control according to the substitution mode (zero, last value, replacement value, or profile specific).
- the service Read Substitute Value returns
 - Substitute Active Flag = SUBSTITUTE
 - Data Item contains the output data buffer updated by the AR
 - Substitution Mode contains the configured substitution mode
 - Substitute Data Valid = GOOD
 - if Substitution Mode is set to ZERO, Substitute Data Item should contain the buffer connected to the outputs.
 - if Substitution Mode is set to another value than ZERO, Substitute Data Item shall contain the buffer connected to the outputs.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1). But the parameter Data has a service specific structure and meaning. Table 179 shows the specific elements of Data for this service.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

This service returns “access / invalid index” for submodules without output data; i.e. for submodules having input data only, or having no IO data.

If the submodule is owned by the AR specified in the service (over the AREP resp. the Target AR UUID), the output data object element as provided/seen within this AR is returned. If the owner sees OK or SUBSTITUTE, the output data is returned accordingly (i.e. output data, IOPS from the IO controller owning the submodule, IOCS from the device), if the owner sees WRONG or NO_SUBMODULE, the service returns “access / invalid index”.

If the submodule is not owned by the AR, the service returns the real data of the submodule. If the real data cannot be determined by the application, the service returns “access / invalid index”.

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.

Table 179 – Data elements of Read Record Output Data Object Element

Parameter name	service specific elements	rsp	cnf
Data	Substitute Active Flag	M	M(=)
	Length IOCS	M	M(=)
	Length IOPS	M	M(=)
	Length Data	M	M(=)
	Data Item	M	M(=)
	IOCS	M	M(=)
	Data	M	M(=)
	IOPS	M	M(=)
	Substitute Value	M	M(=)
	Substitution Mode	M	M(=)
	Substitute Data Item	M	M(=)
	IOCS	M	M(=)
	Data	M	M(=)
	Substitute Data Valid	M	M(=)

Data

This parameter contains the values of the object which has been read.

Substitute Active Flag

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 IOPS

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

Length Data

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

Data Item

This parameter is composed of the following elements:

IOCS

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

Data

This parameter contains the value of the object which has been read and consists of the number of octets indicated in the Length Data of the response. Its content is composed according to the attribute Output Data Structure of the Output Data object. Data contains the output data as given by the IO controller.

IOPS

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

Substitute Value

This parameter is composed of the following elements:

Substitution Mode

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

Substitute Data Item

This parameter is composed of the following elements:

IOCS

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

Data

This parameter contains the value of the object which has been read and consists of the number of octets indicated in the Length Data of the response. Its content is composed according to the attribute Output Data Object Format of the Output Data object.

For the content of the data, refer to the explanation above.

Substitute Data Valid

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

7.3.2.2.5.12 Read Substitute Value

This confirmed and optional service allows reading the substitute value.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1). But the parameter Data has a service specific structure and meaning. Table 180 shows the specific elements of Data for this service.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

This service returns “access / invalid index” if the application does not support reading of values of Substitute Value objects or if the data of the Input Data object cannot be determined by the application.

Table 180 – Data elements of Read Substitute Value

Parameter name	service specific elements	rsp	cnf
Data	Substitute Value	M	M(=)
	Substitution Mode	M	M(=)
	Substitute Data Item	M	M(=)
	IOCS	M	M(=)
	Data	M	M(=)
	Substitute Data Valid	M	M(=)

Data

This parameter contains the values of the object which has been read.

Substitute Value

This parameter is composed of the following elements:

Substitution Mode

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

Substitute Data Item

This parameter is composed of the following elements:

IOCS

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

Data

This parameter contains the Substitute Value which has been read and consists of the number of octets indicated in the Length Data of the response. Its content is composed according to the attribute Output Data Object Format of the Output Data object.

Substitute Data Valid

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

If the submodule does not support Substitute Value either a negative response containing the values Error Decode = "PNIORW", Error code 1 = "invalid index", Error code 2 = "user specific" may be given or else a null record may be returned.

7.3.2.2.5.13 Write Substitute Value

This confirmed and optional service allows writing the substitute value of an Output Data object element.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1). But the parameter Data has a service specific structure and meaning. Table 180 shows the specific elements of Data for this service.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

This service returns "access / invalid index" if the application does not support writing of values of Substitute Value objects or if the data of the Input Data object cannot be determined by the application.

Data

See Read Substitute Value (7.3.2.2.5.12).

7.3.2.2.5.14 Read GSD Data

This confirmed and optional service allows reading GSD Data using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read query service (see 7.3.1.6.5.1.3); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see 7.3.4.2.5.1 for its general description.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

GSD data belongs to the device. The parameter Index identifies the data record index as specified in IEC 61158-6-10.

The model to upload the GSD is described as follows:

- The uploader starts with From Offset Data set to “0” (i.e. start uploading at the begin) and specifies in the Read Query request the maximum length to read in one chunk via the parameter Length
- The IO device responds with Data set according to Table 181; mirrors the requested From Offset Data, specifies for the uploader which value to use for the next call to this service as Next Offset Data, and informs the uploader about the Total Size. The Data returned is a chunk from the ASE’s attribute GSD Data (starting with GSD Data[From Offset Data] up to GSD Data[Next Offset Data]).
- If Next Offset Data is returned with a value “0”, the last chunk of the GSD Data has been read

Table 181 shows the specific elements of Selector for this service.

Table 181 – Selector for Read GSD Data

Parameter name	service specific elements	rsp	cnf
Selector	From Offset Data	M	M(=)

Data

This parameter contains the value of the object which has been read.

From Offset Data

Specifies the start offset for the GSD data.

Attribute type: Unsigned32

Table 182 shows the specific elements of Data for this service.

Table 182 – Data elements of Read GSD Data

Parameter name	service specific elements	rsp	cnf
Data	From Offset Data	M	M(=)
	Next Offset Data	M	M(=)
	Total Size	M	M(=)
	Data	M	M(=)

Data

This parameter contains the value of the object which has been read.

From Offset Data

Mirrors the From Offset Data of the request

Attribute type: Unsigned32

Next Offset Data

Specifies the next offset to read the GSD data

Attribute type: Unsigned32

Special case: 0, if last chunk of GSD data has been transmitted

Total Size

Specifies the total size of the GSD data

Attribute type: Unsigned32

Data

Chunk of GSD Data

7.3.2.2.6 Behavior

7.3.2.2.6.1 General

The real identification object contains the real identification data from the perspective of the IO device. The real identification is detected by local means of the IO device and kept always “up to date”, i.e. it reflects the current configuration of the IO device. The real identification may be changed during operation by local means, such as plug or pull of modules or submodules or ownership handling (takeover, release...). In such a case, the real identification shall be updated appropriately using Local Add / Remove / Update Submodule.

If an AR is affected by a change in the real identification, the IO device shall inform the IO controller/supervisor by means of an Alarm service. The Alarm Types “Controlled By Supervisor”, “Released”, “Pull”, “Plug” or “Plug Wrong Submodule” shall be used to indicate such events.

When receiving the Alarm Notification with Alarm type “Plug” the IO controller/supervisor shall check if the new added submodule needs startup parameter and convey this parameter to the IO device.

When receiving the Alarm Notification with Alarm type “Pull” or “Plug Wrong Submodule” the IO controller may inform the local user.

7.3.2.2.6.2 Behavior in case of configuration change

7.3.2.2.6.2.1 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.

7.3.2.2.6.2.2 Real Submodule

7.3.2.2.6.2.2.1 Primitive definitions

7.3.2.2.6.2.2.2 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Real Submodule State Machine (RSM) are described in the service definition and shown in Table 183.

Table 183 – Remote primitives issued or received by RSMSM

Primitive	Source	Destination	Associated parameters	Functions
UserPlugSM.req	local	RSMSM	RealModuleID, RealSubmoduleID, API, Slot, Subslot	—
UserPullSM.req	local	RSMSM	API, Slot, Subslot	—
UserPlugSM.cnf ()	RSMSM	local		—
UserPullSM.cnf ()	RSMSM	local		—
PlugAlarm.req	RSMSM	PLUGSM	RealModuleID, RealSubmoduleID, API, Slot, Subslot	—
PlugAlarm.cnf ()	PLUGSM	RSMSM	API, Slot, Subslot	—
PlugWrongAlarm.req	RSMSM	PLUGSM	RealModuleID, RealSubmoduleID, API, Slot, Subslot	—
PlugWrongAlarm.cnf ()	PLUGSM	RSMSM	API, Slot, Subslot	—
PullAlarm.req	RSMSM	PULLSM	API, Slot, Subslot	—
PullAlarm.cnf ()	PULLSM	RSMSM	API, Slot, Subslot	—

7.3.2.2.6.2.2.3 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by RSMSM described in the service definition and shown in Table 184.

Table 184 – Local primitives issued or received by RSMSM

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

7.3.2.2.6.2.2.4 State transition diagram

The state transition diagram is shown in Figure 32.

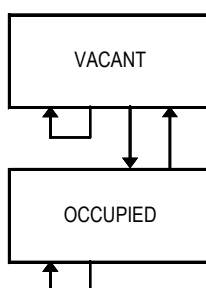


Figure 32 – State transition diagram RSMSM

States of the RSMSM

VACANT

The addressed API, Slot, Subslot is vacant. This means no submodule is plugged or assigned.

OCCUPIED

The addressed API, Slot, Subslot is occupied. This means a submodule is plugged or assigned.

7.3.2.2.6.2.2.5 State machine description

This state machine covers for a particular subslot its occupation by a submodule and raises the appropriate Pull and Plug alarms. Note that the Plug alarm is not issued, when a wrong submodule is plugged into the subslot, if no current owner exists, or if the current owner is the application (SOL).

7.3.2.2.6.2.2.6 State table

Table 185 defines the state table for the selected behavior.

Table 185 – State table RSMSM

#	Current State	Event /Condition =>Action	Next State
1	VACANT	UserPlugSM.req () /GetOwnerFromAdministration () == FREE GetOwnerFromAdministration () == SOL => ignore UserPlugSM.cnf (+)	OCCUPIED
2	VACANT	UserPlugSM.req () /GetOwnerFromAdministration () == OWNER && UserCheckIdent () != WRONG => Create PlugSubmodule alarm PDU for the OWNER PlugAlarm.req () UserPlugSM.cnf (+)	OCCUPIED
3	VACANT	UserPlugSM.req () /GetOwnerFromAdministration () == OWNER && UserCheckIdent () == WRONG => Create PlugWrongSubmodule alarm PDU for the OWNER PlugWrongAlarm.req () UserPlugSM.cnf (+)	OCCUPIED

#	Current State	Event /Condition =>Action	Next State
4	VACANT	UserPullISM.req () => ignore UserPullISM.cnf (+)	VACANT
5	VACANT	PlugAlarm.cnf () => ignore	VACANT
6	VACANT	PlugWrongAlarm.cnf () => ignore	VACANT
7	VACANT	PullAlarm.cnf () => ignore	VACANT
8	OCCUPIED	UserPullISM.req () /GetOwnerFromAdministration () == FREE GetOwnerFromAdministration () == SOL => ignore UserPullISM.cnf (+)	VACANT
9	OCCUPIED	UserPullISM.req () /GetOwnerFromAdministration () == OWNER => Create PullSubmodule alarm PDU for the OWNER PullAlarm.req () UserPullISM.cnf (+)	VACANT
10	OCCUPIED	UserPlugSM.req () => ignore UserPlugSM.cnf (-)	OCCUPIED
11	OCCUPIED	PlugAlarm.cnf () => ignore	OCCUPIED
12	OCCUPIED	PlugWrongAlarm.cnf () => ignore	OCCUPIED
13	OCCUPIED	PullAlarm.cnf () => ignore	OCCUPIED

7.3.2.2.6.2.2.7 Functions, Macros, Timers and Variables

Table 186 contains the functions, macros, timers and variables used by RSMSM and their arguments and descriptions.

Table 186 – Functions, Macros, Timers and Variables used by RSMSM

Name	Type	Function/Meaning
UserCheckIdent	Function	UserCheckIdent(API, Slot, Subslot, RealModuleID, RealSubmoduleID, ExpModuleID, ExpSubmoduleID) Returns the ident information for the expected submodule: OK: is returned, if a submodule is plugged at the specified location (API, slot and subslot) and it's the right one; i.e. its real ident information (RealModuleID, RealSubmoduleID) is equivalent to the expected ident information (ExpModuleID and ExpSubmoduleID). SUBSTITUTE: is returned, if a submodule is plugged at the specified location (API, slot and subslot) and is able to be a decent substitute; i.e. its real ident information (RealModuleID, RealSubmoduleID) is able to substitute the expected ident information (ExpModuleID and ExpSubmoduleID). WRONG: is returned, if a submodule is plugged at the specified location (API, slot and subslot) and it's wrong; i.e. if its real ident information (RealModuleID, RealSubmoduleID) does not match the expected ident information (ExpModuleID and ExpSubmoduleID), if it doesn't support the expected API. NOSUBMODULE: is returned, if no submodule is plugged at the location specified by API, slot and subslot.
GetOwnerFromAdministration	Function	This local function returns the owner of the submodule. FREE: No owner OWNER: The asking AREP is the Owner SOL: The submodule is superordinated locked

7.3.2.2.6.2.2.8 Truth Table

Table 187 contains the truth table used by RSMSM.

Table 187 – Rules for Submodule State.Ident Info

Input	Output - Owner	Output - Not owner
Result of UserCheckIdent()	Submodule State.Ident Info	Submodule State.Ident Info
OK	OK	OK
SUBSTITUTE	SUBSTITUTE	WRONG
WRONG	WRONG	WRONG
NOSUBMODULE	NOSUBMODULE	NOSUBMODULE

7.3.2.2.6.2.3 Ownership and Association of a Submodule

7.3.2.2.6.2.3.1 Common

For the access to a submodule within a particular AR there exist the following “qualities”:

- If the AR is the owner of the submodule, the AR has
 - read access to input data,
 - write access to output data,
 - read access to data records,
 - write access to data records, if being primary or first,
 - and is allowed to create or receive every kind of alarm notification.

- If the AR is associate of the submodule, the AR has
 - read access to input data,
 - read access to data records,
 - and is allowed to create or receive the Return Of Submodule alarm notification.
- A device access AR has
 - read access to data records,
 - and, where the application allows, write access (independent of ownership).
- An implicit AR has
 - read access to data records.

IECNORM.COM : Click to view the full PDF of IEC 61158-5-10:2019

The handling of the ownership for a particular submodule is described in Figure 33:

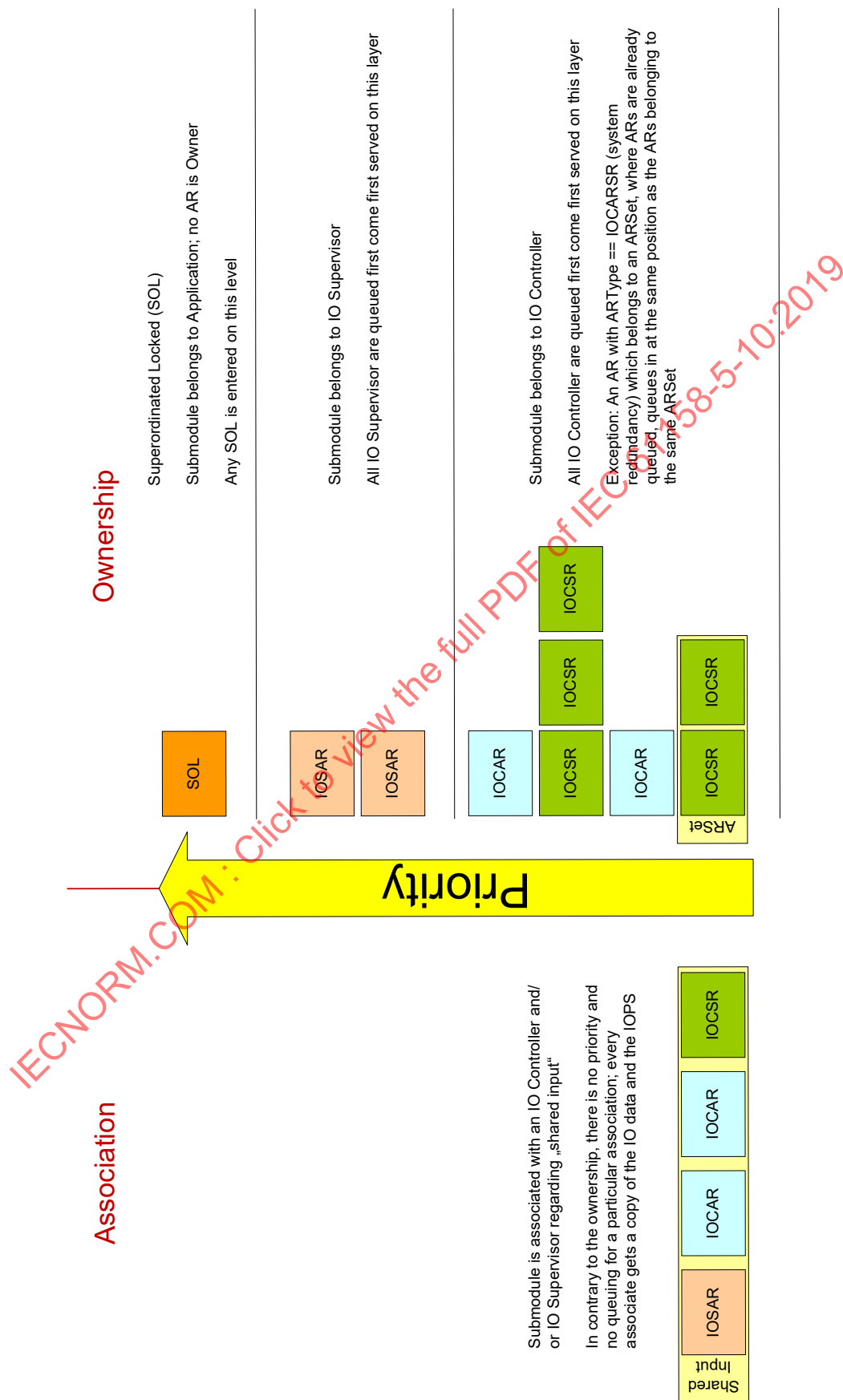


Figure 33 – Ownership handling

The ownership is based on the following principles:

- The ownership of a submodule is in principle handled “first come first served”. Thus the first application relation wanting the submodule will get the ownership; any further application relation will get a “Locked” and is queued at the end.
- The user application is able to lock the submodule for superordinated use at any time. The current owner of the submodule is informed by a “Controlled” alarm.
- If the current owner’s application relation allows a supervisor takeover (as indicated by the ARProperties), an application relation with ARTYPE IOSAR is able to takeover the submodule from its current owner. This application relation is set as the queue head and thus gets the ownership; the now previous owner is then informed by a “Controlled” alarm.
- If the current owner’s application relation doesn’t allow a supervisor takeover (as indicated by the ARProperties), an application relation with ARTYPE IOSAR is not able to takeover the submodule from its current owner. Thus the IO supervisor application relation is queued after the head but after the last waiting IO supervisor AR.
- An application relation with ARTYPE IOCARSR of an already known AR Set (as indicated by the application relations ARUUID; details see IEC 61158-6-10/) is queued in at the same location as the other ARs of this AR Set. Note that all application relations of the AR Set have the same view on the ownership as expressed by the Module Diff Block.
- If the current owner goes, the ownership is handed over to the next application relation waiting in the queue. If the new owner is an application relation with ARTYPE != IOCARSR, a Released alarm is issued to indicate the hand over. If the ARTYPE is IOCARSR and the new owner is in the same AR Set as the previous owner, no Released alarm is issued.
- The ownership is handled in the OWNSM.

The association is based on the following principles:

- The association of a submodule is handled without any priority. All requesting ARs have the same priority; there is no queuing.
- The reason to associate a submodule is to have read access using an I-CR to the input data of a submodule from another IO controller / IO supervisor. This is also called “shared input”.
- Since the validity of the input data is determined by the IOPS, the Return of Submodule is a necessary alarm for an IO controller using an association.
- Since no alarms are allowed on an association beside the Return Of Submodule alarm, an associate has only information about the availability of the input data. Other information like the submodule’s presence (pull, plug, plug wrong), its parameterization state (application ready pending) or its diagnosis of the submodule is not available, since it will not be actualized via some alarm. Thus the Submodule State shall contains in any Module Diff Block for a shared input submodule
 - for the Ident Info (indicating its presence) the value OK,
 - for Advice (indicating its advice information) the value NO_ADVICE,
 - for Maintenance Required (indicating its maintenance required information) the value NO_MAINTENANCE_REQUIRED,
 - for Maintenance Demanded (indicating its maintenance demanded information) the value NO_MAINTENANCE_DEMANDED,
 - for Fault (indicating its diagnosis information) the value NO_FAULT,
 - for ARInfo (indicating ownership problems) the value OWN,
 independent of the submodule’s state. In other words this information is only available for the application having the ownership on the submodule. The rules for setting the IOPS to BAD e.g. in case of a pulled or a wrongly plugged submodule (see 7.3.1.3.5) will take care that the submodule’s input data is not useable in such a case.

- The association is handled in the ASSSM.

If the submodule is not usable by the next one in the ownership queue due to being the wrong submodule (WRONG) plugged as waited for or having no submodule plugged at the subslot (NOSUBMODULE). In this case no Released alarm is sent to the new owner as an optimization (“hidden ownership change”), because the submodule is still not usable for him. Note however that all further alarms regarding this submodule (e.g. a Plug alarm) are sent to the new owner.

Thus an IO controller implementation shall accept alarms for submodules even for those he is not owning from his current point of view.

7.3.2.2.6.2.3.2 Primitive definitions

7.3.2.2.6.2.3.3 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by Ownership Submodule State Machine (OWNSM) and Association Submodule State Machine (ASSSM) are described in the service definition and shown in Table 188.

Table 188 – Remote primitives issued or received by OWNSM and ASSSM

Primitive	Source	Destination	Associated parameters	Functions
Connect.req	FAL	OWNSM, ASSSM	see 7.3.1.2.5.1	—
Read.req	FAL	OWNSM, ASSSM	see 7.3.1.6.5.1.1	—
Release.req	FAL	OWNSM, ASSSM	see 7.3.1.2.5.2	—
Write.req	FAL	OWNSM, ASSSM	see 7.3.1.6.5.2.1	—
UserSetSMState.cnf	local	OWNSM	API, Slot, Subslot	—
UserSetSMState.req	local	OWNSM	Command, API, Slot, Subslot	—
Read.rsp	OWNSM, ASSSM	FAL	see 7.3.1.6.5.1.1	—
Write.rsp	OWNSM, ASSSM	FAL	see 7.3.1.6.5.2.1	—
ReleasedAlarm.req	OWNSM	PLUGSM	ANY.AREP, ANY.API, ANY.Slot, ANY.Subslot	—
ControlledAlarm.req	OWNSM	PULLSM	ANY.AREP, ANY.API, ANY.Slot, ANY.Subslot	—
ReleasedAlarm.cnf	PLUGSM	OWNSM	ANY.AREP, ANY.API, ANY.Slot, ANY.Subslot	—
ControlledAlarm.cnf	PULLSM	OWNSM	ANY.AREP, ANY.API, ANY.Slot, ANY.Subslot	—

7.3.2.2.6.2.3.4 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by OWNSM are described in the service definition and shown in Table 189.

Table 189 – Local primitives issued or received by OWNSM

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

7.3.2.2.6.2.3.5 State transition diagram

7.3.2.2.6.2.3.6 OWNSM

The state transition diagram is shown in Figure 32.

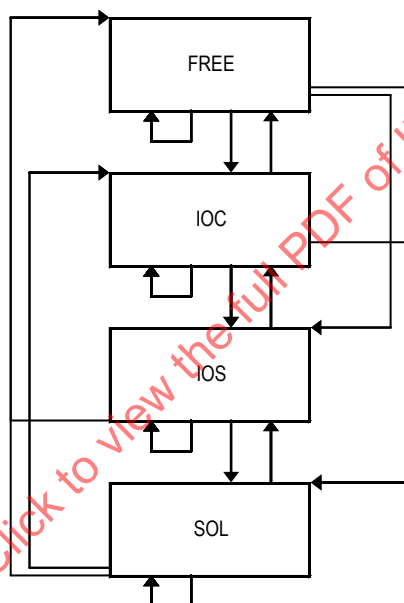


Figure 34 – State transition diagram OWNSM

States of the OWNSM

FREE	The submodule has no owner
SOL	The submodule is superordinated locked
IOS	The submodule is owned by an IO supervisor
IOC	The submodule is owned by an IO controller

7.3.2.2.6.2.3.7 ASSSM

The state transition diagram is shown in Figure 35.

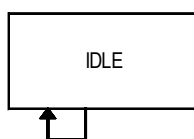


Figure 35 – State transition diagram ASSSM

States of the ASSSM

IDLE

The submodule is waiting for an association

7.3.2.2.6.2.3.8 State machine description

7.3.2.2.6.2.3.9 OWNSM

The state machine treats the ownership of a particular submodule. The handover of the ownership to an IO supervisor or to the application (superordinated locked) is accompanied by the Controlled By Supervisor alarm, the release by the Released alarm.

The state machine handles the ownership queue (i.e. those application relations waiting in line for the particular submodule) and the flag SupervisorTakeoverAllowed.

It additionally filters write requests, if issued via a different application relation than the current owner.

7.3.2.2.6.2.3.10 ASSSM

The state machine treats the association of a particular submodule.

7.3.2.2.6.2.3.11 State table

Table 190 defines the state table for the selected behavior of the OWNSM. This state table shall be used if Submodule Properties.Shared Input is set to 0x00.

Table 191 defines the state table for the selected behavior of the ASSSM. This state table shall be used if Submodule Properties.Shared Input is set to 0x01.

Table 190 – State table OWNSM

#	Current State	Event /Condition =>Action	Next State
1	FREE	Connect.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	FREE
2	FREE	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType != IOSAR => AddToOwnershipAdministration ()	IOC
3	FREE	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType == IOSAR => AddToOwnershipAdministration ()	IOS

#	Current State	Event /Condition =>Action	Next State
4	FREE	UserSetSMState.req () /Command == SOL => ignore UserSetSMState.cnf ()	SOL
5	FREE	UserSetSMState.req () /Command != SOL => ignore UserSetSMState.cnf ()	FREE
6	FREE	Release.req () => ignore	FREE
7	FREE	ReleasedAlarm.cnf () => ignore	FREE
8	FREE	ControlledAlarm.cnf () => ignore	FREE
9	FREE	Write.ind () => ErrorDecode := PNORW ErrorCode1 := access denied Write.rsp (-)	FREE
10	FREE	Read.ind () => Fetch data for read response Read.rsp () //NOTE Depending whether the user is able to return data (+) or (-) is returned	FREE
11	IOC	Connect.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	IOC
12	IOC	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType != IOSAR => AddToOwnershipAdministration ()	IOC
13	IOC	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType == IOSAR && Owner.ARProperties.SupervisorTakeOverAllowed == TRUE => AddToOwnershipAdministration () Create ControlledBySupervisor alarm PDU to be sent to the now previous owner ControlledAlarm.req ()	IOS
14	IOC	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType == IOSAR && Owner.ARProperties.SupervisorTakeOverAllowed == FALSE => AddToOwnershipAdministration ()	IOC
15	IOC	UserSetSMState.req () /Command == SOL => AddToOwnershipAdministration ()	SOL

#	Current State	Event /Condition =>Action	Next State
		Create ControlledBySupervisor alarm PDU to be sent to the now previous owner ControlledAlarm.req () UserSetSMState.cnf ()	
16	IOC	UserSetSMState.req () /Command != SOL => ignore UserSetSMState.cnf ()	IOC
17	IOC	Release.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	IOC
18	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == FREE => ignore	FREE
19	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOC && (UserCheckIdent () == WRONG UserCheckIdent () == NOSUBMODULE) => ignore //NOTE No Released alarm is sent, because the submodule is still not usable for the new owner.	IOC
20	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOC && (UserCheckIdent () == OK UserCheckIdent () == SUBSTITUTE) => Create Released alarm PDU to be sent to the new owner ReleasedAlarm.req ()	IOC
21	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOS && (UserCheckIdent () == WRONG UserCheckIdent () == NOSUBMODULE) //NOTE IOS was blocked by TakeoverNotAllowed => ignore //NOTE No Released alarm is sent, because the submodule is still not usable for the new owner.	IOS
22	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOS && (UserCheckIdent () == OK UserCheckIdent () == SUBSTITUTE) //NOTE IOS was blocked by TakeoverNotAllowed => Create Released alarm PDU to be sent to the new owner ReleasedAlarm.req ()	IOS
23	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOC, ARset =>	IOC

#	Current State	Event /Condition =>Action	Next State
		ignore //NOTE SystemRedundancy case	
24	IOC	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == UNCHANGED => ignore	IOC
25	IOC	ReleasedAlarm.cnf () => ignore	IOC
26	IOC	ControlledAlarm.cnf () => ignore	IOC
27	IOC	Write.ind () /AREP == GetCurrentOwner () //NOTE Accept write only from the current owner => Give data for write to user and fetch status for write response Write.rsp () //NOTE Depending whether the user is able to handle data (+) or (-) is returned	IOC
28	IOC	Write.ind () /AREP != GetCurrentOwner () //NOTE Accept write only from the current owner => ErrorDecode := PNORW ErrorCode1 := access denied Write.rsp (-)	IOC
29	IOC	Read.ind () => Fetch data for read response Read.rsp () //NOTE Depending whether the user is able to return data (+) or (-) is returned	IOC
30	IOS	Connect.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	IOS
31	IOS	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType != IOSAR => AddToOwnershipAdministration ()	IOS
32	IOS	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock && ARType == IOSAR => AddToOwnershipAdministration ()	IOS
33	IOS	UserSetSMState.req () /Command == SOL => AddToOwnershipAdministration () Create ControlledBySupervisor alarm PDU to be sent to the now previous owner ControlledAlarm.req () UserSetSMState.cnf ()	SOL
34	IOS	UserSetSMState.req () /Command != SOL =>	IOS

#	Current State	Event /Condition =>Action	Next State
		ignore UserSetSMState.cnf ()	
35	IOS	Release.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	IOS
36	IOS	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == FREE => ignore	FREE
37	IOS	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOC && (UserCheckIdent () == WRONG UserCheckIdent () == NOSUBMODULE) => ignore //NOTE No Released alarm is sent, because the submodule is still not usable for the new owner.	IOC
38	IOS	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOC && (UserCheckIdent () == OK UserCheckIdent () == SUBSTITUTE) => Create Released alarm PDU to be sent to the new owner ReleasedAlarm.req ()	IOC
39	IOS	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOS && (UserCheckIdent () == WRONG UserCheckIdent () == NOSUBMODULE) => ignore //NOTE No Released alarm is sent, because the submodule is still not usable for the new owner.	IOS
40	IOS	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == IOS && (UserCheckIdent () == OK UserCheckIdent () == SUBSTITUTE) => Create Released alarm PDU to be sent to the new owner ReleasedAlarm.req ()	IOS
41	IOS	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock && RemoveAndGetOwnerFromAdministration () == UNCHANGED => ignore	IOS
42	IOS	ReleasedAlarm.cnf () => ignore	IOS
43	IOS	ControlledAlarm.cnf () => ignore	IOS
44	IOS	Write.ind () /AREP == GetCurrentOwner ()	IOS

#	Current State	Event /Condition =>Action	Next State
		//NOTE Accept write only from the current owner => Give data for write to user and fetch status for write response Write.rsp () //NOTE Depending whether the user is able to handle data (+) or (-) is returned	
45	IOS	Write.ind () /AREP != GetCurrentOwner () //NOTE Accept write only from the current owner => ErrorDecode := PNORW ErrorCode1 := access denied Write.rsp (-)	IOS
46	IOS	Read.ind () => Fetch data for read response Read.rsp () //NOTE Depending whether the user is able to return data (+) or (-) is returned	IOS
47	SOL	Connect.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	SOL
48	SOL	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock => AddToOwnershipAdministration ()	SOL
49	SOL	UserSetSMState.req () /Command == SOL => ignore UserSetSMState.cnf ()	SOL
50	SOL	UserSetSMState.req () /Command != SOL && RemoveAndGetOwnerFromAdministration () == FREE => ignore UserSetSMState.cnf ()	FREE
51	SOL	UserSetSMState.req () /Command != SOL && RemoveAndGetOwnerFromAdministration () == IOC && (UserCheckIdent () == WRONG UserCheckIdent () == NOSUBMODULE) => //NOTE No Released alarm is sent, because the submodule is still not usable for the new owner. UserSetSMState.cnf ()	IOC
52	SOL	UserSetSMState.req () /Command != SOL && RemoveAndGetOwnerFromAdministration () == IOC && (UserCheckIdent () == OK UserCheckIdent () == SUBSTITUTE) => Create Released alarm PDU to be sent to the new owner ReleasedAlarm.req () UserSetSMState.cnf ()	IOC
53	SOL	UserSetSMState.req () /Command != SOL && RemoveAndGetOwnerFromAdministration () == IOS &&	IOS

#	Current State	Event /Condition =>Action	Next State
		(UserCheckIdent () == WRONG UserCheckIdent () == NOSUBMODULE) => //NOTE No Released alarm is sent, because the submodule is still not usable for the new owner. UserSetSMState.cnf ()	
54	SOL	UserSetSMState.req () /Command != SOL && RemoveAndGetOwnerFromAdministration () == IOS && (UserCheckIdent () == OK UserCheckIdent () == SUBSTITUTE) => Create Released alarm PDU to be sent to the new owner ReleasedAlarm.req () UserSetSMState.cnf ()	IOS
55	SOL	Release.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	SOL
56	SOL	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock => RemoveAndGetOwnerFromAdministration () //NOTE In this case only the ownership administration is updated	SOL
57	SOL	ReleasedAlarm.cnf () => ignore	SOL
58	SOL	ControlledAlarm.cnf () => ignore	SOL
59	SOL	Write.ind () => ErrorDecode := PNORW ErrorCode1 := access denied Write.rsp (-)	SOL
60	SOL	Read.ind () => Fetch data for read response Read.rsp () //NOTE Depending whether the user is able to return data (+) or (-) is returned	SOL

Table 191 – State table ASSSM

#	Current State	Event /Condition =>Action	Next State
1	IDLE	Connect.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	IDLE
2	IDLE	Connect.req () /Submodule is in list of the ExpectedSubmoduleBlock => AddToAssociation ()	IDLE
3	IDLE	Release.req () /Submodule is not in list of the ExpectedSubmoduleBlock => ignore	IDLE
4	IDLE	Release.req () /Submodule is in list of the ExpectedSubmoduleBlock => RemoveFromAssociation ()	IDLE
5	IDLE	Write.ind () => ErrorDecode := PNORW ErrorCode1 := access denied Write.rsp (-)	IDLE
6	IDLE	Read.ind () => Fetch data for read response Read.rsp () //NOTE Depending whether the user is able to return data (+) or (-) is returned	IDLE

7.3.2.2.6.2.3.12 Functions, Macros, Timers and Variables

Table 192 contains the functions, macros, timers and variables used by OWNSM and their arguments and descriptions.

Table 192 – Functions, Macros, Timers and Variables used by OWNSM

Name	Type	Function/Meaning
RemoveAndGetOwnerFromAdministration	Function	This local function removes the given AREP from the ownership administration. It returns whether/where to the ownership changed and the AREP of the application relation now owning the submodule. The ownership is primarily handled by the first come first served principle. This means that the first connect.req (first AR) requesting a "free" submodule will get the ownership. The exceptions to this principle are higher priority ownership request by the IO supervisor or local means (Superordinated lock) or by the TakeoverNotAllowed property of a IOC-AR. Principle: A supervisor may take away a submodule from an IO controller and local means may take away a submodule from an IO supervisor or IO controller. FREE This submodule has no longer an owner IOC The ownership of the submodule is transferred to an IOC IOC, ARset The ownership of the submodule is transferred to an IOC within the same ARset as the previous owner IOS The ownership of the submodule is transferred to an IOS UNCHANGED The ownership of the submodule did not change; i.e. the AREP removed was not the current owner
GetCurrentOwner	Function	This local function returns the AREP of the submodule's current owner
AddToOwnershipAdministration	Function	This local function adds the new owner or a candidate, addressed by the AREP, to the ownership administration.

Name	Type	Function/Meaning
UserCheckIdent	Function	This local function asks the User to check the expected ModuleID and SubmoduleID against the real one. Returns the ident information for the expected submodule: OK: is returned, if a submodule is plugged at the specified location (API, slot and subslot) and it's the right one; i.e. its real ident information (RealModuleID, RealSubmoduleID) is equivalent to the expected ident information (ExpModuleID and ExpSubmoduleID). SUBSTITUTE: is returned, if a submodule is plugged at the specified location (API, slot and subslot) and is able to be a decent substitute; i.e. its real ident information (RealModuleID, RealSubmoduleID) is able to substitute the expected ident information (ExpModuleID and ExpSubmoduleID). WRONG: is returned, if a submodule is plugged at the specified location (API, slot and subslot) and it's wrong; i.e. if its real ident information (RealModuleID, RealSubmoduleID) does not match the expected ident information (ExpModuleID and ExpSubmoduleID). NOSUBMODULE: is returned, if no submodule is plugged at the location specified by API, slot and subslot.

Table 193 – Functions, Macros, Timers and Variables used by ASSSM

Name	Type	Function/Meaning
AddToAssociation	Function	This local function adds the new associate, addressed by the AREP, to the association. Adding a submodule to the association shall not create an entry in the module diff block.
RemoveFromAssociation	Function	This local function removes the new associate, addressed by the AREP, to the association.

7.3.2.2.6.2.3.13 Truth Table

Table 194 contains the truth table used by OWNSM.

Table 194 – Rules for Submodule State.AR Info

Input		Output
Current Owner	Parameter Check Pending	Submodule State.AR Info
OWN Owner is in same AR Set	No	OWN
OWN Owner is in same AR Set	Yes	APPLICATION_READY_PENDING
Superordinated Locked	-	SUPERORDINATED_LOCKED
Other IO controller	-	LOCKED_BY_IO_CONTROLLER
Other IO Supervisor	-	LOCKED_BY_IO_SUPERVISOR

7.3.2.2.6.2.4 Plug

7.3.2.2.6.2.4.1 Primitive definitions

7.3.2.2.6.2.4.2 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by plug state machine (PLUGSM) are described in the service definition and shown in Table 195.

Table 195 – Remote primitives issued or received by PLUGSM

Primitive	Source	Destination	Associated parameters	Functions
Alarm.cnf ()	FAL	PLUGSM	see 7.3.1.5.5	—
ApplicationReady.cnf ()	FAL	PLUGSM	see 7.3.1.2.5.5	—
PrmEnd.ind	FAL	PLUGSM	see 7.3.1.2.5.4	—
Write.ind	FAL	PLUGSM	see 7.3.1.6.5.2.1	—
ReturnOfSubmodulAlarm.req	local	PLUGSM	AREP, API, Slot, Subslot, UserInfo;	—
UserControl.rsp ()	local	PLUGSM	AREP, PrmEnd	—
UserWriteRecord.rsp ()	local	PLUGSM		—
ReleasedAlarm.req	OWNSM	PLUGSM	AREP, API, Slot, Subslot, UserInfo;	—
Alarm.req	PLUGSM	FAL	see 7.3.1.5.5	—
ApplicationReady.req	PLUGSM	FAL	see 7.3.1.2.5.5	—
PrmEnd.rsp ()	PLUGSM	FAL	see 7.3.1.2.5.4	—
Write.rsp ()	PLUGSM	FAL	see 7.3.1.6.5.2.1	—
ReturnOfSubmodulAlarm.cnf ()	PLUGSM	local	AREP, API, Slot, Subslot, UserInfo;	—
UserControl.ind	PLUGSM	local	AREP, PrmEnd	—
UserWriteRecord.ind	PLUGSM	local		—
ReleasedAlarm.cnf ()	PLUGSM	OWNSM	AREP, API, Slot, Subslot, UserInfo;	—
PlugAlarm.cnf ()	PLUGSM	RSMSM		—
PlugWrongAlarm.cnf ()	PLUGSM	RSMSM	AREP, API, Slot, Subslot, UserInfo;	—
PlugAlarm.req	RSMSM	PLUGSM	AREP, API, Slot, Subslot, UserInfo;	—

Primitive	Source	Destination	Associated parameters	Functions
PlugWrongAlarm.req	RSMSM	PLUGSM	AREP, API, Slot, Subslot, UserInfo;	—

7.3.2.2.6.2.4.3 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by PLUGSM are described in the service definition and shown in Table 196.

Table 196 – Local primitives issued or received by PLUGSM

Primitive	Source	Destination	Associated parameters	Functions
Local New Output_ind	FAL	PLUGSM	see 7.3.1.3.5.10	—
Local Set Input_req	PLUGSM	FAL	see 7.3.1.3.5.1	—
Local Set Output IOCS_req	PLUGSM	FAL	see 7.3.1.3.5.7	—

7.3.2.2.6.2.4.4 State transition diagram

The state transition diagram is shown in Figure 36.

IECNORM.COM : Click to view the full PDF of IEC 61158-5-10:2019

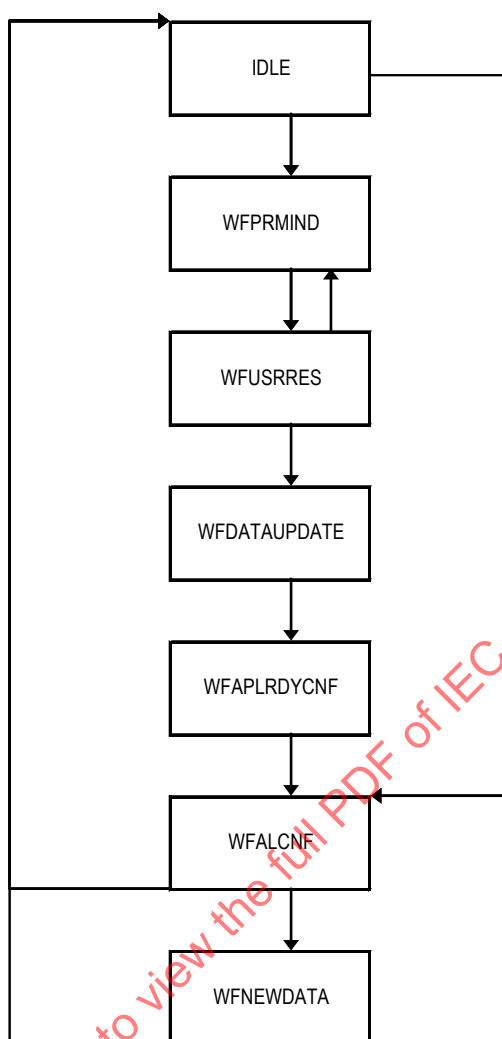


Figure 36 – State transition diagram PLUGSM

States of PLUGSM

IDLE	Idle state
WFPRMIND	Wait for PrmEnd indication
WFUSRRES	Wait for User response
WFDATAUPDATE	Wait for Data Update
WFAPLRDYCNF	Wait for ApplicationReady confirmation
WFALCNF	Wait for Alarm confirmation
WFNEWDATA	Wait for New Data indication

7.3.2.2.6.2.4.5 State machine description

This state machine handles a plug of a submodule including the re-parameterization of the submodule. The plug sequence is initiated by plugging a submodule (including a wrong one), by ownership change or else by return of a submodule (transition from BAD to GOOD).

When physically plugging a submodule, the service Local Add Submodule or Local Update Submodule is called by the application. This results in an update of the real identification. When checking the updated real identification against the expected identification, PlugAlarm.req() or PlugWrongAlarm.req() are called, when the slot / subslot where the submodule was plugged is part of some expected identification. This state machine is started at this point.

A second case is the virtual plug of the submodule, i.e. supervisor or some other IO controller, which locked this submodule, releases it, and the ownership is transferred. In this case ReleasedAlarm.req() is called, and the state machine is started at this point.

A third case is when IOPS turns to GOOD. In this case ReturnOfSubmoduleAlarm.req() is called, and the state machine is started at this point.

The state machine is concurrently started together with the transmission of the associated alarm to the IO controller.

7.3.2.2.6.2.4.6 State table

Table 197 defines the state table for the plug behavior.

Table 197 – State table PLUGSM

#	Current State	Event /Condition =>Action	Next State
1	IDLE	PlugAlarm.req () => TYPE = Plug Alarm.req ()	WFPRMIND
2	IDLE	ReleasedAlarm.req () => TYPE = Released Alarm.req ()	WFPRMIND
3	IDLE	PlugWrongAlarm.req () => TYPE = Wrong Alarm.req ()	WFALCNF
4	IDLE	ReturnOfSubmoduleAlarm.req () => TYPE = Return Alarm.req ()	WFALCNF
5	WFPRMIND	Write.ind () => UserWriteRecord.ind ()	WFUSRRES
6	WFPRMIND	PrmEnd.ind () => UserControl.ind ()	WFUSRRES
7	WFUSRRES	UserWriteRecord.rsp () => Write.rsp ()	WFPRMIND
8	WFUSRRES	UserControl.rsp () => PrmEnd.rsp ()	WFDATAUPDATE
9	WFDATAUPDATE	UserUpdate () => Local Set Input_req () Local Set Output IOCS_req () ApplicationReady.req ()	WFAPLRDYCNF
10	WFAPLRDYCNF	ApplicationReady.cnf () =>	WFALCNF
11	WFALCNF	Alarm.cnf () /((TYPE = Plug) (TYPE = Released)) && (PNIOSTATUS = OK) => Reset NewData Flag	WFNEWDATA
12	WFALCNF	Alarm.cnf () /((TYPE = Wrong) && (PNIOSTATUS = OK)) => PlugWrongAlarm.cnf (+)	IDLE

#	Current State	Event /Condition =>Action	Next State
13	WFALCNF	Alarm.cnf () /(TYPE = Return) && (PNIOSTATUS = OK) => ReturnOfSubmodulAlarm.cnf (+)	IDLE
14	WFALCNF	Alarm.cnf () /(TYPE = Plug) && (PNIOSTATUS <> OK) => PlugAlarm.cnf (-)	IDLE
15	WFALCNF	Alarm.cnf () /((TYPE = Released) && (PNIOSTATUS <> OK) => ReleasedAlarm.cnf (-)	IDLE
16	WFALCNF	Alarm.cnf () /(TYPE = Wrong) && (PNIOSTATUS <> OK) => PlugWrongAlarm.cnf (-)	IDLE
17	WFALCNF	Alarm.cnf () /(TYPE = Return) && (PNIOSTATUS <> OK) => ReturnOfSubmodulAlarm.cnf (-)	IDLE
18	WFNEWDATA	Local New Output_ind () /TYPE = Plug => PlugAlarm.cnf (+)	IDLE
19	WFNEWDATA	Local New Output_ind () /TYPE = Released => ReleasedAlarm.cnf (+)	IDLE

7.3.2.2.6.2.4.7 Functions, Macros, Timers and Variables

Table 198 contains the functions, macros, timers and variables used by PLUGSM and their arguments and descriptions.

Table 198 – Functions, Macros, Timers and Variables used by PLUGSM

Name	Type	Function/Meaning
UserUpdate	Function	This local function is an acknowledge from the user to use the data and issue an application ready. It uses the following parameters: AREP, API, Slot, Subslot, Data, IOXS

7.3.2.2.6.2.5 Pull

7.3.2.2.6.2.5.1 Primitive definitions

7.3.2.2.6.2.5.2 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by pull state machine (PULLSM) are described in the service definition and shown in Table 199.

Table 199 – Remote primitives issued or received by PULLSM

Primitive	Source	Destination	Associated parameters	Functions
Alarm.cnf ()	FAL	PULLSM	see 7.3.1.5.5	—
ControlledAlarm.req	OWNSM	PULLSM	AREP, API, Slot, Subslot, UserInfo;	—
Alarm.req	PULLSM	FAL	see 7.3.1.5.5	—
ControlledAlarm.cnf ()	PULLSM	OWNSM	AREP, API, Slot, Subslot, UserInfo;	—
PullAlarm.cnf ()	PULLSM	RSMSM	AREP, API, Slot, Subslot, UserInfo;	—
PullAlarm.req	RSMSM	PULLSM	AREP, API, Slot, Subslot, UserInfo;	—

7.3.2.2.6.2.5.3 Primitives exchanged between local machines

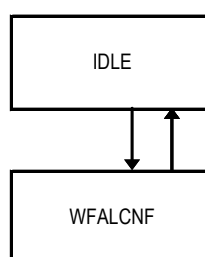
The local service primitives including their associated parameters issued or received by PULLSM are described in the service definition and shown in Table 200.

Table 200 – Local primitives issued or received by PULLSM

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

7.3.2.2.6.2.5.4 State transition diagram

The state transition diagram is shown in Figure 37.

**Figure 37 – State transition diagram PULLSM**

States of PULLSM

IDLE	Idle state
WFALCNF	Wait for Alarm confirmation

7.3.2.2.6.2.5.5 State machine description

This state machine handles a pull of a submodule. The pull sequence is initiated by pulling a submodule or by ownership change.

When physically pulling a submodule, the service Local Remove Submodule is called by the application. This results in an update of the real identification. When checking the updated real identification against the expected identification, PullAlarm.req() is called, when the slot / subslot where the submodule was pulled is part of some expected identification. This state machine is started at this point.

A second case is the virtual pull of the submodule, i.e. supervisor locks this submodule, and the ownership is withdrawn. In this case ControlledAlarm.req() is called, and the state machine is started at this point.

The state machine is concurrently started together with the transmission of the associated alarm to the IO controller.

7.3.2.2.6.2.5.6 State table

Table 201 defines the state table PULLSM.

Table 201 – State table PULLSM

#	Current State	Event /Condition =>Action	Next State
1	IDLE	PullAlarm.req () => TYPE = Pull Alarm.req ()	WFALCNF
2	IDLE	ControlledAlarm.req () => TYPE= Controlled Alarm.req ()	WFALCNF
4	WFALCNF	Alarm.cnf () /(TYPE = Pull) && (PNIOSTATUS = OK) => PullAlarm.cnf (+)	IDLE
5	WFALCNF	Alarm.cnf () /(TYPE = Controlled) && (PNIOSTATUS = OK) => ControlledAlarm.cnf (+)	IDLE
6	WFALCNF	Alarm.cnf () /(TYPE = Pull) && (PNIOSTATUS <> OK) => PullAlarm.cnf(-)	IDLE
7	WFALCNF	Alarm.cnf () /(TYPE = Controlled) && (PNIOSTATUS <> OK) => ControlledAlarm.cnf(-)	IDLE

7.3.2.2.6.2.5.7 Functions, Macros, Timers and Variables

Table 202 contains the functions, macros, timers and variables used by PULLSM and their arguments and descriptions.

Table 202 – Functions, Macros, Timers and Variables used by PULLSM

Name	Type	Function/Meaning
—	—	—

7.3.2.2.6.3 Adaption of Real Identification during Connect

After powering on, a device shall set its real identification. If there are multiple possible RIs, the device has to choose one. During Connect phase (between Connect.ind and Connect.rsp), this can be adapted (if needed).

The following cases are possible:

Default is a modular device or one with a fixed real identification. After power on, this real identification has to be created with Add/Remove/Update Submodule. Changes to the real configuration of the devices lead to changes of the real identification data.

In case of multiple possible RIs, the application has to check the Expected Identification of the incoming Connect.ind and adapt the submodules of the real identification data with Add/Remove/Update. Afterwards, the Connect.rsp is generated with the now adapted RI.

7.3.2.2.6.4 Upload & Retrieval

The user plugs a submodule and the standard sequence for plugging is run (module parameterization). The submodule detects that the IOC only partly delivered the required parameters. If it is capable of Upload&Retrieval, a diagnosis (parameter missing / error) is announced. An alarm (Upload&Retrieval alarm) is send to the IOC. In the IOC, an application reacts upon the Upload&Retrieval alarm and sends the parameters saved in the PLC to the submodule. Once all parameters are written, the submodule removes the diagnosis item. If the IOC ignores the alarm, the diagnosis item remains set.

The APO decides whether the Upload&Retrieval Diagnosis is set and this feature is used.

7.3.2.2.6.5 Submodule and channel behavior

A submodule consists of one or more channels, which are in state activated or deactivated.

An activated output channel provides data to the attached actor using a process value (according to the user application in the IO controller) or a substitute value (e.g. fixed value, holding last value, ZERO, etc.).

A deactivated output channel provides a vendor specific value "ZERO" to the attached actor (e.g. free from voltage or current).

An activated input channel provides input data and diagnosis according to the attached sensor.

A deactivated input channel may not provide input data or diagnosis.

A submodule has two states, activated and deactivated. If the state of the submodule is deactivated, all channels of the submodule are deactivated. If the state of the submodule is activated, a particular channel may be activated or deactivated, depending on parameterization or function of the channel.

The state of the submodule is defined as follows:

- After PowerOn
 - If parameters for the submodule reside in persistency, the parameters decide the behavior of the submodule and its channels.
 - Without parameters in persistency, the behavior of the submodule and its channels is vendor specific (activated or deactivated).
- After application relationship is established
 - If the submodule is not owned by the application relationship, submodule and its channels continue to behave as before.
 - Otherwise, the submodule is activated. If parameters are written to the submodule, the parameters control the behavior of the channels. If no parameters are written and the submodule is able to work with default parameter data, all channels are implicitly activated.
- After application relationship is aborted or closed, the behavior of the submodule and its channels continues to be controlled by the parameters written in the application relationship (e.g. substitute value or ZERO).
- After dynamic reconfiguration, where the submodule was removed from the application relationship
 - The submodule is deactivated.
 - The persistency for this submodule is set to the vendor specific default.
 - The behavior for the submodule is according to the vendor specific default behavior (channels are preferred deactivated).
- After reset to factory concerning the submodule's persistent data (RESET_APPLICATION_DATA)
 - The persistency for this submodule is set to the vendor specific default.
 - The behavior for the submodule is according to the vendor specific default (channels are preferred deactivated).

For persistency of record data the persistency defined in the Communication Interface Management ASE is used; see 7.3.3.2.6.2 for details.

7.3.2.3 I&M Data class

7.3.2.3.1 Overview

This class provides information to support the end user during various scenarios of asset life cycle management be it identification, configuration, commissioning, parameterization, diagnosis, maintenance, repair, firmware update, asset management, audit trailing, and alike.

Assets are represented in two possible characteristics:

- An asset modeled according to this document (e.g. IO data, parameterization...) is represented by a submodule or a group of submodules; its life cycle information indicated by I&M data.
- The life cycle information of an asset not modeled by this document is indicated in Asset Management Data.

For further details see 7.3.2.3.6.

7.3.2.3.2 Relations

The services are bound to other services as shown in Table 203:

Table 203 – Binding I&M data services

Service	Binding	Usage
Read I&M0 Filter Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read I&M0 Data	AR:Record Data:Read	used by IO controller or IO supervisor
Write I&M1 Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read I&M1 Data	AR:Record Data:Read	used by IO controller or IO supervisor
Write I&M2 Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read I&M2 Data	AR:Record Data:Read	used by IO controller or IO supervisor
Write I&M3 Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read I&M3 Data	AR:Record Data:Read	used by IO controller or IO supervisor
Write I&M4 Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read I&M4 Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read I&M5 Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read Asset Management Record	AR:Record Data:Read	used by IO controller or IO supervisor

7.3.2.3.3 Template

This class specifies the object for the parameter data that are related to the IO devices itself and/or to the modules/submodules of the IO device.

The IO device context object is described by the following template:

ASE: Real Identification ASE

CLASS: I&M Data

CLASS ID: not used

PARENT CLASS: TOP

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: List of I&M Carriers
- 2.1 (m) Attribute: API
- 2.2 (m) Attribute: Slot Number
- 2.3 (m) Attribute: Subslot Number
- 2.4 (m) Attribute: I&M0 Data
- 2.4.1 (m) Attribute: Vendor ID
- 2.4.2 (m) Attribute: Order ID
- 2.4.3 (m) Attribute: IM Serial Number
- 2.4.4 (m) Attribute: IM Hardware Revision
- 2.4.5 (m) Attribute: IM Software Revision
- 2.4.5.1 (m) Attribute: Prefix
- 2.4.5.2 (m) Attribute: Functional Enhancement
- 2.4.5.3 (m) Attribute: Bug Fix
- 2.4.5.4 (m) Attribute: Internal Change

2.4.6	(m) Attribute:	IM Revision Counter
2.4.7	(m) Attribute:	IM Profile ID
2.4.8	(m) Attribute:	IM Profile Specific Type
2.4.9	(m) Attribute:	IM Version
2.4.10	(m) Attribute:	IM Supported
2.5	(o) Attribute:	I&M1 Data
2.5.1	(m) Attribute:	Function
2.5.2	(m) Attribute:	Location
2.6	(o) Attribute:	I&M2 Data
2.6.1	(m) Attribute:	Installation Date
2.7	(o) Attribute:	I&M3 Data
2.7.1	(m) Attribute:	Description
2.8	(o) Attribute:	I&M4 Data
2.9	(o) Attribute:	I&M5 Data
2.9.1	(m) Attribute:	Number Of Entries
2.9.1.1	(m) Attribute:	I&M5 Data Format
2.9.1.2	(c) Constraint:	I&M5 Data Format = I&M5Data
2.9.1.1.1	(m) Attribute:	IM Annotation
2.9.1.1.2	(m) Attribute:	Vendor ID
2.9.1.1.3	(m) Attribute:	IM Order ID
2.9.1.1.4	(m) Attribute:	IM Serial Number
2.9.1.1.5	(m) Attribute:	IM Hardware Revision
2.9.1.1.6	(m) Attribute:	IM Software Revision
2.9.1.3	(c) Constraint:	I&M5 Data Format = AMData
2.9.1.3.1	(m) Attribute:	IM Unique Identifier
2.9.1.3.2	(m) Attribute:	AM Location
2.9.1.3.3	(m) Attribute:	IM Annotation
2.9.1.3.4	(m) Attribute:	IM Order ID
2.9.1.3.5	(o) Attribute:	AM Software Revision
2.9.1.3.6	(o) Attribute:	AM Hardware Revision
2.9.1.3.7	(m) Attribute:	IM Serial Number
2.9.1.3.8	(o) Attribute:	IM Software Revision
2.9.1.3.9	(m) Attribute:	AM Device Identification
2.9.1.3.10	(m) Attribute:	AM Type Identification
2.9.1.3.11	(o) Attribute:	IM Hardware Revision
2.10	(m) Attribute:	Representative
2.10.1	(m) Attribute:	Module
2.10.2	(m) Attribute:	Device
3	(o) Attribute:	Asset Management Data
3.1	(m) Attribute:	IM Unique Identifier
3.2	(m) Attribute:	AM Location
3.3	(m) Attribute:	IM Annotation
3.4	(m) Attribute:	IM Order ID
3.5	(o) Attribute:	AM Software Revision
3.6	(o) Attribute:	AM Hardware Revision
3.7	(m) Attribute:	IM Serial Number
3.8	(o) Attribute:	IM Software Revision
3.9	(m) Attribute:	AM Device Identification
3.10	(m) Attribute:	AM Type Identification
3.11	(o) Attribute:	IM Hardware Revision

SERVICES:

1	(m) OpsService:	Read I&M0 Filter Data
2	(m) OpsService:	Read I&M0 Data

3	(o) OpsService: Write I&M1 Data
4	(o) OpsService: Read I&M1 Data
5	(o) OpsService: Write I&M2 Data
6	(o) OpsService: Read I&M2 Data
7	(o) OpsService: Write I&M3 Data
8	(o) OpsService: Read I&M3 Data
9	(o) OpsService: Write I&M4 Data
10	(o) OpsService: Read I&M4 Data
11	(o) OpsService: Read I&M5 Data
12	(o) OpsService: Read Asset Management Record

7.3.2.3.4 Attributes

Implicit

The key attribute Implicit indicates that the I&M data object is implicitly addressed by the services.

List of I&M Carriers

This attribute lists those submodules (according to the real identification) being a carrier of I&M data.

One list element consists of the following attributes:

API

This attribute defines the number of the application process the submodule is assigned to.

Attribute type: Unsigned32

Allowed values: 0 to 0xFFFFFFFF

Slot Number

This attribute defines the slot the submodule is assigned to.

Attribute type: Unsigned16

Allowed values: 0 to 0x7FFF

Subslot Number

This attribute defines the subslot the submodule is assigned to.

Attribute type: Unsigned16

Allowed values: 0 to 0x9FFF

I&M0 Data

This attribute consists of the following attributes.

Vendor ID

This attribute defines the ID of the submodule's manufacturer. 0x0000 is used in case a manufacturer does not want to handle an ID.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF

Order ID

This attribute defines the complete order number or at least a relevant part that allows unambiguous identification of the submodule within the manufacturer's order spectrum.

The attribute is filled as visible string. Unused octets shall be set to 0x20 (blank).

Attribute type: VisibleString[20]

IM Serial Number

This attribute defines the serial number of the submodule. The serial number is a unique production number of the device manufacturer even for devices with the same hardware, software or firmware revision.

The attribute is filled as visible string. Unused octets shall be set to 0x20 (blank).

Attribute type: VisibleString[16]

IM Hardware Revision

This attribute defines the hardware revision of the submodule.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF

IM Software Revision

This attribute defines the software or firmware revision of the submodule. Any component that carries software shall present its software version even if the user may not be aware of it. If the submodule contains no software, the software revision should be set to "V0.0.0".

This attribute consists of the following attributes:

Prefix

The prefix attribute forms the "Vx.y.z" part of the software revision.

Attribute type: VisibleString[1]

Preferred values:

"V" (= officially released version) or "R" (= Revision)

"P" = Prototype

"U" = Under Test (field test)

"T" = Test Device

Functional Enhancement

The functional enhancement attribute forms the "Vx.y.z" part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Bug Fix

The bug fix attribute forms the "Vx.y.z" part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Internal Change

The internal change attribute forms the "Vx.y.z" part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

IM Revision Counter

This attribute is not used in the context of this document.

Attribute type: Unsigned16

Allowed values: 0

IM Profile ID

This attribute defines the ID of the application profile the submodule is following. Submodules not following any application profile shall use 0x0000.

Attribute type: Unsigned16

Allowed values: 0x0000 (Non profile device), 0x0001 – 0xF6FF (administrative number), 0xF700 – 0xFFFE (reserved for profiles)

IM Profile Specific Type

In case a submodule follows a special application profile, this attribute offers information about the usage of its channels and/or sub devices. The submodule/channel information shall be according to the respective definitions of the application profile.

In conjunction with Profile ID == 0 the attribute indicates the type of sensor or actuator.

Attribute type: Unsigned16

Allowed values:

Unspecified (0x0000): This value can be used if none of the well defined values is suitable

Standard Controller (0x0001): This value shall be used for a PLC

PC-based Station (0x0002): This value shall be used for a personal computer based station, if no other value is suitable.

IO-Module (0x0003): This value shall be used for submodules offering a remote IO functionality.

Communication Module (0x0004): This value shall be used for submodules working as a communication module, for example an IO Link Master.

Interface Module (0x0005): This value shall be used for a submodule representing the IO device.

Active Network Infrastructure Component (0x0006): This value shall be used for ethernet bridges / switches or media converter.

Media attachment unit (0x0007): This value shall be used for media attachment units, e.g. an adapter to connect different Ethernet MAU types

Manufacturer Specific Type (0x0100 – 0x7FFF): This value is defined by the vendor of the device.

In case a submodule follows an application profile, the definition is stated there.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF

IM Version

This attribute indicates the implemented version of the I&M functions as structure with major and minor version. Octet 1 (MSB) contains the major version number, e.g. 1 of version 1.0; octet 2 (LSB) contains the minor version number, e.g. 0 of version 1.0.

Attribute type: OctetString[2]

IM Supported

This attribute indicates the availability of further I&M records. Bit 0 indicates the availability of Profile specific I&M, Bit 1 the availability of I&M1, Bit 2 the availability of I&M2, ... and Bit 15 the availability of I&M15.

Attribute type: V2

I&M1 Data

This optional attribute consists of the following attributes.

IM Tag Function

This attribute indicates the submodule's function or task. This may be a standard symbolic tag ("AKZ") out of a list or any other type of label defined by a configuration tool. Unused characters shall be set to 0x20 (blank).

Attribute type: VisibleString[32]

IM Tag Location

This attribute indicates the submodule's location. This may be a standard symbolic tag ("OKZ") out of a list or any other type of label defined by a configuration tool. Unused characters shall be set to 0x20 (blank).

Attribute type: VisibleString[22]

I&M2 Data

This optional attribute consists of the following attribute.

IM Date

This attribute indicates the date of installation or commissioning of a device or module. The content is a fixed date and time format according to ISO 8601: YYYY-MM-DD T HH:MM, where YYYY is the year in the usual Gregorian calendar, MM is the month of the year between 01 (January) and 12 (December), DD is the day of the month between 01 and 31, where HH is the hour of the day (0 - 23) and MM the minutes of the hour (0 - 59).

Time and date are normally separated by a "T" but for this appearance shall be separated by a blank character.

For example, the fourth day of February in the year 1995 is written in the standard notation as:

1995-02-04 16:23

Unused characters shall be set to 0x20 (blank).

Attribute type: VisibleString[16]

I&M3 Data

This optional attribute consists of the following attributes.

IM Descriptor

This attribute allows storing any individual additional information and annotation.

Unused characters shall be set to 0x20 (blank).

Attribute type: VisibleString[54]

I&M4 Data

This optional attribute should only be used together with Functional Safety.

I&M5 Data

This optional attribute allows identifying additional information for this asset. It consists of the following attributes.

Number Of Entries

This attribute indicates the number of entries in the following list.

Attribute type: Unsigned16

Data Format

This attribute identifies the data format used for the I&M5 data.

Allowed values: IMFormat, AMFormat

IMFormat

If Data Format is “IMFormat”, OEM information for this asset is specified. There is at most one information in “IMFormat”. For other information use “AMFormat”.

IM Annotation

This conditional attribute is present when Data Format is “IMFormat”.

It allows to identify the OEM part using text.

The attribute is filled as visible string. Unused octets shall be set to 0x20 (blank).

Attribute type: UnicodeString8[64]

Allowed values: 0 to 0xFFFF

IM Order ID

This conditional attribute is present when Data Format is “IMFormat”.

This attribute defines the complete order number or at least a relevant part that allows unambiguous identification of the submodule within the manufacturer's order spectrum.

The attribute is filled as visible string. Unused octets shall be set to 0x20 (blank).

Attribute type: UnicodeString8[64]

Vendor ID

This conditional attribute is present when Data Format is “IMFormat”.

This attribute defines the ID of the submodule's manufacturer according to the attribute in I&M0 data in this class

IM Serial Number

This conditional attribute is present when Data Format is “IMFormat”.

This attribute defines the serial number of the submodule according to the attribute in I&M0 data in this class.

IM Hardware Revision

This conditional attribute is present when Data Format is “IMFormat”.

This attribute defines the hardware revision of the submodule.

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF

IM Software Revision

This conditional attribute is present when Data Format is “IMFormat”.

This attribute defines the software or firmware revision of the submodule.

This attribute consists of the following attributes:

Prefix

The prefix attribute forms the “Vx.y.z” part of the software revision.

Attribute type: VisibleString[1]

Preferred values:

“V” (= officially released version) or “R” (= Revision)

“P” = Prototype

“U” = Under Test (field test)

“T” = Test Device

Functional Enhancement

The functional enhancement attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Bug Fix

The bug fix attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Internal Change

The internal change attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

AMFormat

If Data Format is “AMFormat”, the information specifies additional information for this asset. AM Type Information specifies the type of information.

IM Unique Identifier

This conditional attribute is present when Data Format is “AMFormat”.

This attribute identifies the asset uniquely during the lifetime of the asset. It is a manufacturer created unique identifier (UUID) according to ISO/IEC 9834-8.

Attribute type: UUID

AM Location

This conditional attribute is present when Data Format is “AMFormat”.

This attribute is used to specify the asset's location. The coding is defined in IEC 61158-6-10.

In case of I&M5 Data AM Location shall be set to “0”.

Attribute type: OctetString[16]

IM Annotation

This conditional attribute is present when Data Format is “AMFormat”.

This attribute specifies the asset's annotation; used for a human readable description of the asset. The format and content of IM Annotation is vendor specific.

The string is left justified and filled with blanks (0x20) if the annotation is shorter than 64 characters.

Attribute type: UnicodeString8[64]

IM Order ID

This conditional attribute is present when Data Format is “AMFormat”.

This attribute specifies the asset's complete order number or at least a relevant part that allows unambiguous identification of the asset within the manufacturer's order spectrum. The format and content of IM Order ID is vendor specific.

The string in the annotation is left justified and filled with blanks (0x20) if the IM Order ID is shorter than 64 characters.

Attribute type: UnicodeString8[64]

AM Software Revision

This conditional attribute is present when Data Format is “AMFormat”.

This optional attribute is present only if the asset contains software or firmware.

It contains the manufacturer defined software revision in a “free” format. If the asset's software revision can be described using IM Software Revision (e.g. “V1.2.3”), then this field shall be filled with blanks.

For an evaluating tool: If this field contains a value (i.e. is not filled with blanks only), show this value, otherwise the value in IM Software Revision.

Attribute type: UnicodeString8[64]

AM Hardware Revision

This conditional attribute is present when Data Format is “AMFormat”.

This optional attribute is present only if the asset contains hardware.

It contains the manufacturer defined hardware revision in a “free” format. If the asset's hardware revision can be described using IM_Hardware_Revision (e.g. “17”), then this field shall be filled with blanks.

For an evaluating tool: If this field contains a value (i.e. is not filled with blanks only), show this value, otherwise the value in IM Hardware Revision.

Attribute type: UnicodeString8[64]

IM Serial Number

This conditional attribute is present when Data Format is “AMFormat”.

This attribute defines the asset's serial number. The serial number is a unique production number of the device manufacturer even for devices with the same hardware, software or firmware revision.

The string in the IM Serial Number is left justified and filled with blanks (0x20) if the IM Order ID is shorter than 16 characters.

Attribute type: VisibleString[16]

IM Software Revision

This conditional attribute is present when Data Format is “AMFormat”.

This optional attribute is present only if the asset contains software or firmware.

This attribute defines the software revision of the asset.

If the software revision is specified using AM Software Revision, fill this field with “V0.0.0”.

This attribute consists of the following attributes:

Prefix

The prefix attribute forms the “Vx.y.z” part of the software revision.

Attribute type: VisibleString[1]

Preferred values:

“V” (= officially released version) or “R” (= Revision)

“P” = Prototype

“U” = Under Test (field test)

“T” = Test Device

Functional Enhancement

The functional enhancement attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Bug Fix

The bug fix attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Internal Change

The internal change attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

AM Device Identification

This conditional attribute is present when Data Format is “AMFormat”.

This attribute is used to specify the asset’s organization, vendor, device and device sub identification. The coding is defined in IEC 61158-6-10.

The device identification is hierarchically structured, meaning

- organization defines the scope for vendor,
- vendor the scope for device,
- and device the scope for device sub identification.

Attribute type: OctetString[8]

AM Type Identification

This conditional attribute is present when Data Format is “AMFormat”.

This attribute defines the asset’s type according to the attribute IM Profile Specific Type in I&M0 data in this class. This attribute specifies the type of the information for this asset.

The special value 0x8000 indicates that OEM information for this asset is specified.

IM Hardware Revision

This conditional attribute is present when Data Format is “AMFormat”.

This optional attribute is present only if the asset contains hardware.

This attribute defines the asset’s hardware revision according to the attribute in I&M0 data in this class.

If the hardware revision is specified using AM Hardware Revision, fill this field with “0x0000”.

Representative

This attribute defines whether this submodule is a representative for module and/or device.

It consists of

Module

This attribute defines whether this submodule is a representative for the module.

Attribute type: Boolean

Allowed values: FALSE, TRUE

Device

This attribute defines whether this submodule is a representative for the device.

Attribute type: Boolean

Allowed values: FALSE, TRUE

Asset Management Data

This attribute is used to specify the assets not modeled using this document; i.e. the assets not represented by a submodule.

It consists of the following elements.

Number Of Entries

This attribute indicates the number of entries in the following list.

IM Unique Identifier

This attribute identifies the asset uniquely during the lifetime of the asset. It is a manufacturer created unique identifier (UUID) according to ISO/IEC 9834-8.

Attribute type: UUID

AM Location

This attribute is used to specify the asset’s location. The coding is defined in IEC 61158-6-10.

AM Location allows to specify the location in two format alternatives.

One is a twelve level format, whereas each level allows to specify a number between 0 and 1 022 resp. an indication that this level is not used. A sample would be “12.7.5.6”. Using this format the AM Location allows to locate the assets in a tree structure; how to locate the position in this tree → handbuch.

The other is a slotnumber / subslotnumber format, whereas the physical position of the asset is noted with a begin and end slotnumber and a begin and end subslotnumber. It is also possible to express that the asset covers a whole slot or multiple whole slots. A

sample would be “covers slot 1 and 2 on the whole”. Using this format, the AM Location is aligned to the physical layout of the device.

Attribute type: OctetString[16]

IM Annotation

This attribute specifies the asset’s annotation; used for a human readable description of the asset. The format and content of IM Annotation is vendor specific.

The string is left justified and filled with blanks (0x20) if the annotation is shorter than 64 characters.

Attribute type: UnicodeString8[64]

IM Order ID

This attribute specifies the asset’s complete order number or at least a relevant part that allows unambiguous identification of the asset within the manufacturer’s order spectrum. The format and content of IM Order ID is vendor specific.

The string in the annotation is left justified and filled with blanks (0x20) if the IM Order ID is shorter than 64 characters.

Attribute type: UnicodeString8[64]

AM Software Revision

This optional attribute is present only if the asset contains software or firmware.

It contains the manufacturer defined software revision in a “free” format. If the asset’s software revision can be described using IM Software Revision (e.g. “V1.2.3”), then this field shall be filled with blanks.

For an evaluating tool: If this field contains a value (i.e. is not filled with blanks only), show this value, otherwise the value in IM Software Revision.

Attribute type: UnicodeString8[64]

AM Hardware Revision

This optional attribute is present only if the asset contains hardware.

It contains the manufacturer defined hardware revision in a “free” format. If the asset’s hardware revision can be described using IM_Hardware_Revision (e.g. “17”), then this field shall be filled with blanks.

For an evaluating tool: If this field contains a value (i.e. is not filled with blanks only), show this value, otherwise the value in IM Hardware Revision.

Attribute type: UnicodeString8[64]

IM Serial Number

This attribute defines the asset’s serial number. The serial number is a unique production number of the device manufacturer even for devices with the same hardware, software or firmware revision.

The string in the IM Serial Number is left justified and filled with blanks (0x20) if the IM Order ID is shorter than 16 characters.

Attribute type: VisibleString[16]

IM Software Revision

This optional attribute is present only if the asset contains software or firmware.

This attribute defines the software revision of the asset according to the attribute in I&M0 data in this class.

If the software revision is specified using AM Software Revision, fill this field with “V0.0.0”.

This attribute consists of the following attributes:

Prefix

The prefix attribute forms the “Vx.y.z” part of the software revision.

Attribute type: VisibleString[1]

Preferred values:

“V” (= officially released version) or “R” (= Revision)

“P” = Prototype

“U” = Under Test (field test)

“T” = Test Device

Functional Enhancement

The functional enhancement attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Bug Fix

The bug fix attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

Internal Change

The internal change attribute forms the “Vx.y.z” part of the software revision.

Attribute type: Unsigned8

Allowed values: 0 to 0xFF

AM Device Identification

This attribute is used to specify the asset’s organization, vendor, device and device sub identification. The coding is defined in IEC 61158-6-10.

The device identification is hierarchically structured, meaning

- organization defines the scope for vendor,
- vendor the scope for device,
- and device the scope for device sub identification.

Attribute type: OctetString[8]

AM Type Identification

This attribute specifies the asset’s type.

This attribute defines the asset’s type according to the attribute IM Profile Specific Type in I&M0 data in this class.

IM Hardware Revision

This optional attribute is present only if the asset contains hardware.

This attribute defines the asset's hardware revision. If the asset's hardware revision is specified using AM Hardware Revision, fill this field with "0x0000".

Attribute type: Unsigned16

Allowed values: 0 to 0xFFFF

7.3.2.3.5 Services

7.3.2.3.5.1 Read I&M0 Filter Data

This service allows reading I&M0 Filter Data using a record data object with device scope (see 7.3.1.6.1).

I&M0 Filter Data indicates all submodules being a carrier of discrete identification and maintenance data. I&M1, I&M2, I&M3, and I&M4, if supported, can be written on such submodules.

Optionally the submodules acting as module representative are indicated.

I&M0 Filter Data indicates one device representative on which I&M1, I&M2, and I&M3 shall be writeable.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 204 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 204 – Data elements of Read I&M0 Filter Data

Parameter name	service specific elements	rsp	cnf
Data	I&M0 Filter Data	M	M(=)
	I&M0 Carrier Data	M	M(=)
	List of APs	M	M(=)
	API	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(=)
	Module Representative Data	O	O(=)
	List of APs	M	M(=)
	API	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(=)
	Device Representative Data	M	M(=)
	List of APs	M	M(=)

Parameter name	service specific elements	rsp	cnf
	API	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(=)

Data

This parameter contains the values of the object which has been read.

I&M0 Filter Data

This attribute consists of the following attributes:

I&M0 Carrier Data

This attribute contains all submodules being carrier of identification and maintenance data.

It consists of the following attributes.

List of APs

This attribute is composed of the following list elements:

API

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

List of Slots

This attribute defines the slots of the application process. One Slot 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 of the corresponding attribute of the ASE object.

List of Subslots

This attribute consists of the following attributes.

Subslot Number

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

Submodule Ident Number

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

Module Representative Data

This optional attribute indicates the submodules acting as module representative and thus carrying the module's I&M data; at most one submodule being carrier of identification and maintenance information shall be returned per module.

It consists of the following attributes.

List of APs

This attribute is composed of the following list elements:

API

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

List of Slots

This attribute defines the slots of the application process. One Slot 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 of the corresponding attribute of the ASE object.

List of Subslots

This attribute consists of the following attributes.

Subslot Number

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

Submodule Ident Number

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

Device Representative Data

This attribute contains exactly one submodule, which is a carrier of identification and maintenance information and provides access to I&M1, I&M2, I&M3 and I&M4.

This attribute consists of the following attributes.

List of APs

This attribute is composed of the following list elements:

API

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

List of Slots

This attribute defines the slots of the application process. One Slot 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 of the corresponding attribute of the ASE object.

List of Subslots

This attribute consists of the following attributes.

Subslot Number

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

Submodule Ident Number

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

7.3.2.3.5.2 Read I&M0 Data

This service allows reading I&M0 Data using a record data object with subslot scope (see 7.3.1.6.1).

This confirmed service may be used to read I&M0 Data. This service shall only be used in conjunction with the implicit AR, IO AR or Supervisor AR.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 205 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 205 – Data elements of Read I&M0 Data

Parameter name	service specific elements	rsp	cnf
Data	I&M0 Data	C	C(=)
	Vendor ID	M	M(=)
	Order ID	M	M(=)
	IM Serial Number	M	M(=)
	IM Hardware Revision	M	M(=)
	IM Software Revision	M	M(=)
	IM Revision Counter	M	M(=)
	IM Profile ID	M	M(=)
	IM Profile Specific Type	M	M(=)
	IM Version	M	M(=)
	IM Supported	M	M(=)

Data

This parameter contains the values of the object which has been read.

I&M0 Data

This parameter consists of the following elements.

Vendor ID

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

Order ID

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

IM Serial Number

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

IM Hardware Revision

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

IM Software Revision

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

IM Revision Counter

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

IM Profile ID

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

IM Profile Specific Type

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

IM Version

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

IM Supported

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

The following parameters have a specific meaning at this service:

Length (Result(+))

The parameter Length indicates the number of octets of the I&M0 Data read.

Special case (legacy submodules / modules):

The value zero, in conjunction with an entry in the I&M0 Carrier Data, indicates that the submodule is not able to retrieve its I&M0 data.

In this case the engineering system may use locally generated I&M0 data.

7.3.2.3.5.3 Write I&M1 Data

This service allows writing I&M1 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 206 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 206 – Data elements of Write I&M1 Data

Parameter name	service specific elements	req	ind
Data	I&M1 Data	M	M(=)
	IM Tag Function	M	M(=)
	IM Tag Location	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

I&M1 Data

This parameter consists of the following elements.

IM Tag Function

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

IM Tag Location

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

7.3.2.3.5.4 Read I&M1 Data

This service allows reading I&M1 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 206 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write I&M1 Data (7.3.2.3.5.3).

7.3.2.3.5.5 Write I&M2 Data

This service allows writing I&M2 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 207 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 207 – Data elements of Write I&M2 Data

Parameter name	service specific elements	req	ind
Data	I&M2 Data	M	M(=)
	IM Date	M	M(=)

Data

This parameter contains the value of the object which may be used to write.

I&M2 Data

This parameter consists of the following elements.

IM Date

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

7.3.2.3.5.6 Read I&M2 Data

This service allows reading I&M2 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 207 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write I&M2 Data in 7.3.2.3.5.5.

7.3.2.3.5.7 Write I&M3 Data

This service allows writing I&M3 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 208 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 208 – Data elements of Write I&M3 Data

Parameter name	service specific elements	req	ind
Data	I&M3 Data	M	M(=)
	IM Descriptor	M	M(=)

Data

This parameter contains the value of the object which may be used to write.

I&M3 Data

This parameter consists of the following elements.

IM Descriptor

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

7.3.2.3.5.8 Read I&M3 Data

This service allows reading I&M3 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 208 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write I&M3 Data in 7.3.2.3.5.7.

7.3.2.3.5.9 Write I&M4 Data

This service allows writing I&M4 Data using a record data object with subslot scope (see 7.3.1.6.1).

If the I&M4 Data is used in the context of application profile functional safety, the data should only be written by the functional safety application. The Write I&M4 Data service should be denied, due to the functional safety assignment.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 209 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 209 – Data elements of Write I&M4 Data

Parameter name	service specific elements	req	ind
Data	I&M4 Data	M	M(=)

Data

This parameter contains the value of the object which may be used to write.

I&M4 Data

This parameter is defined by the functional safety profile.

7.3.2.3.5.10 Read I&M4 Data

This service allows reading I&M4 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 209 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write I&M4 Data in 7.3.2.3.5.9.

7.3.2.3.5.11 Read I&M5 Data

This service allows reading I&M5 Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 210 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 210 – Data elements of Read I&M5 Data

Parameter name	service specific elements	rsp	cnf
Data	I&M5 Data	C	C(=)
	Number Of Entries	M	M(=)
	Data Format	M	M(=)
	IMFormat	C	M(=)
	IM Annotation	M	M(=)
	Vendor ID	M	M(=)
	IM Order ID	M	M(=)
	IM Serial Number	M	M(=)
	IM Hardware Revision	M	M(=)
	IM Software Revision	M	M(=)
	AMFormat	C	M(=)
	IM Unique Identifier	M	M(=)
	AM Location	M	M(=)
	IM Annotation	M	M(=)
	IM Order ID	M	M(=)
	AM Software Revision	U	U(=)
	AM Hardware Revision	U	U(=)
	IM Serial Number	M	M(=)
	IM Software Revision	U	U(=)
	AM Device Identification	M	M(=)
	AM Type Identification	M	M(=)
	IM Hardware Revision	U	U(=)

Data

This parameter contains the value of the object which has been read.

I&M5 Data

This parameter consists of the following elements.

Number Of Entries

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

Data Format

This attribute identifies the data format used for the I&M5 data.

Allowed values: IMFormat, AMFormat

IMFormat

IM Annotation

This conditional attribute is present when Data Format is "IMFormat".

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

IM Order ID

This conditional attribute is present when Data Format is "IMFormat".

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

Vendor ID

This conditional attribute is present when Data Format is "IMFormat".

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

IM Serial Number

This conditional attribute is present when Data Format is "IMFormat".

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

IM Hardware Revision

This conditional attribute is present when Data Format is "IMFormat".

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

IM Software Revision

This conditional attribute is present when Data Format is "IMFormat".

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

AMFormat

IM Unique Identifier

This conditional attribute is present when Data Format is "AMFormat".

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

AM Location

This conditional attribute is present when Data Format is "AMFormat".

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

IM Annotation

This conditional attribute is present when Data Format is “AMFormat”.

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

IM Order ID

This conditional attribute is present when Data Format is “AMFormat”.

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

AM Software Revision

This conditional attribute is present when Data Format is “AMFormat”.

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

AM Hardware Revision

This conditional attribute is present when Data Format is “AMFormat”.

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

IM Serial Number

This conditional attribute is present when Data Format is “AMFormat”.

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

IM Software Revision

This conditional attribute is present when Data Format is “AMFormat”.

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

AM Device Identification

This conditional attribute is present when Data Format is “AMFormat”.

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

AM Type Identification

This conditional attribute is present when Data Format is “AMFormat”.

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

IM Hardware Revision

This conditional attribute is present when Data Format is “AMFormat”.

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

7.3.2.3.5.12 Read Asset Management Record

This service allows reading Asset Management Data using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 211 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 211 – Data elements of Read Asset Management Data

Parameter name	service specific elements	rsp	cnf
Data	Asset Management Data	C	C(=)
	Number Of Entries	M	M(=)
	IM Unique Identifier	M	M(=)
	AM Location	M	M(=)
	IM Annotation	M	M(=)
	IM Order ID	M	M(=)
	AM Software Revision	U	U(=)
	AM Hardware Revision	U	U(=)
	IM Serial Number	M	M(=)
	IM Software Revision	U	U(=)
	AM Device Identification	M	M(=)
	AM Type Identification	M	M(=)
	IM Hardware Revision	U	U(=)

Data

This parameter contains the value of the object which has been read.

Asset Management Data

This parameter contains the value of the corresponding attribute of the ASE object and consists of the following elements.

Number Of Entries

This attribute indicates the number of entries in the following list.

IM Unique Identifier

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

AM Location

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

IM Annotation

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

IM Order ID

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

AM Software Revision

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

AM Hardware Revision

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

IM Serial Number

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

IM Software Revision

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

AM Device Identification

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

AM Type information

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

IM Hardware Revision

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

7.3.2.3.6 Behavior**7.3.2.3.6.1 I&M Data**

As stated in 7.3.2.3.1 an asset modeled according to this document is represented by a submodule or a group of submodules.

Exactly one of the asset's submodules is responsible for carrying the identification and maintenance data. This submodule is called the carrier. Only the carrier allows write access (if supported) to the identification and maintenance data.

To allow each application relation to access the identification and maintenance information of a particular submodule (e.g. shared device), each submodule of the group shall support reading identification and maintenance data. It thus acts as a proxy to its carrier (if not being itself the carrier) and returns the identification and maintenance information as stored at the carrier.

The following rules apply to I&M0 filter data (see 7.3.2.3.5.1):

- The list of carriers is returned with I&M0 carrier data.
- There is at least one submodule of the device, where I&M1, I&M2, and I&M3 can be written. This submodule is indicated in the device representative list.

The I&M0 Data of the device representative shall be consistent to RPC Annotation, System Description and Chassis ID of the device. The System Description may differ for historic reasons, but should not. The Vendor ID within RPC Annotation in I&M0 Data shall be identical to one used in the GSD.

If I&M1, I&M2, and I&M3 are supported, their default is either NULL (NULL response when reading the data record; preferred) or attributes filled with 0x20 (blank).

Devices following the application profile functional safety shall provide I&M4. In this case the restrictions of functional safety for I&M4 apply. Devices not following the application profile may provide I&M4; however it is not recommended to do so.

I&M5 allows offering a list of identification information for a particular submodule.

Devices built by using separately handable OEM parts (e.g. pluggable interface submodules) provide their own identification using I&M0 as usual. The OEM identification, in particular order number and versioning information of the OEM, may additionally be offered using I&M5. Use for the OEM identification either the IMFormat or the AMFormat vom AM Type Identification set to 0x8000.

Additional identification information (e.g. software driver...) is always stated in AMFormat, using an appropriate AM Type Identification.

7.3.2.3.6.2 Asset Management Data

As stated in 7.3.2.3.1 there could be assets which are not modeled according to this document. All such assets are stated in the asset management data.

7.3.2.4 Isochronous Mode Application Class

7.3.2.4.1 Overview

This class 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 class is an option and does not effect non-isochronous applications.

Figure 38 shows the basic model for isochronous applications:

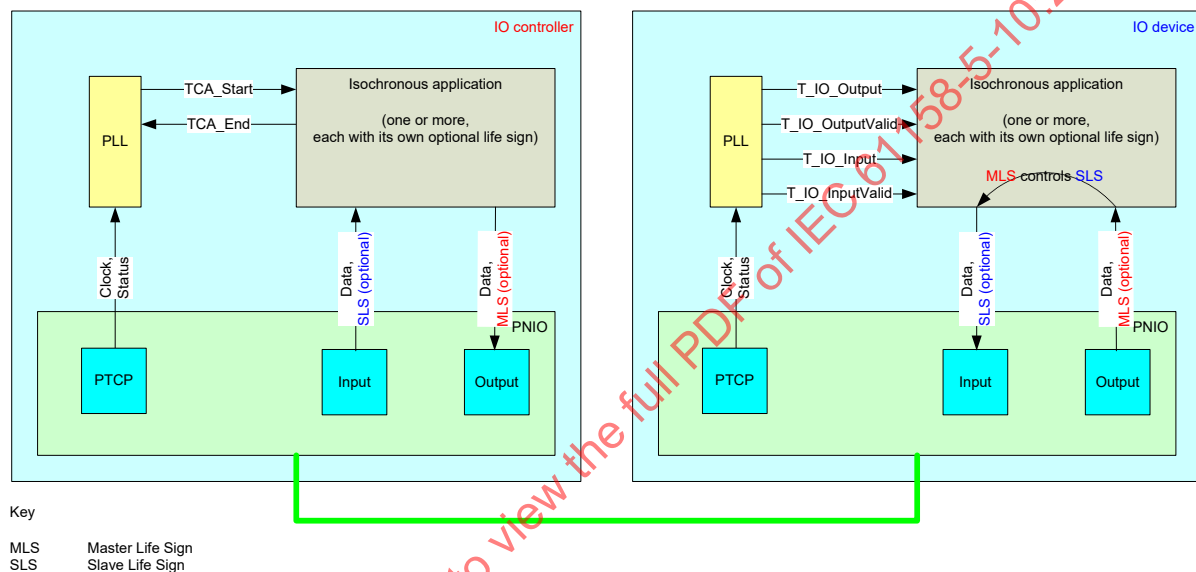


Figure 38 – Basic model for isochronous applications

Isochronous Mode couples signal measurement, transfer and processing, transfer and signal output to the conveyance of the fieldbus. Isochronous mode is application coupling and not an attribute of the transmitting media.

Thus one should **not** confuse isochronous mode with an equidistant fieldbus. In fact it is possible to model isochronous mode onto a non-equidistant fieldbus or run applications freely on top of an equidistant fieldbus. However to reduce complexity isochronous mode is modeled typically onto an equidistant fieldbus, to make the transfer of data robust and deterministic.

Isochronous mode means, that a bunch of peripheral inputs, their processing within an application program on the controller and a bunch of peripheral outputs are related to a common pulse.

Figure 39 shows the means for isochronous mode; in particular the common pulse is defined by the start of the data cycle (T_DC), T_IO_Input relates peripheral inputs and T_IO_Output relates peripheral outputs to the data cycle. The isochronous application within the IO controller is related to the data cycle by local means.

For increasing the transmission's robustness data loss is accepted even within an equidistant fieldbus. The data hold time (as configured by the user) defines the maximum number of data losses accepted. A typical value is 3 * data actualization time (the default, however configurable by the user), which accepts two losses of data.

Thus a data actuality window is opened. The application per se does not know, whether a specific peripheral input value is related to this current data cycle or to some past data cycle. This applies also to a specific peripheral output; again the output does not know, whether the data value to be set is out of this current data cycle or from some past data cycle. The actuality of data may be spread over 2 data cycles without getting any notice of the peripheral bus system.

If on base of a data actuality window an isochronous application is to be modeled, the application needs means to uncover whether data is from the current cycle or some past cycle. For this the application cycles are synchronized by application means: the so called “**Life Sign**” (Master Life Sign and Slave Life Sign).

Figure 39 shows the relation of the class attributes within the isochronous application model.

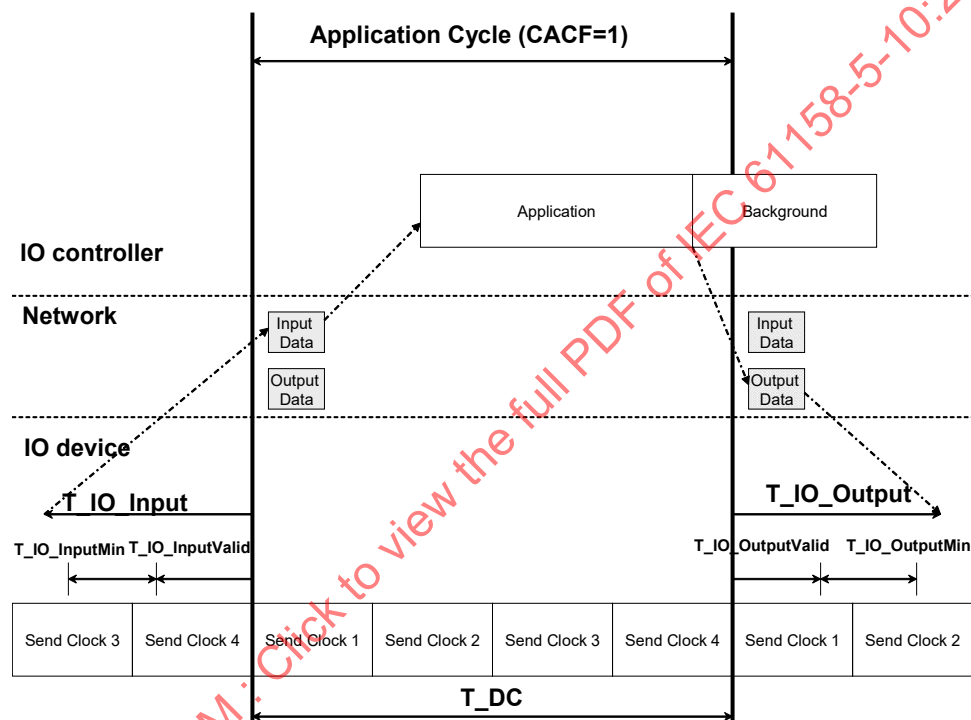


Figure 39 – General isochronous application model (example CACF == 1)

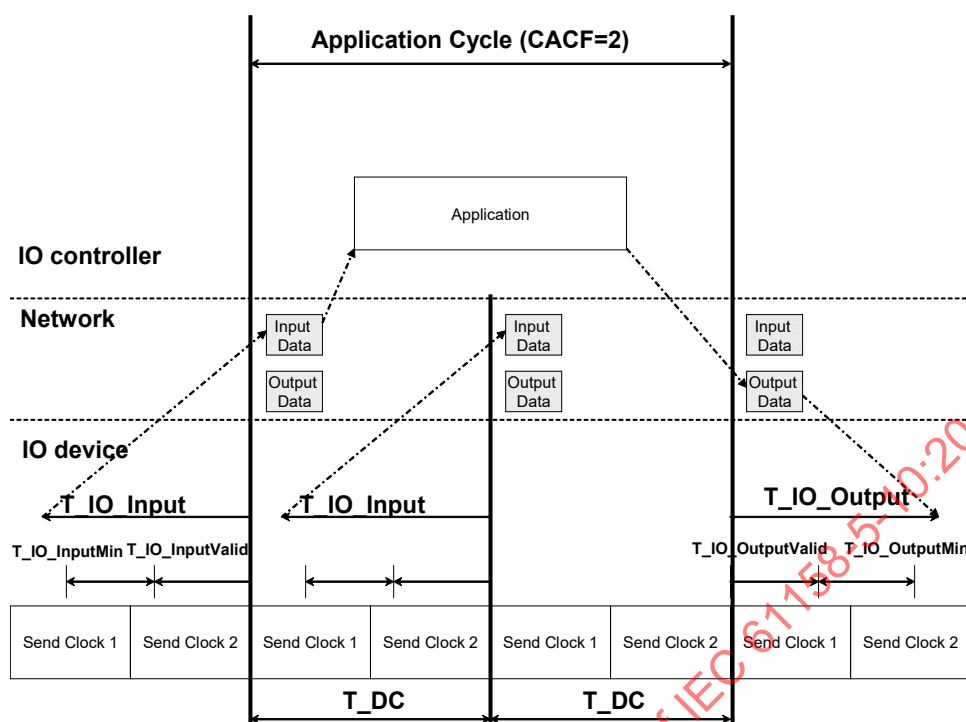


Figure 40 – General isochronous application model (example CACF == 2)

The application in the IO controller is started with a reduction to the data cycle (CACF). It may thus run in parallel to the data cycle (if CACF == 1) or reduced to the data cycle (if CACF > 1). The controller application provides new output values every controller application cycle ($CACF * T_{DC} * T_{DC\_BASE} * 31,25 \mu s$).

The data from/for a particular submodule is transported via RTC data services in every data cycle (T_{DC}). The application in the IO device runs in every data cycle; thus in every data cycle new input values are available for the IO controller ($T_{DC} * T_{DC\_BASE} * 31,25 \mu s$).

New output values are optionally indicated by the IO controller by a change of the MLS (master life sign) which is a part of the output data. The MLS changes thus in every controller application cycle ($CACF * T_{DC} * T_{DC\_BASE} * 31,25 \mu s$). The values are sampled by the IO device at T_{IO_Output} after the data cycle begins, when the new data was being received.

New input values are optionally indicated by the IO device by a change of the SLS (slave life sign) which is a part of the input data. The SLS changes thus in every data cycle ($T_{DC} * T_{DC\_BASE} * 31,25 \mu s$). The values are sampled by the IO device at T_{IO_Input} before the next data cycle begins.

7.3.2.4.2 Relations

The services are bound to other services as shown in Table 212:

Table 212 – Binding Isochronous Mode Application services

Service	Binding	Usage
Write Isochronous Mode Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read Isochronous Mode Data	AR:Record Data:Read	used by IO controller or IO supervisor

7.3.2.4.3 Template

An Isochronous Mode Application process object is described by the following template:

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 IsoM Submodules
2.1	(m) Attribute: API
2.2	(m) Attribute: Slot Number
2.3	(m) Attribute: Subslot Number
2.4	(m) Attribute: IsoM Data
2.4.1	(m) Attribute: Time Data Cycle
2.4.2	(m) Attribute: Time IO Input
2.4.3	(m) Attribute: Time IO Output
2.4.4	(m) Attribute: Time IO Input Valid
2.4.5	(m) Attribute: Time IO Output Valid
2.4.6	(m) Attribute: Controller Application Cycle Factor
3.	(m) Attribute: Device Parameter
3.1	(m) Attribute: Time Data Cycle Base
3.2	(m) Attribute: Time IO Base
3.3	(m) Attribute: Time Data Cycle Min
3.4	(m) Attribute: Time Data Cycle Max
3.5	(m) Attribute: Time IO Input Min
3.6	(m) Attribute: Time IO Output Min
SERVICES:	
1	(m) OpsService: Write Isochronous Mode Data
2	(m) OpsService: Read Isochronous Mode Data

7.3.2.4.4 Attributes

Implicit

The key attribute Implicit indicates that the Isochronous Mode Application process object is implicitly addressed by the service.

List of IsoM Submodules

This attribute list contains the following attributes:

API

This attribute defines the number of the application process the submodule is assigned to.

Slot Number

This attribute defines the slot the submodule is assigned to.

Subslot Number

This attribute defines the subslot the submodule is assigned to.

IsoM Data

This attribute consists of the following attributes:

Time Data Cycle (T_DC)

This attribute indicates the time factor for the data cycle. This time includes all parts of the isochronous data cycle to convey inputs and outputs.

The time of the data cycle is calculated by $T\_DC * T\_DC\_BASE * 31,25 \mu s$.

Time base: $T\_DC\_BASE * 31,25 \mu s$

Attribute type: Unsigned16

Allowed values: 1 to 1 024

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 (T_IO_Input) is related to the start of T_DC and describes a time **before** the start of T_DC.

T_IO_Input has a step size of T_IO_Base; i.e. $T\_IO\_Input = n * T\_IO\_BASE$.

For T_IO_Input the following relation shall be applied:

$$(T\_IO\_InputValid + T\_IO\_InputMin) \leq T\_IO\_Input \leq T\_DC$$

Time base: 1 ns

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

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 (T_IO_Output) is related to the start of T_DC and describes a time **after** the start of T_DC.

T_IO_Output has a step size of T_IO_Base; i.e. $T\_IO\_Output = n * T\_IO\_BASE$.

For T_IO_Output the following relation shall be applied:

$$(T\_IO\_OutputValid + T\_IO\_OutputMin) \leq T\_IO\_Output \leq T\_DC$$

Time base: 1 ns

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

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 and describes a time **before** the start of T_DC.

T_IO_InputValid has a step size of T_IO_Base; i.e. $T\_IO\_InputValid = n * T\_IO\_BASE$.

Time base: 1 ns

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

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 and describes a time **after** the start of T_DC.

T_IO_OutputValid has a step size of T_IO_Base; i.e. $T\_IO\_OutputValid = n * T\_IO\_BASE$.

Time base: 1 ns

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

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 the application.

The isochronous application task of the IO controller is started with a local time offset (the so called T_CA_START) from the cycle begin, where global cycle counter modulo $(T\_DC * CACF) == 0$.

Time base: T_DC

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Device Parameter

This attribute consists of the following attributes.

Time Data Cycle Base (T_DC_Base)

This attribute determines the granularity for data cycle adjustment.

This field is part of the GSD.

Time base: 31,25 µs

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time IO Base (T_IO_Base)

This attribute determines the granularity for input and output delay time adjustment.

The time for T_IO_Base is calculated by $T\_IO\_Base * 1 \text{ ns}$.

This field is part of the GSD.

Time base: 1 ns

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time Data Cycle Min (T_DC_Min)

This attribute determines the smallest possible data cycle.

The time for T_DC_Min is calculated by $T\_DC\_Min * T\_DC\_Base * 31,25 \text{ µs}$.

This field is part of the GSD.

Time base: $T\_DC\_Base * 31,25 \text{ µs}$

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time Data Cycle Max (T_DC_Max)

This attribute determines the longest possible data cycle.

The time for T_DC_Max is calculated by $T\_DC\_Max * T\_DC\_Base * 31,25 \mu s$.

This field is part of the GSD.

Time base: $T\_DC\_Base * 31,25 \mu s$

Attribute type: Unsigned16

Allowed values: 1 to 1 024

Time IO Input Min (T_IO_InputMin)

This attribute contains the factor for the smallest possible input refresh delay time. It may be the smallest possible time to copy or preprocess the input data.

This point of time is related to the start of T_DC and describes a time **before** the start of T_DC.

The time for T_IO_InputMin is calculated by $T\_IO\_InputMin * T\_IO\_Base * 1 ns$.

This field is part of the GSD.

Time base: $T\_IO\_Base * 1 ns$

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

Time IO Output Min (T_IO_OutputMin)

This attribute contains the factor for the smallest possible output refresh delay time. It may be the smallest possible time to copy or preprocess the output data.

This point of time is related to the start of T_DC and describes a time **after** the start of T_DC.

The time for T_IO_OutputMin is calculated by $T\_IO\_OutputMin * T\_IO\_Base * 1 ns$.

This field is part of the GSD.

Time base: $T\_IO\_Base * 1 ns$

Attribute type: Unsigned32

Allowed values: 1 to 32 000 000

7.3.2.4.5 Services**7.3.2.4.5.1 Write Isochronous Mode Data**

This service allows writing Isochronous Mode Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 213 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 213 – Data elements of Write Isochronous Mode Data

Parameter name	service specific elements	req	ind
Data	Isochronous Mode 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(=)

Data

This parameter contains the values of the object which may be used to write.

Isochronous Mode 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.

7.3.2.4.5.2 Read Isochronous Mode Data

This service allows reading Isochronous Mode Data using a record data object with subslot scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 213 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write Isochronous Mode Data (7.3.2.4.5.1).

7.3.2.4.6 Behavior**7.3.2.4.6.1 Overview of relations to other ASE objects**

For performance reasons, the value for the Output Data object shall 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.

The Isochronous Mode Application model is a superset of the model described in IEC 61158-5-3.

Figure 41 shows the timing relations between different involved ASEs for a particular submodule.

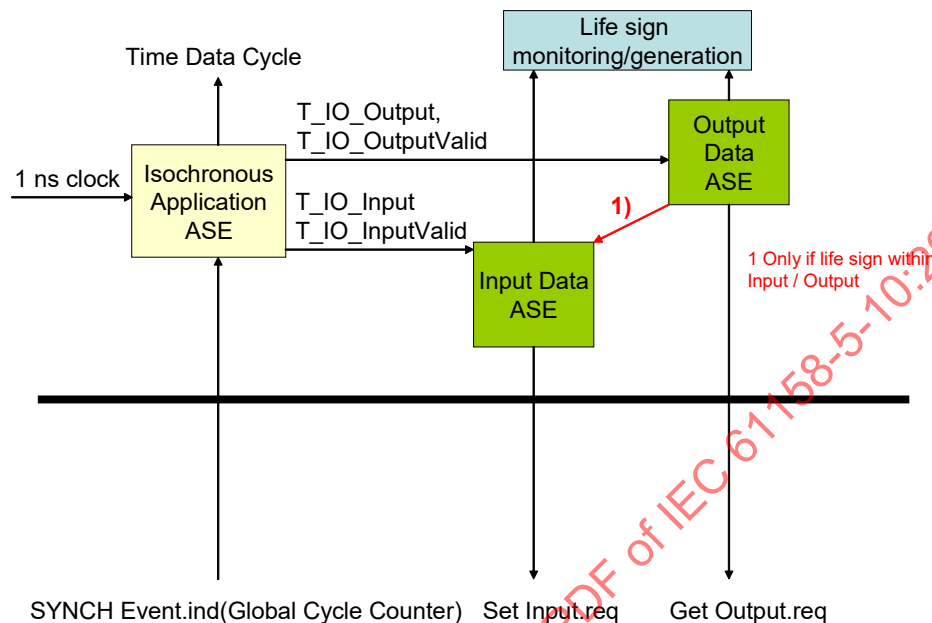


Figure 41 – ASE relations in an IO device operating in isochronous mode for a submodule

7.3.2.4.6.2 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 PTCP 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 an IO system with enabled isochronous mode functionality IO devices with basic functionality can be used in conjunction with synchronized IO devices:

- if the SYNCH-Event.ind is issued with the status “REMOTE” then the application is synchronized with other devices,
- if the SYNCH-Event.ind is issued with the status “LOCAL” then the isochronous application shall be informed that the application may be not synchronized with other devices.

7.3.2.4.6.3 State machine description for isochronous mode

7.3.2.4.6.3.1 General

The behavior of an Output Data object in case of Isochronous Mode is defined by the Isochronous Mode Output Data state machine (ISOM_OUT) state machine.

The ISOM_OUT is triggered by the Isochronous Mode Sync state machine (ISOM_SYNC).

The ISOM_OUT and ISOM_SYNC state machine represents a part of the User functionality and are included to ensure interoperability.

The ISOM_OUT and ISOM_SYNC state machine provides events to other parts of the User functionality. The ISOM_SYNC state machine has interfaces to the Isochronous Mode Input Data state machine (ISOM_IN) and the ISOM_OUT state machine for co-ordination purposes.

The main functionalities of the ISOM_OUT state machine are:

- Check the correct sequence: SYNCH, New Output, read out the Output Data at Time IO Output.
- Transfer the received Output Data to the application on time.
- Check the Controllers Sign of Life (C-LS).

The main functionalities of the ISOM_SYNC 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.

7.3.2.4.6.3.2 ISOM_SYNC

7.3.2.4.6.3.2.1 Primitive definitions

7.3.2.4.6.3.2.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by ISOM_SYNC state machine are described in the service definition and shown in Table 214.

Table 214 – Remote primitives issued or received by ISOM_SYNC

Primitive	Source	Destination	Associated parameters	Functions
Local SYNCH Event.ind	FAL	ISOM_SYNC	see 7.3.3.3.5.4	—

7.3.2.4.6.3.2.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by ISOM_SYNC state machine are described in the service definition and shown in Table 215.

Table 215 – Local primitives issued or received by ISOM_SYNC

Primitive	Source	Destination	Associated parameters	Functions
SYNCH_Status_ind	ISOM_SYNC	ISOM_IN, ISOM_OUT	state	—
Start_O_ind	ISOM_SYNC	ISOM_OUT	—	—
SYNCH_I_ind	ISOM_SYNC	ISOM_IN	—	—
SYNCH_O_ind	ISOM_SYNC	ISOM_OUT	—	—

7.3.2.4.6.3.2.2 State transition diagram

Figure 42 shows the State transition diagram of the ISOM_SYNC state machine.

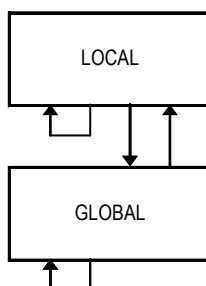


Figure 42 – State transition diagram of ISOM_SYNC

States of ISOM_SYNC

LOCAL

The clock is driven by the local time system

GLOBAL

The clock is driven by the global time system, synchronized for the corresponding devices

7.3.2.4.6.3.2.3 State machine description

This state machine indicates the ISOM_IN and ISOM_OUT state machines the state of the SYNCH signal (LOCAL or REMOTE) and starts the ISOM_IN and ISOM_OUT state machines.

7.3.2.4.6.3.2.4 State table

Table 216 shows the states of an ISOM_SYNC state machine.

Table 216 – State table ISOM_SYNC

#	Current State	Event /Condition => Action	Next State
1	LOCAL	Local SYNCH Event.ind () /Status == LOCAL => SYNCH_Status_ind (LOCAL)	LOCAL
2	LOCAL	Local SYNCH Event.ind () /Status == GLOBAL && (GlobalCycleCounter mod SendClock*ReductionRatio) != 0 => ignore	GLOBAL
3	LOCAL	Local SYNCH Event.ind () /Status == GLOBAL && (GlobalCycleCounter mod SendClock*ReductionRatio) == 0 => Start_I_ind Start_O_ind SYNCH_I_ind SYNCH_O_ind SYNCH_Status_ind (GLOBAL)	GLOBAL
4	GLOBAL	Local SYNCH Event.ind () /Status == LOCAL => ignore	LOCAL
5	GLOBAL	Local SYNCH Event.ind () /Status == GLOBAL && (GlobalCycleCounter mod SendClock*ReductionRatio) != 0 => ignore	GLOBAL

#	Current State	Event /Condition =>Action	Next State
6	GLOBAL	Local SYNCH Event.ind () /Status == GLOBAL && (GlobalCycleCounter mod SendClock*ReductionRatio) == 0 => Start_I_ind Start_O_ind SYNCH_I_ind SYNCH_O_ind SYNCH_Status_ind (GLOBAL)	GLOBAL

7.3.2.4.6.3.2.5 Functions, Macros, Timers and Variables

Table 217 contains the functions, macros, timers and variables used by ISOM_SYNC state machine and their arguments and descriptions.

Table 217 – Functions, Macros, Timers and Variables used by the ISOM_SYNC

Name	Type	Function/Meaning
—	—	—

7.3.2.4.6.3.3 ISOM_OUT

7.3.2.4.6.3.3.1 Primitive definitions

7.3.2.4.6.3.3.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by ISOM_OUT machine are described in the service definition and shown in Table 218.

Table 218 – Remote primitives issued or received for ISOM_OUT

Primitive	Source	Destination	Associated parameters	Functions
Alarm.req	ISOM_OUT	FAL	see 7.3.1.5.5	—

7.3.2.4.6.3.3.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by ISOM_OUT state machine are described in the service definition and shown in Table 219.

Table 219 – Local primitives issued or received for ISOM_OUT

Primitive	Source	Destination	Associated parameters	Functions
Local Get Output_cnf	FAL	ISOM_OUT	see 7.3.1.3.5.8	Confirms Get_Output request and passes new output data to output state machine
Local New Output_ind	FAL	ISOM_OUT	see 7.3.1.3.5.10	Indicates new output data has been received
Local Get Output_req	ISOM_OUT	FAL	see 7.3.1.3.5.8	
Start_O.ind	ISOM_SYNC	ISOM_OUT	—	Start Output Data processing
Stop_O.ind	ISOM_SYNC	ISOM_OUT	—	Stop Output Data processing
SYNCH_O.ind	ISOM_SYNC	ISOM_OUT	—	Trigger Signal for Output Data Processing

7.3.2.4.6.3.3.2 State transition diagram

Figure 43 shows the State transition diagram of the ISOM_OUT state machine.

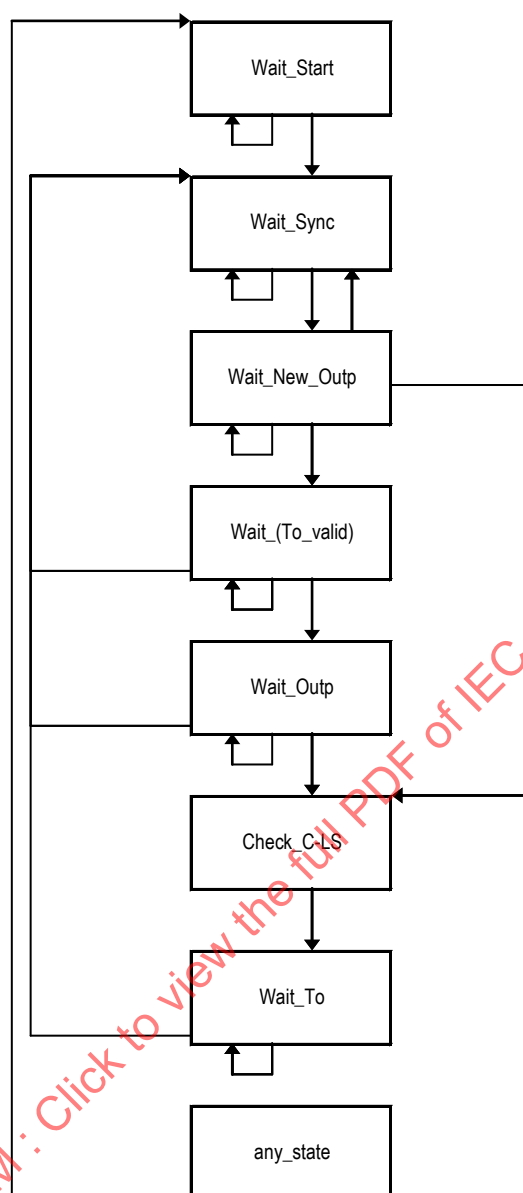


Figure 43 – State transition diagram ISOM_OUT

States of the ISOM_OUT state machine

Wait_Start	Waits for the start of the state machine.
Wait_Sync	Waits for the PLL being is sync.
Wait_New_Outp	Wait for a new output indication.
Wait_(To_valid)	Waits for the expiration of the To_Valid timer.
Wait_To	Waits for the expiration of the To timer.
Wait_Outp	Waits for the return of the Get:output request.
Check_C-LS	Checks the controller life sign.
any_state	Any state of the above.

7.3.2.4.6.3.3.3 State machine description

This state machine drives the output of a particular isochronous submodule according to the SYNCH signal as being provided by the ISOM_SYNC state machine and according to the

T_IO_OutputValid and T_IO_Output times as being configured by the Isochronous Mode data record.

The state machine issues alarms, whenever the outputs are not being received in time (e.g. after T_IO_OutputValid) or else the outputs have not been given to the application in time (e.g. after T_IO_Output).

7.3.2.4.6.3.3.4 State table

Table 220 shows the states of an ISOM_OUT state machine.

Table 220 – State table ISOM_OUT

#	Current State	Event /Condition =>Action	Next State
1	Wait_Start	Start_O.ind => O_Buffer:=NIL wrong_output_upd:=0	Wait_Sync
2	Wait_Start	OTHERS => ignore	Wait_Start
3	Wait_Sync	SYNCH_O.ind => StartTimer (To_valid) StartTimer (To)	Wait_New_Outp
4	Wait_Sync	OTHERS => ignore	Wait_Sync
5	Wait_New_Outp	Local New Output_ind () /wrong_output_upd != 1 => if (wrong_output_upd>0) wrong_output_upd:= wrong_output_upd-1 // NOTE No sequence alarm is outstanding or still in trouble with the outputs	Wait_(To_valid)
6	Wait_New_Outp	Local New Output_ind () /wrong_output_upd == 1 => wrong_output_upd := 0 // NOTE output sequence trouble is over; take back diagnosis IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, NoFault)	Wait_(To_valid)
7	Wait_New_Outp	TimerExpired (To_valid) /wrong_output_upd < limit => wrong_output_upd:= wrong_output_upd+n // NOTE output sequence error; not issued since below limit	Check_C-LS
8	Wait_New_Outp	TimerExpired (To_valid) /wrong_output_upd >= limit => // NOTE Sequence Error is issued, since output sequence errors are above limit IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Check_C-LS
9	Wait_New_Outp	TimerExpired (To) /wrong_output_upd < limit => wrong_output_upd:= wrong_output_upd+n // NOTE output sequence error; not issued since below limit	Wait_Sync

#	Current State	Event /Condition =>Action	Next State
10	Wait_New_Outp	TimerExpired (To) /wrong_output_upd >= limit => // NOTE Sequence Error is issued, since output sequence errors are above limit IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Wait_Sync
11	Wait_New_Outp	OTHERS => ignore	Wait_New_Outp
12	Wait_(To_valid)	TimerExpired (To_valid) => Local Get Output_req ()	Wait_Outp
13	Wait_(To_valid)	TimerExpired (To) /wrong_output_upd < limit => wrong_output_upd:= wrong_output_upd+n // NOTE output sequence error; not issued since below limit	Wait_Sync
14	Wait_(To_valid)	TimerExpired (To) /wrong_output_upd >= limit && ALARM_PENDING == FALSE => // NOTE Sequence Error is issued, since output sequence errors are above limit IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Wait_Sync
15	Wait_(To_valid)	Local New Output_ind () /wrong_output_upd < limit => wrong_output_upd:= wrong_output_upd+n // NOTE output sequence error; not issued since below limit	Wait_(To_valid)
16	Wait_(To_valid)	Local New Output_ind () /wrong_output_upd >= limit => // NOTE Sequence Error is issued, since output sequence errors are above limit IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Wait_(To_valid)
17	Wait_(To_valid)	OTHERS => ignore	Wait_(To_valid)
18	Wait_Outp	Local Get Output_cnf () /New_Flag => O_Buffer := Subslot_Output_Data	Check_C-LS
19	Wait_Outp	Local Get Output_cnf () /!New_Flag => ignore // NOTE old value is kept	Check_C-LS
20	Wait_Outp	TimerExpired (To) /wrong_output_upd < limit => wrong_output_upd:= wrong_output_upd+n // NOTE output sequence error; not issued since below limit	Wait_Sync
21	Wait_Outp	TimerExpired (To) /wrong_output_upd >= limit => // NOTE Sequence Error is issued, since output sequence errors are above limit IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Wait_Sync
22	Wait_Outp	OTHERS => ignore	Wait_Outp

#	Current State	Event /Condition =>Action	Next State
23	Check_C-LS	/CheckC-LS () == VALID => // NOTE O_Buffer is either new or old (in case of errors)	Wait_To
24	Check_C-LS	/CheckC-LS () == INVALID => O_Buffer := substitute value IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Wait_To
25	Wait_To	TimerExpired (To) => SYNCH_OUT	Wait_Sync
26	Wait_To	Local New Output_ind () /wrong_output_upd < limit => wrong_output_upd:= wrong_output_upd+n // NOTE output sequence error; not issued since below limit	Wait_To
27	Wait_To	Local New Output_ind () /wrong_output_upd >= limit => // NOTE Sequence Error is issued, since output sequence errors are above limit IssueAlarm (IsochroneModeAlarm, OutputTimeFailure, Fault)	Wait_To
28	Wait_To	OTHERS => ignore	Wait_To
29	any_state	Stop_O.ind =>	Wait_Start

7.3.2.4.6.3.3.5 Functions, Macros, Timers and Variables

Table 221 contains the functions, macros, timers and variables used by ISOM_OUT state machine and their arguments and descriptions.

Table 221 – Functions, Macros, Timers and Variables used by the ISOM_OUT

Name	Type	Function/Meaning
To	Timer	Timer to control the takeover of the data by the application.
To_valid	Timer	Timer to control the data being available from the communication.
SYNCH_OUT	Function	Pass the Output Buffer to the Output User Data
CheckC-LS	Function	Checks the controller life sign according IEC 61800-7-203:2015 6.3.12 "user data reliability". VALID and INVALID indicates the validity of the life sign according to the rules described there. If the use of the controller life sign is not intended by the submodule, VALID is returned always.
IssueAlarm	Function	The local diagnosis is updated to the current state and a change of the state is issued as alarm. The alarm confirmation is handled internally by the function. If there is a congestion of alarms, alarms may not be send if they nullify each other.
StartTimer	Function	Starts a timer
TimerExpired	Function	Indication of a timer's expiration

7.3.2.4.6.3.4 ISOM_IN

7.3.2.4.6.3.4.1 Primitive definitions

7.3.2.4.6.3.4.1.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by ISOM_IN state machine are described in the service definition and shown in Table 222.

Table 222 – Remote primitives issued or received for ISOM_IN

Primitive	Source	Destination	Associated parameters	Functions
—	—	—	—	—

7.3.2.4.6.3.4.1.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by ISOM_IN state machine are described in the service definition and shown in Table 223.

Table 223 – Local primitives issued or received for ISOM_IN

Primitive	Source	Destination	Associated parameters	Functions
Local Set Input_cnf ()	FAL	ISOM_IN	see 7.3.1.3.5.1	Confirms Set_Input request
Local Set Input_req	ISOM_IN	FAL	see 7.3.1.3.5.1	Passes application input data to FAL
Start_I.ind	ISOM_SYNC	ISOM_IN	—	Start Input Data processing
Stop_I.ind	ISOM_SYNC	ISOM_IN	—	Stop Input Data processing
SYNCH_I.ind	ISOM_SYNC	ISOM_IN	Status	Trigger Signal for Input Data Processing

7.3.2.4.6.3.4.2 State transition diagram

Figure 44 shows the State transition diagram of the ISOM_IN state machine.

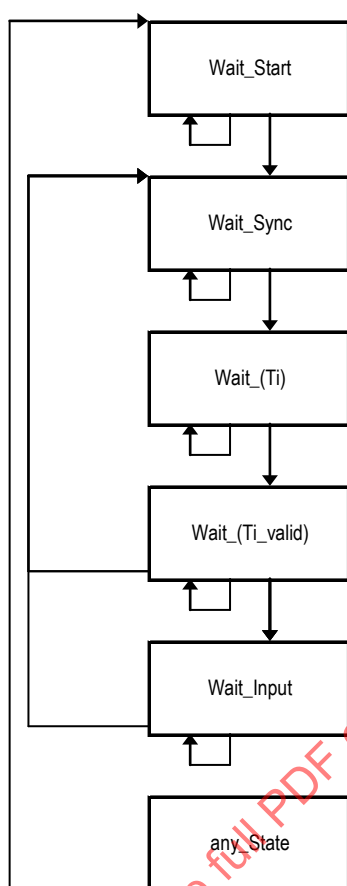


Figure 44 – State transition diagram ISOM_IN

States of the ISOM_IN state machine

Wait_Start	Waits for the start of the state machine.
Wait_Sync	Waits for the PLL being is sync.
Wait_(Ti)	Waits for the expiration of the Ti timer.
Wait_(Ti_valid)	Waits for the expiration of the Ti_valid timer.
Wait_Input	Waits for the return of the Set_input request.
any_State	Any state of the above.

7.3.2.4.6.3.4.3 State machine description

This state machine drives the input of a particular isochronous submodule according to the SYNCH signal as being provided by the ISOM_SYNC state machine and according to the T_IO_InputValid and T_IO_Input times as being configured by the IsoM data record.

The state machine issues alarms, whenever the inputs are not being ready for transmission in time (i.e. after T_IO_InputValid).

7.3.2.4.6.3.4.4 State table

Table 224 shows the states of an ISOM_IN state machine.

Table 224 – State table ISOM_IN

#	Current State	Event /Condition =>Action	Next State
1	Wait_Start	Start_I.ind => SLS:=0 ALARM_PENDING := FALSE wrong_input_upd:=0	Wait_Sync
2	Wait_Start	OTHERS => ignore	Wait_Start
3	Wait_Sync	SYNCH_I.ind(Output_State) => StartTimer (Ti) StartTimer (Ti_valid) Store Output_State	Wait_(Ti)
4	Wait_Sync	OTHERS => ignore	Wait_Sync
5	Wait_(Ti)	TimerExpired (Ti) => SYNCH_IN // NOTE Fills the I_Buffer from the application	Wait_(Ti_valid)
6	Wait_(Ti)	OTHERS => ignore	Wait_(Ti)
7	Wait_(Ti_valid)	TimerExpired (Ti_valid) /CheckLS () == VALID && I_Buffer != NIL => Input_Data:= I_Buffer inc_LS(SLS) Input_Data.SLS:= SLS if (wrong_input_upd>0) wrong_input_upd= wrong_input_upd-1 if (wrong_input_upd==0) State := NoFault else State := Fault Local Set Input.req () IssueAlarm (IsochroneModeAlarm, InputTimeFailure, State)	Wait_Input
8	Wait_(Ti_valid)	TimerExpired (Ti_valid) /CheckLS () == INVALID && I_Buffer != NIL => Input_Data:= I_Buffer SLS := 0 Input_Data.SLS:= SLS if (wrong_input_upd>0) wrong_input_upd= wrong_input_upd-1 if (wrong_input_upd==0) State := NoFault else State := Fault Local Set Input.req () IssueAlarm (IsochroneModeAlarm, InputTimeFailure, State)	Wait_Input

#	Current State	Event /Condition =>Action	Next State
9	Wait_(Ti_valid)	TimerExpired (Ti_valid) /CheckLS () == NONE && I_Buffer != NIL => Input_Data:= I_Buffer if (wrong_input_upd>0) wrong_input_upd= wrong_input_upd-1 if (wrong_input_upd==0) State := NoFault else State := Fault Local Set Input.req () IssueAlarm (IsochroneModeAlarm, InputTimeFailure, State)	Wait_Input
10	Wait_(Ti_valid)	TimerExpired (Ti_valid) /I_Buffer == NIL && wrong_input_upd >= limit // NOTE Application didn't return buffer and input sequence errors above limit => IssueAlarm (IsochroneModeAlarm, InputTimeFailure, Fault)	Wait_Sync
11	Wait_(Ti_valid)	TimerExpired (Ti_valid) /I_Buffer == NIL && wrong_input_upd < limit => wrong_input_upd= wrong_input_upd+n	Wait_Sync
12	Wait_(Ti_valid)	OTHERS => ignore	Wait_(Ti_valid)
13	Wait_Input	Local Set Input.cnf () =>	Wait_Sync
14	Wait_Input	OTHERS => ignore	Wait_Input
15	any_State	Stop_I.ind =>	Wait_Start

7.3.2.4.6.3.4.5 Functions, Macros, Timers and Variables

Table 225 contains the functions, macros, timers and variables used by ISOM_IN state machine and their arguments and descriptions.

Table 225 – Functions, Macros, Timers and Variables used by the ISOM_IN

Name	Type	Function/Meaning
Ti	Timer	Timer to control the takeover of the data from the application.
Ti_valid	Timer	Timer to control the data being available for the communication.
SYNCH_IN	Function	Samples and collects the Values of the Input User Data
IssueAlarm	Function	The local diagnosis is updated to the current state and a change of the state is issued as alarm. The alarm confirmation is handled internally by the function. If there is a congestion of alarms, alarms may not be send if they nullify each other.
StartTimer	Function	Starts a timer
TimerExpired	Function	Indication of a timer's expiration
SLS	Variable	Submodule Life Sign

7.3.3 Communication Interface Management ASE

7.3.3.1 Overview

This ASE specifies the structure of the necessary parameters for the used protocols, e.g. IEEE 802.3, IP, UDP, RPC, or DCP, which are inherited by common ASEs. The Communication Interface Management ASE manages the parameter in a common database. It represents the physical device as shown in Figure A.1. The Communication Interface Management 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. Without a Station Name the operation of an IO device is not permitted.

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 in general, and additionally a non-volatile IP address for fast start-up mode. The DCP server is only passive in this procedure.

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.

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 can be 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 Autoconfiguration Data enables a particular IO controller to utilize an IO device without an explicit configuration in an engineering system.

Furthermore, the Communication Interface Management ASE defines specific object attributes and access methods to handle interface and port submodules. Besides the address resolution

and neighborhood detection, these submodules contain additional attributes e.g. for WORKING_CLOCK synchronization and RT_CLASS_3 routing information.

A check of the remote system data shall be done during startup and operation. The real remote system data of the peer connected to a port are retrieved by LLDP requests. So, the check of remote system data shall be done after every reception of a LLDP indication.

To run RT_CLASS_3 communication relationships between devices within a RT_CLASS_3 domain, the conformity between the engineered and the real peers of every device within the domain shall be guaranteed.

7.3.3.2 Communication Interface Management class

7.3.3.2.1 Overview

See 7.3.3.1.

7.3.3.2.2 Relations

The services are bound to other services as shown in Table 226:

Table 226 – Binding Communication Interface Management services

Service	Binding	Usage
Read PDev Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Real Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Expected Data	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Interface Data Real	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Interface Adjust	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Interface Adjust	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Interface FSU Data Adjust	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Interface FSU Data Adjust	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD NC Data Check	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD NC Data Check	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Port Data Real	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Port Data Check	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Port Data Check	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Port Data Adjust	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Port Data Adjust	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Port FO Data Real	AR:Record Data:Read	used by IO controller or IO supervisor

Service	Binding	Usage
Write PD Port FO Data Check	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Port FO Data Check	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Port FO Data Adjust	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Port FO Data Adjust	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Port Statistic	AR:Record Data:Read	used by IO controller or IO supervisor

7.3.3.2.3 Template

A Communication Interface Management object is described by the following template:

ASE: Communication Interface Management ASE
CLASS: Communication Interface Management
CLASS ID: not used
PARENT CLASS: IEEE 802.3, IEEE 802.1AB, IP suite, DNS, DHCP, DCP

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: List of Interfaces
 - 2.1 (m) Attribute: API
 - 2.1 (m) Attribute: Slot Number
 - 2.2 (m) Attribute: Subslot Number
 - 2.3 (m) Attribute: PD Interface Data Real
 - 2.3.1 (m) Attribute: Chassis ID
 - 2.3.2 (m) Attribute: MAC Address
 - 2.3.3 (m) Attribute: IP Parameter Value
 - 2.3.3.1 (m) Attribute: IP Address
 - 2.3.3.2 (m) Attribute: Subnet Mask
 - 2.3.3.3 (m) Attribute: Standard Gateway
 - 2.4 (m) Attribute: PD Interface Data Adjust
 - 2.4.1 (m) Attribute: Multiple Interface Mode
 - 2.5 (o) Attribute: PD Interface FSU Data Adjust
 - 2.5.1 (o) Attribute: FS Hello Block
 - 2.5.1.1 (m) Attribute: FS Hello Mode
 - 2.5.1.2 (m) Attribute: FS Hello Interval
 - 2.5.1.3 (m) Attribute: FS Hello Retry
 - 2.5.1.4 (m) Attribute: FS Hello Delay
 - 2.5.2 (o) Attribute: Fast Startup Block
 - 2.5.2.1 (m) Attribute: Data
 - 2.6 (o) Attribute: PD NC Data Check
 - 2.6.1 (m) Attribute: Maintenance Demanded Drop Budget
 - 2.6.2 (m) Attribute: Maintenance Required Drop Budget
 - 2.6.3 (m) Attribute: Error Drop Budget
- 2.7 (m) Attribute: List of Ports
 - 2.7.1 (m) Attribute: API
 - 2.7.2 (m) Attribute: Slot Number
 - 2.7.3 (m) Attribute: Subslot Number
 - 2.7.4 (m) Attribute: Media Type
 - 2.7.5 (m) Attribute: Fiber Optic type

2.7.6	(m) Attribute:	Fiber Optic Cable type
2.7.7	(o) Attribute:	Fiber Optic Manufacturer Specific
2.7.7.1	(m) Attribute:	Vendor ID
2.7.7.2	(m) Attribute:	Manufacturer Specific Fiber Optic Data
2.7.8	(o) Attribute:	Power Budget Control
2.7.8.1	(m) Attribute:	Maintenance Demanded Power Budget
2.7.8.2	(m) Attribute:	Maintenance Required Power Budget
2.7.8.3	(m) Attribute:	Error Power Budget
2.7.9	(m) Attribute:	Preamble Length
2.7.10	(m) Attribute:	Domain Boundary
2.7.10.1	(m) Attribute:	Ingress
2.7.10.2	(m) Attribute:	Egress
2.7.11	(m) Attribute:	Multicast Boundary
2.7.12	(m) Attribute:	Peer To Peer Boundary
2.7.12.1	(m) Attribute:	LLDP
2.7.12.2	(m) Attribute:	PTCP
2.7.12.3	(m) Attribute:	Time
2.7.13	(m) Attribute:	DCP Boundary
2.7.14	(o) Attribute:	PD Port Data Real
2.7.14.1	(m) Attribute:	Own Port Name
2.7.14.2	(m) Attribute:	Line Delay
2.7.14.3	(m) Attribute:	MAU Type
2.7.14.4	(o) Attribute:	MAU Type Extension
2.7.14.5	(m) Attribute:	Link State
2.7.14.6	(m) Attribute:	RTClass3_PortStatus
2.7.14.7	(m) Attribute:	Number Of Peers
2.7.14.8	(m) Attribute:	List of Peers
2.7.14.8.1	(m) Attribute:	Peer Port Name
2.7.14.8.2	(m) Attribute:	Peer Station Name
2.7.14.8.3	(m) Attribute:	Line Delay
2.7.14.8.4	(m) Attribute:	Peer MAC Address

SERVICES:

1	(o) OpsService:	Read PDev Data
2	(m) OpsService:	Read PD Real Data
3	(m) OpsService:	Read PD Expected Data
4	(m) OpsService:	Read PD Interface Data Real
5	(m) OpsService:	Write PD Interface Adjust
6	(m) OpsService:	Read PD Interface Adjust
7	(o) OpsService:	Write PD Interface FSU Data Adjust
8	(o) OpsService:	Read PD Interface FSU Data Adjust
9	(o) OpsService:	Write PD NC Data Check
10	(o) OpsService:	Read PD NC Data Check
11	(m) OpsService:	Read PD Port Data Real
12	(m) OpsService:	Read PD Port Data Real Extended
13	(m) OpsService:	Write PD Port Data Check
14	(m) OpsService:	Read PD Port Data Check
15	(m) OpsService:	Write PD Port Data Adjust
16	(m) OpsService:	Read PD Port Data Adjust
17	(o) OpsService:	Read PD Port FO Data Real
18	(o) OpsService:	Write PD Port FO Data Check
19	(o) OpsService:	Read PD Port FO Data Check
20	(o) OpsService:	Write PD Port FO Data Adjust

21 (o) OpsService: Read PD Port FO Data Adjust

22 (m) OpsService: Read PD Port Statistic

7.3.3.2.4 Attributes

Implicit

The key attribute Implicit indicates that the Communication Interface Management object is implicitly addressed by the service.

List of Interfaces

This attribute list lists all interfaces related to the Communication Interface Management object. It contains the following attributes:

API

This attribute contains the API containing the interface submodule.

The interface submodule is always assigned to the default API.

Attribute type: Unsigned32

Allowed Values: 0x00000000

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 227.

Table 227 – Subslot number for interface submodules

Value (hexadecimal)	Meaning
0x8000	Interface Submodule 1
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
0x8800	Interface Submodule 9
0x8900	Interface Submodule 10
0x8A00	Interface Submodule 11
0x8B00	Interface Submodule 12
0x8C00	Interface Submodule 13
0x8D00	Interface Submodule 14
0x8E00	Interface Submodule 15
0x8F00	Interface Submodule 16

Value (hexadecimal)	Meaning
0x9000	Special case system redundancy R1/R2: Interface Submodule 1 on down or below delivering endnode
0x9100	Special case system redundancy R1/R2: Interface Submodule 2 on down or below delivering endnode
0x9200	Special case system redundancy R1/R2: Interface Submodule 3 on down or below delivering endnode
0x9300	Special case system redundancy R1/R2: Interface Submodule 4 on down or below delivering endnode
0x9400	Special case system redundancy R1/R2: Interface Submodule 5 on down or below delivering endnode
0x9500	Special case system redundancy R1/R2: Interface Submodule 6 on down or below delivering endnode
0x9600	Special case system redundancy R1/R2: Interface Submodule 7 on down or below delivering endnode
0x9700	Special case system redundancy R1/R2: Interface Submodule 8 on down or below delivering endnode
0x9800	Special case system redundancy R1/R2: Interface Submodule 9 on down or below delivering endnode
0x9900	Special case system redundancy R1/R2: Interface Submodule 10 on down or below delivering endnode
0x9A00	Special case system redundancy R1/R2: Interface Submodule 11 on down or below delivering endnode
0x9B00	Special case system redundancy R1/R2: Interface Submodule 12 on down or below delivering endnode
0x9C00	Special case system redundancy R1/R2: Interface Submodule 13 on down or below delivering endnode
0x9D00	Special case system redundancy R1/R2: Interface Submodule 14 on down or below delivering endnode
0x9E00	Special case system redundancy R1/R2: Interface Submodule 15 on down or below delivering endnode
0x9F00	Special case system redundancy R1/R2: Interface Submodule 16 on down or below delivering endnode

PD Interface Data Real

This attribute contains the following attributes:

Chassis ID

This attribute contains the Chassis ID according to the IEEE 802.1AB ASE.

Attribute type: VisibleString[240]

MAC Address

This attribute contains the value of the IEEE 802 MAC address of the interface.

Attribute type: OctetString[6]

IP Parameter Value

This parameter consists of the following elements:

IP Address

This attribute contains the current IP address of the interface according to IETF RFC 791.

Attribute type: Unsigned32

Subnet Mask

This attribute contains the current subnet mask according to IETF RFC 791.

Attribute type: Unsigned32

Standard Gateway

This attribute contains the current IP address of the standard gateway according to IETF RFC 791.

Attribute type: Unsigned32

PD Interface Data Adjust

This attribute contains the following attributes:

Multiple Interface Mode

This attribute contains the Multiple Interface Mode according to the IEEE 802.1AB ASE.

PD Interface FSU Data Adjust

This optional attribute contains the following attributes to control the interface's fast startup behavior:

FS Hello Block

This optional attribute contains the following attributes to control the DCP Hello service:

FS Hello Mode

This attribute contains the currently active model to convey the Hello service of the DCP ASE.

Attribute type: Unsigned32

Allowed values: OFF, ON_LINKUP, ON_LINKUP_AFTER_DELAY

Default value: OFF

FS Hello Interval

This attribute contains the interval between retries of the Hello service request.

Attribute type: Unsigned32

Allowed values: 30_MILLISECONDS, 50_MILLISECONDS, 100_MILLISECONDS, 300_MILLISECONDS, 500_MILLISECONDS, 1_SECOND

Default value: 30_MILLISECONDS

FS Hello Retry

This attribute contains the number of retries for the Hello request service.

Attribute type: Unsigned32

Allowed values: 1 to 15

Default value: 3

FS Hello Delay

This attribute contains the initial delay for the first Hello service request. The value is valid only if the attribute DCP Hello Mode is set to ON_LINKUP_AFTER_DELAY.

Attribute type: Unsigned32

Allowed values: NO_DELAY, 50_MILLISECONDS, 100_MILLISECONDS, 500_MILLISECONDS, 1_SECOND

Default value: NO_DELAY

Fast Startup Block

This optional attribute contains the following attributes:

Data

This attribute contains manufacturer specific fast startup data.

Attribute type: OctetString

PD NC Data Check

This optional attribute contains the following attributes:

Maintenance Demanded Drop Budget

This attribute contains the number of dropped frames that is accepted per second. If the number of dropped frames on a port per second exceeds this value, a Maintenance Demanded message is generated. The maintenance message can be deactivated.

Attribute type: Unsigned32

Allowed Values: DISABLED, Maintenance Demanded Drop Budget

Maintenance Required Drop Budget

This attribute contains the number of dropped frames that is accepted per second. If the number of dropped frames on a port per second exceeds this value, a Maintenance Required message is generated. The maintenance message can be deactivated.

Attribute type: Unsigned32

Allowed Values: DISABLED, Maintenance Required Drop Budget

Error Drop Budget

This attribute contains the number of dropped frames that is accepted per second. If the number of dropped frames on a port per second exceeds this value, a Diagnosis message is generated. The diagnosis message can be deactivated.

Attribute type: Unsigned32

Allowed Values: DISABLED, Error Drop Budget

List of Ports

This attribute list lists all ports related to the interface. It contains the following attributes:

API

This attribute contains the API containing the port submodule.

The port submodule is always assigned to the default API.

Attribute type: Unsigned32

Allowed Values: 0x00000000

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 228. The value for the parameter “i” in Table 228 shall correspond with the related interface submodule. Each interface may contain up to 255 ports.

The related interface module may be placed in a different slot.

Table 228 – Subslot number for port submodules

Value (hexadecimal)	Meaning
0x8i01	First Port Submodule ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i02	Port Submodule 2 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x8i03	Port Submodule 3 ($0 \leq i \leq F$; 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)
0x9i01	Special case system redundancy and R devices: First Port Submodule ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
0x9i02	Special case system redundancy and R devices: Port Submodule 2 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)
...	...
0x9iFF	Special case system redundancy and R devices: Port Submodule 255 ($0 \leq i \leq F$; shall correspond to the subslot number of the related interface submodule)

Media Type

This attribute contains the media type of the local port.

Attribute type: Unsigned16

Allowed Values: UNKNOWN, COPPER_CABLE, FIBER_OPTIC_CABLE, RADIO_COMMUNICATION

Fiber Optic type

This attribute is present only if the port's Media Type attribute is equal to FIBER_OPTIC_CABLE. It contains the type of the fiber optic media connected to the port.

Attribute type: Unsigned32

Allowed Values shall be used according to IEC 61158-6-10.

Fiber Optic Cable type

This attribute is present only if the port's Media Type attribute is equal to FIBER_OPTIC_CABLE. It contains the type of the fiber optic cable connected to the port.

Attribute type: Unsigned32

Allowed Values shall be used according to IEC 61158-6-10.

Fiber Optic Manufacturer Specific

This optional attribute is present only if the port's Media Type attribute is equal to FIBER_OPTIC_CABLE and if the vendor supports vendor specific fiber optic data. This attribute contains the following attributes

Vendor ID

This attribute shall contain the vendor ID.

Attribute type: Unsigned 16

Manufacturer Specific Fiber Optic Data

This attribute shall contain vendor specific fiber optic data

Power Budget Control

This optional attribute is present only if the port's Media Type attribute is equal to FIBER_OPTIC_CABLE and power budget control is supported by the port's hardware. This optional attribute controls the power budget. It contains the following attributes

Maintenance Demanded Power Budget

This attribute contains a power budget in 0,1 dB steps that is used to generate a Maintenance Demanded message if the current power budget of a port is less or equal to this value. The maintenance message can be deactivated.

Attribute type: Unsigned32

Unit: 0,1 dB

Allowed Values: DISABLED, Maintenance Demanded Power Budget

Maintenance Required Power Budget

This attribute contains a power budget in 0,1 dB steps that is used to generate a Maintenance Required message if the current power budget of a port is less or equal to this value. The maintenance message can be deactivated.

Attribute type: Unsigned32

Unit: 0,1 dB

Allowed Values: DISABLED, Maintenance Required Power Budget

Error Power Budget

This attribute contains a power budget in 0,1 dB steps that is used to generate a Diagnosis Message if the current power budget of a port is less or equal to this value. The diagnosis message can be deactivated.

Attribute type: Unsigned32

Unit: 0,1 dB

Allowed Values: DISABLED, Error Power Budget

Preamble Length

This attribute contains the number of preamble octets the PHY shall use before the start delimiter (SD) is send. A value of 7 preamble octets is the default (as defined for Ethernet); a value of 1 preamble octet indicates a shortening of the preamble and shall only be used, if both sending and receiving peer support preamble shortening as indicated by GSD.

Attribute type: Unsigned16

Allowed Values: 0x00 (seven octets preamble shall be used), 0x01 (one octet preamble shall be used)

Default Value: 0x00 (seven octets preamble shall be used)

Domain Boundary

This attribute consists of the following attributes:

Ingress

This attribute contains the value of the ingress synchronization domain boundary. Incoming frames with a destination mac address in the defined group will be blocked. The value shall be a list of allowed values for each multicast group.

Allowed values:

BLOCK_MULTICAST_ADDRESS_GROUP_0,
PASS_MULTICAST_ADDRESS_GROUP_0,

...

BLOCK_MULTICAST_ADDRESS_GROUP_31,
PASS_MULTICAST_ADDRESS_GROUP_31

The multicast groups are defined for the following IEEE 802 addresses:

ADDRESS_GROUP_0:

01-0E-CF-00-04-00,
01-0E-CF-00-04-20,
01-0E-CF-00-04-40,
01-0E-CF-00-04-80

ADDRESS_GROUP_1:

01-0E-CF-00-04-01,
01-0E-CF-00-04-21,
01-0E-CF-00-04-41,
01-0E-CF-00-04-81

...

ADDRESS_GROUP_31:

01-0E-CF-00-04-1F,
01-0E-CF-00-04-3F,
01-0E-CF-00-04-5F,
01-0E-CF-00-04-9F

ADDRESS_GROUP_0 shall be used for the PTCIP working clock, ADDRESS_GROUP_1 to ADDRESS_GROUP_31 are reserved according to IEC 61158-6-10.

Egress

This attribute contains the value of the egress synchronization domain boundary. Outgoing frames with a destination mac address in the defined group will be blocked. The allowed values shall be the same as for the Ingress attribute.

Multicast Boundary

This attribute contains the indication of the multicast boundary.

Attribute type: Unsigned32

Peer to Peer Boundary

This attribute contains the indication of peer to peer boundaries. It contains of the flowing attributes

LLDP

This attribute contains the indication of the LLDP boundary. If set to BLOCK, the IEEE 802.1AB shall not send LLDP frames (egress filter).

Attribute type: Unsigned8

Allowed values: PASS, BLOCK

PTCP

This attribute contains the indication of the PTCP boundary. If set to BLOCK, the Precision time control ASE shall not send PTCP_DELAY frames (egress filter).

Attribute type: Unsigned8

Allowed values: PASS, BLOCK

Time

This attribute contains the indication of the Time boundary. If set to BLOCK, the IEEE 802.1AS ASE shall not send PATH_DELAY frames (egress filter).

Attribute type: Unsigned8

Allowed values: PASS, BLOCK

DCP Boundary

This attribute contains the indication of the DCP multicast boundary.

Attribute type: Unsigned32

Allowed Values: PASS, BLOCK_HELLO, BLOCK_IDENTIFY, BLOCK_BOTH

PD Port Data Real

This attribute contains the current PD port data. It consists of the following attributes:

Own Port Name

This attribute contains the local port's name.

NOTE This value is independent from the old or new usage of the ChassisID and PortID in LLDP. The notation is always port-xyz or port-abcde-xyz.

Attribute type: VisibleString[255]

Line Delay

This attribute contains the value of the line delay as determined by this port.

The value 0 indicates that the line delay is unknown.

Attribute type: Unsigned32

Time Base: 1 ns

MAU Type

This attribute is defined in the IEEE 802.3 ASE.

MAU Type Extension

This optional attribute is defined in the IEEE 802.3 ASE.

Link State

This attribute is defined in the IEEE 802.3 ASE.

RTClass3_PortStatus

This attribute is defined in the IEEE 802.1AB ASE.

Number Of Peers

This attribute contains the number of peers contained in the following List of Peers.

Each device decides in its implementation the maximum number of peers contained in the list (recommended: up to 4).

Special case: If a device implementation supports 1 peer only, multiple neighbors are indicated by setting Number Of Peers to 2 and returning NULL (i.e. empty string...) in the second list entry.

Special case: If a device implementation supports more than 1 peer, but more than the supported number of peers are detected on the port, Number Of Peers is set to the number of supported peers.

If more peers than the maximum number of supported peers exist at a particular port, it is local behavior which peers are held in this list.

Attribute type: Unsigned8

Allowed values: 0 to 255

List of Peers

This attribute contains the actual list of peers.

It consists of the following attributes:

Peer Port Name

This attribute contains the value of the peer's port name.

Attribute type: VisibleString[255]

Peer Station Name

This attribute contains the value of the peer's station name.

Attribute type: VisibleString[255]

Time Base: 1 ns

Line Delay

This attribute contains the value of the line delay as determined by the peer.

The value 0 indicates that the line delay is unknown.

Attribute type: Unsigned32

Peer MAC Address

This attribute contains the value of the peer's MAC address.

Attribute type: OctetString[6]

7.3.3.2.5 Services

7.3.3.2.5.1 Read PDev Data

This service allows reading PDev Data using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 229 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 229 – Data elements of Read PDev Data

Parameter name	service specific elements	rsp	cnf
Data	PD IR Data	C	C(=)
	PD Sync Data	C	C(=)
	PD IR Subframe Data	C	C(=)
	PD Time Data	C	C(=)

Data

This parameter contains the values of the object which has been read.

PD IR Data

This conditional parameter is present only if the interface has PD IR data. It is composed analogous to the data as defined in 7.3.3.4.5.2.

PD Sync Data

This conditional parameter is present only if the interface has PD sync data. It is composed analogous to the data as defined in 7.3.3.3.5.2.

PD IR Subframe Data

This conditional parameter is present only if the interface has PD IR subframe data. It is composed analogous to the data as defined in 7.3.3.4.5.4.

PD Time Data

This conditional parameter is present only if the interface has PD IR subframe data. It is composed analogous to the data as defined in 7.3.3.6.5.2.

7.3.3.2.5.2 Read PD Real Data

This service allows reading PD Real Data using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 230 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 230 – Data elements of Read PD Real Data

Parameter name	service specific elements	rsp	cnf
Data	List of Submodules	M	M(=)
	Slot Number	M	M(=)
	Subslot Number	M	M(=)
	PD Port Data Real	C	C(=)
	PD Port Data Real Extended	C	C(=)
	PD Interface MRP Data Real	C	C(=)
	PD Port MRP Data Real	C	C(=)
	PD Port FO Data Real	C	C(=)
	PD Interface Data Real	C	C(=)
	PD Port Statistic	C	C(=)
	PD Port MRPIC Data Real	C	C(=)

Data

This parameter contains the values of the object which has been read.

List of Submodules

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.

PD Port Data Real

This conditional parameter is present only if the submodule is a port submodule. It is composed analogous to the data as defined in 7.3.3.2.5.11.

PD Port Data Real Extended

This conditional parameter is present only if the submodule is a port submodule.

PD Interface MRP Data Real

This conditional parameter is present only if the submodule is an interface submodule and supports MRP. It is composed analogous to the data as defined in 7.3.3.5.5.1.

PD Port MRP Data Real

This conditional parameter is present only if the submodule is a port submodule and supports MRP. It is composed analogous to the data as defined in 7.3.3.5.5.6.

PD Port FO Data Real

This conditional parameter is present only if the submodule is a fiber optic port submodule. It is composed analogous to the data as defined in 7.3.3.2.5.17.

PD Interface Data Real

This conditional parameter is present only if the submodule is an interface submodule. It is composed analogous to the data as defined in 7.3.3.2.5.4.

PD Port Statistic

This conditional parameter is present only if the submodule is a port submodule. It is composed analogous to the data as defined in 7.3.3.2.5.22.

PD Port MRPIC Data Real

This conditional parameter is present only if the submodule is a port submodule and supports MRP interconnection. It is composed analogous to the data as defined in 7.3.3.5.5.9.

7.3.3.2.5.3 Read PD Expected Data

This service allows reading PD Expected Data using a record data object with device scope (see 7.3.1.6.1).

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 231 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 231 – Data elements of Read PD Expected Data

Parameter name	service specific elements	rsp	cnf
Data	List of Submodules	M	M(=)
	Slot Number	M	M(=)
	Subslot Number	M	M(=)
	PD Port Data Check	C	C(=)
	PD Port Data Adjust	C	C(=)
	PD Interface MRP Data Check	C	C(=)
	PD Interface MRP Data Adjust	C	C(=)
	PD Port MRP Data Adjust	C	C(=)
	PD Port FO Data Check	C	C(=)
	PD Port FO Data Adjust	C	C(=)
	PD NC Data Check	C	C(=)
	PD Interface FSU Data Adjust	C	C(=)
	PD Interface Adjust	C	C(=)
	PD Port MRPIC Data Check	C	C(=)
	PD Port MRPIC Data Adjust	C	C(=)

Data

This parameter contains the values of the object which has been read.

List of Submodules

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.

PD Port Data Check

This conditional parameter is present only if the submodule is a port submodule and PD port data check has been written. It is composed analogous to the data as defined in 7.3.3.2.5.14.

PD Port Data Adjust

This conditional parameter is present only if the submodule is a port submodule and PD port data adjust has been written. It is composed analogous to the data as defined in 7.3.3.2.5.16.

PD Interface MRP Data Check

This conditional parameter is present only if the submodule is an interface submodule and PD interface MRP data check has been written. It is composed analogous to the data as defined in 7.3.3.5.5.3.

PD Interface MRP Data Adjust

This conditional parameter is present only if the submodule is an interface submodule and PD interface MRP data adjust has been written. It is composed analogous to the data as defined in 7.3.3.5.5.5.

PD Port MRP Data Adjust

This conditional parameter is present only if the submodule is an interface submodule and PD port MRP data adjust has been written. It is composed analogous to the data as defined in 7.3.3.5.5.7.

PD Port FO Data Check

This conditional parameter is present only if the submodule is a fiber optic port submodule and PD port FO data check has been written. It is composed analogous to the data as defined in 7.3.3.2.5.19.

PD Port FO Data Adjust

This conditional parameter is present only if the submodule is a fiber optic port submodule and PD port FO data adjust has been written. It is composed analogous to the data as defined in 7.3.3.2.5.21.

PD NC Data Check

This conditional parameter is present only if the submodule is an interface submodule and PD NC data check has been written. It is composed analogous to the data as defined in 7.3.3.2.5.10.

PD Interface FSU Data Adjust

This conditional parameter is present only if the submodule is an interface submodule and PD interface FSU data adjust has been written. It is composed analogous to the data as defined in 7.3.3.2.5.8.

PD Interface Adjust

This conditional parameter is present only if the submodule is an interface submodule and PD interface adjust has been written. It is composed analogous to the data as defined in 7.3.3.2.5.6.

PD Port MRPIC Data Check

This conditional parameter is present only if the submodule is an interface submodule and PD port MRPIC data check has been written. It is composed analogous to the data as defined in 7.3.3.5.5.10.

PD Port MRPIC Data Adjust

This conditional parameter is present only if the submodule is an interface submodule and PD port MRPIC data adjust has been written. It is composed analogous to the data as defined in 7.3.3.5.5.12.

7.3.3.2.5.4 Read PD Interface Data Real

This service allows reading PD Interface Data Real using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 232 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 232 – Data elements of Read PD Interface Data Real

Parameter name	service specific elements	rsp	cnf
Data	Chassis ID	M	M(=)
	MAC Address	M	M(=)
	IP Parameter Value	M	M(=)
	IP Address	M	M(=)
	Subnet Mask	M	M(=)
	Standard Gateway	M	M(=)

Data

This parameter contains the values of the object which has been read.

Chassis ID

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

MAC Address

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

IP Parameter Value

This parameter consists of the following elements:

IP Address

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

Subnet Mask

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

Standard Gateway

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

7.3.3.2.5.5 Write PD Interface Adjust

This service allows writing PD Interface Adjust Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1). PD Interface Data Adjust belongs to a submodule, which is addressed by the default API (zero), its slot and subslot number. The parameter Index identifies the data record index as specified in IEC 61158-6-10. If the addressed submodule does not represent an interface an error is returned.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 233 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 233 – Data elements of Write PD Interface Adjust

Parameter name	service specific elements	req	ind
Data	Multiple Interface Mode	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

Multiple Interface Mode

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

7.3.3.2.5.6 Read PD Interface Adjust

This service allows reading PD Interface Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 233 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Interface Adjust in 7.3.3.2.5.5.

7.3.3.2.5.7 Write PD Interface FSU Data Adjust

This service allows writing PD Interface FSU Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 234 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 234 – Data elements of Write PD Interface FSU Data Adjust

Parameter name	service specific elements	req	ind
Data	FS Hello Block	U	U(=)
	FS Hello Mode	M	M(=)
	FS Hello Interval	M	M(=)
	FS Hello Retry	M	M(=)
	FS Hello Delay	M	M(=)
	Fast Startup Block	U	U(=)
	Data	M	M(=)

Data

This parameter contains the value of the object which may be used to write.

FS Hello Block

This parameter consists of the following elements:

FS Hello Mode

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

FS Hello Interval

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

FS Hello Retry

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

FS Hello Delay

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

Fast Startup Block

This optional parameter consists of the following elements:

Data

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

7.3.3.2.5.8 Read PD Interface FSU Data Adjust

This service allows reading PD Interface FSU Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 234 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Interface FSU Data Adjust in 7.3.3.2.5.7.

7.3.3.2.5.9 Write PD NC Data Check

This service allows writing PD NC Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 235 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 235 – Data elements of Write PD NC Data Check

Parameter name	service specific elements	req	ind
Data	Maintenance Required Drop Budget	M	M(=)
	Maintenance Demanded Drop Budget	M	M(=)
	Error Drop Budget	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

Maintenance Required Drop Budget

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

Maintenance Demanded Drop Budget

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

Error Drop Budget

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

7.3.3.2.5.10 Read PD NC Data Check

This service allows reading PD NC Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 235 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD NC Data Check in 7.3.3.2.5.9.

7.3.3.2.5.11 Read PD Port Data Real

This service allows reading PD Port Data Real using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 236 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 236 – Data elements of Read PD Port Data Real

Parameter name	service specific elements	rsp	cnf
Data	Own Port Name	M	M(=)
	Number Of Peers	C	C(=)
	List of Peers	C	C(=)
	Peer Port Name	M	M(=)
	Peer Station Name	M	M(=)
	Line Delay	M	M(=)
	Peer MAC Address	M	M(=)
	MAU Type	M	M(=)
	RTClass3_PortStatus	M	M(=)
	Multicast Boundary	M	M(=)
	Link State	M	M(=)
	Media Type	M	M(=)

Data

This parameter contains the values of the object which has been read.

Own Port Name

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

Number Of Peers

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

List of Peers

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

Peer Port Name

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

Peer Station Name

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

Line Delay

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

Peer MAC Address

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.

RTClass3_PortStatus

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

Multicast Boundary

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

Link State

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

Media Type

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

7.3.3.2.5.12 Read PD Port Data Real Extended

This service allows reading PD Port Data Real Extended using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 237 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 237 – Data elements of Read PD Port Data Real Extended

Parameter name	service specific elements	rsp	cnf
Data	Own Port Name	M	M(=)
	Line Delay	M	M(=)
	Media Type	M	M(=)
	Multicast Boundary	M	M(=)
	MAU Type	M	M(=)
	MAU Type Extension	M	M(=)
	Link State	M	M(=)
	RTClass3_PortStatus	M	M(=)
	Number Of Peers	C	C(=)
	List of Peers	C	C(=)
	Line Delay	M	M(=)
	MAU Type	M	M(=)
	MAU Type Extension	M	M(=)
	Peer MAC Address	M	M(=)
	Peer Port Name	M	M(=)
	Peer Station Name	M	M(=)

Data

This parameter contains the values of the object which has been read.

Own Port Name

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

Line Delay

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

Media Type

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

Multicast Boundary

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.

MAU Type Extension

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

Link State

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

RTClass3_PortStatus

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

Number Of Peers

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

List of Peers

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

Line Delay

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.

MAU Type Extension

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

Peer MAC Address

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

Peer Port Name

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

Peer Station Name

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

7.3.3.2.5.13 Write PD Port Data Check

This service allows writing PD Port Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 238 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 238 – Data elements of Write PD Port Data Check

Parameter name	service specific elements	req	ind
Data	List of Peers	U	U(=)
	Peer Port ID	M	M(=)
	Peer Chassis ID	M	M(=)
	Line Delay	U	U(=)
	MAU type	U	U(=)
	Link State	U	U(=)
	Sync Difference	U	U(=)
	MAU Type Difference	U	U(=)

Data

This parameter contains the values of the object which may be used to write.

List of Peers

This optional 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.

Line Delay

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

MAU type

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

Link State

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

Sync Difference

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

MAU Type Difference

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

7.3.3.2.5.14 Read PD Port Data Check

This service allows reading PD Port Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 238 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

7.3.3.2.5.15 Write PD Port Data Adjust

This service allows writing PD Port Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 239 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 239 – Data elements of Write PD Port Data Adjust

Parameter name	service specific elements	req	ind
Data	Adjust Domain Boundary	U	U(=)
	Adjust Multicast Boundary	U	U(=)
	Adjust MAU Type	U	U(=)
	Adjust Link State	U	U(=)
	Adjust Peer to Peer Boundary	U	U(=)
	Adjust DCP Boundary	U	U(=)
	Adjust Preamble Length	U	U(=)
	Adjust MAU Type Extension	U	U(=)

Data

This parameter contains the values of the object which may be used to write.

Adjust Domain Boundary

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

Adjust Multicast Boundary

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

Adjust MAU Type

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

Adjust Link State

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

Adjust Peer to Peer Boundary

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

Adjust DCP Boundary

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

Adjust Preamble Length

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

Adjust MAU Type Extension

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

7.3.3.2.5.16 Read PD Port Data Adjust

This service allows writing PD Port Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 239 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Port Data Adjust in 7.3.3.2.5.15.

7.3.3.2.5.17 Read PD Port FO Data Real

This service allows reading PD Port FO Data Real using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 240 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 240 – Data elements of Read Port FO Data Real

Parameter name	service specific elements	rsp	cnf
Data	Fiber Optic type	M	M(=)
	Fiber Optic Cable type	M	M(=)
	Fiber Optic Manufacturer Specific	C	C(=)
	Vendor ID	M	M(=)
	Manufacturer Specific Fiber Optic Data	M	M(=)
	Fiber Optic Power Budget Real	M	M(=)

Data

This parameter contains the values of the object which has been read.

Fiber Optic type

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

Fiber Optic Cable type

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

Fiber Optic Manufacturer Specific

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

Vendor ID

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

Manufacturer Specific Fiber Optic Data

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

Fiber Optic Power Budget Real

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

7.3.3.2.5.18 Write PD Port FO Data Check

This service allows writing PD Port FO Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 241 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 241 – Data elements of Write PD Port FO Data Check

Parameter name	service specific elements	req	ind
Data	Maintenance Required Power Budget	M	M(=)
	Maintenance Demanded Power Budget	M	M(=)
	Error Power Budget	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

Maintenance Required Power Budget

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

Maintenance Demanded Power Budget

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

Error Power Budget

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

7.3.3.2.5.19 Read PD Port FO Data Check

This service allows reading PD Port FO Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 241 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Port FO Data Check in 7.3.3.2.5.18.

7.3.3.2.5.20 Write PD Port FO Data Adjust

This service allows writing PD Port FO Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 242 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 242 – Data elements of Write PD Port FO Data Adjust

Parameter name	service specific elements	req	ind
Data	Adjust Fiber Optic type	M	M(=)
	Adjust Fiber Optic Cable type	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

Adjust Fiber Optic type

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

Adjust Fiber Optic Cable type

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

7.3.3.2.5.21 Read PD Port FO Data Adjust

This service allows reading PD Port FO Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 242 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Port FO Data Adjust in 7.3.3.2.5.20.

7.3.3.2.5.22 Read PD Port Statistic

This service allows writing PD Port Statistic using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface and port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 243 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 243 – Data elements of Read PD Port Statistic

Parameter name	service specific elements	rsp	cnf
Data	CounterStatus	C	C(=)
	ifInOctets	M	M(=)
	ifOutOctets	M	M(=)
	ifInDiscards	M	M(=)
	ifOutDiscards	M	M(=)
	ifInErrors	M	M(=)
	ifOutErrors	M	M(=)

Data

This parameter contains the values of the object which has been read.

CounterStatus

This parameter indicates whether the value of each of the following attributes is valid or invalid. Bitwise coding is used to address the validity of all following attributes.

ifInOctets

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

ifOutOctets

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

ifInDiscards

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

ifOutDiscards

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

ifInErrors

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

ifOutErrors

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

7.3.3.2.6 Behavior**7.3.3.2.6.1 General**

The startup procedure for an IO controller provides the following steps:

- start MAC interface
- start LLDP
- start DCP receiver, DCP sender, and DHCP client (if DHCP enabled)
- use local configured name of station or wait for it
- use local configured IP address or wait for it
- 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 “normal” startup procedure for an IO device provides the following steps:

- start MAC interface
- start LLDP
- start DCP receiver and DHCP client (if DHCP enabled)
- use name of station (if persistent) or wait for it
- wait for connection establishment

NOTE 1 The IP address assignment to the IO device is typically part of the connection establishment.

The “fast” startup procedure for an IO device provides the following steps:

- start MAC interface (auto-negotiation off)
- start LLDP
- start DCP receiver and sender
- use name of station (if persistent) or wait for it
- send DCP hello if configured
- wait for connection establishment

NOTE 2 The IP address of the IO device will be typically assigned persistent; thus IP address assignment is typically not part of the connection establishment.

The IO controller provides the following steps to fast startup a related IO device:

- Wait for DCP hello indication, if configured
- CM Connect using stored IP address

The detailed procedure for startup is provided in the protocol specification.

7.3.3.2.6.2 Persistency

Persistency is used to store data over power off.

To this data belongs

- Data set by using DCP:Set where BlockQualifier is set to SAVE_PERMANENT; these are IP Suite, Name Of Station, DHCP Parameters, or manufacturer specific data.

- Data records, where column Persistence in the tables stated in IEC 61158-6-10, Subclause Record index shows “Yes”, and written during a parameterization sequence up to Prm End (initiated with Connect.req, Prm Begin.req, or Plug.req).
- Data records, where column Persistence in the tables stated in IEC 61158-6-10, Subclause Record index shows “User specific definition”, and written within or outside a parametrization sequence, and the vendor of the device decided to store the data record in the persistency.
- Data written using SNMP: Set, where the data shall be stored persistent; for this specification these are at least sysContact, sysName, and sysLocation.

This data is stored in the persistency and automatically retrieved after power on. In case of ResetToFactory, the persistent parameters are deleted as defined for the suboption; see 6.3.7.3.2 for details.

7.3.3.2.6.3 CIM parameters

For data records belonging to the CIM ASE (abbreviated in the following as CIM parameters) additional rules apply.

With Prm End the parameters are checked for consistency (as usual).

If the CIM parameters are consistent, they will be stored in the persistency and applied to interface and ports. If the parameters stored already in the persistency are however identical to the ones to be applied, neither persistency nor parameterization of interface and ports shall be changed to avoid unnecessary disturbance of the network.

If the CIM parameters are inconsistent, they are replaced by the default parameterization of interface and ports. The default parameters are stored in the persistency and applied to interface and ports. With this it is ensured that the device is still reachable on the network and is able to communicate.

In detail:

1) AR connect at PrmEnd

If an AR is connected and contains consistent CIM parameters:

- If the CIM parameters are already applied (i.e. the CIM parameters delivered in the AR are the same as the CIMN parameters in the persistency), CIM parameterization shall be omitted to avoid unnecessary disturbance of interface and ports.
- Otherwise the CIM parameters are applied and saved to the persistency. They replace the previous CIM parameters completely.

If an AR is connected and contains inconsistent CIM parameters:

- The CIM parameters are deleted from the persistency. Default CIM parameters are applied.

If an AR is connected and no CIM parameters:

- The current CIM parameters remain applied (i.e. no re-parameterization to default).

2) AR release / abort

The current CIM parameters remain applied (i.e. no re-parameterization to default).

3) Power on

The CIM parameters are retrieved from the persistency.

If the persistency contains consistent CIM parameters, the CIM parameters are applied.

Otherwise the CIM parameters are deleted from the persistency. Default CIM parameters are applied.

7.3.3.3 PTCP class

7.3.3.3.1 Overview

"Precision Transparent Clock Protocol" (PTCP) is used for WORKING_CLOCK synchronization.

Two device roles are specified: Sync master and sync slave. The WORKING_CLOCKs of all sync slaves are synchronized onto the WORKING_CLOCK of the sync master. The IO controller does not have to be the sync master. PTCP supports redundancy.

7.3.3.3.2 Relations

The services are bound to other services as shown in Table 244:

Table 244 – Binding PTCP services

Service	Binding	Usage
Write PD Sync Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Sync Data	AR:Record Data:Read	used by IO controller or IO supervisor
Local Sync State Info	Local	used by FAL
Local SYNCH Event	Local	used by FAL
Local	AR:Alarm:Alarm Notification	used by PTCP class

7.3.3.3.3 Template

A PTCP object is described by the following template:

ASE: Communication Interface Management ASE

CLASS: PTCP

CLASS ID: not used

PARENT CLASS: PTCP

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: List of Interfaces
- 2.1 (m) Attribute: API
- 2.2 (m) Attribute: Slot Number
- 2.3 (m) Attribute: Subslot Number
- 2.4 (m) Attribute: PTCP Subdomain UUID
- 2.5 (m) Attribute: IR Data UUID
- 2.6 (m) Attribute: Reserved Interval Begin
- 2.7 (m) Attribute: Reserved Interval End
- 2.8 (m) Attribute: PLL Window
- 2.9 (m) Attribute: Sync Send Factor
- 2.10 (m) Attribute: Send Clock Factor
- 2.11 (m) Attribute: PTCP Timeout Factor
- 2.12 (m) Attribute: PTCP Takeover Timeout Factor
- 2.13 (m) Attribute: PTCP Master Startup Time
- 2.14 (m) Attribute: Sync Properties
 - 2.14.1 (m) Attribute: Role
 - 2.14.2 (m) Attribute: Sync Class
- 2.15 (m) Attribute: PTCP Master Priority 1

2.15.1	(m) Attribute:	Priority
2.15.1	(m) Attribute:	Level
2.16	(m) Attribute:	PTCP Master Priority 2
2.16	(m) Attribute:	PTCP Subdomain Name

SERVICES:

1	(o) OpsService:	Write PD Sync Data
2	(o) OpsService:	Read PD Sync Data
3	(o) OpsService:	Local Sync State Info
4	(m) OpsService:	Local SYNCH Event

7.3.3.3.4 Attributes

Implicit

The key attribute Implicit indicates that the Communication Interface Management object is implicitly addressed by the service.

List of Interfaces

This attribute list lists all interfaces related to the PTCP object.

An entry consists of the following attributes:

API

This attribute contains the API containing the interface submodule.

The interface submodule is always assigned to the default API.

Attribute type: Unsigned32

Allowed Values: 0x00000000

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 227.

PTCP Subdomain UUID

This attribute contains the unique identifier of the PTCP subdomain which is configured by the engineering system. A PTCP subdomain is a set of IO controller and/or IO devices with synchronized send clocks.

Attribute type: UUID

IR Data UUID

This attribute contains the unique identifier of the isochronous realtime project which is configured by the engineering system.

Attribute type: UUID

Reserved Interval Begin

This attribute contains the start of RT_CLASS_2 reserved bandwidth relatively to the current phase in nanoseconds.

Attribute type: Unsigned32

Allowed Values: 0 (meaning no reserved bandwidth for RT_CLASS_2 is defined)

RT_CLASS_2 is usable in the context of legacy startup only. Only in that case, the value may be set to a value different from 0.

Reserved Interval End

This attribute contains the end of RT_CLASS_2 reserved bandwidth relatively to the current phase in nanoseconds.

Attribute type: Unsigned32

Allowed Values: 0 (meaning no reserved bandwidth for RT_CLASS_2 is defined)

RT_CLASS_2 is usable in the context of legacy startup only. Only in that case, the value may be set to a value different from 0.

PLL Window

This attribute contains the maximum jitter of the start of the isochronous cycle with a time base of 1 ns. The value 0 indicates no PLL Window is defined. The attribute has no relevance for the Sync Master.

If the actual value of the local clock of the Sync Slave is inside $\pm$ PLL Window of the desired value, the clock is adjusted. Otherwise the sync aborts and restarts.

Attribute type: Unsigned32

Time Base: 1 ns

Sync Send Factor

This attribute contains the send interval of the sync PDUs. The Sync Send Factor shall be set in multiples of 31,25 μ s.

Attribute type: Unsigned16

The values shall be in the range of 1 to 2 764 800 000.

Send Clock Factor

This attribute is defined in 7.3.1.3.4.

PTCP Timeout Factor

This attribute contains the Timeout Factor for detecting the loss of sync messages. The Time Base is the Sync Send Factor.

Attribute type: Unsigned16

Allowed values: 0x0000 to 0x00FF

Default value: 0x0006

PTCP Takeover Timeout Factor

This attribute contains a timeout value for detection the loss of sync messages from the current sync master. The sync slave shall try to find a new sync master after the expiration of this time. The PTCP takeover timeout value shall be in the range from 32 ms to 16 352 ms for Sync ID = CLOCK and in the range from 32 ms to 2 759 400 000 ms for Sync ID = TIME.

Attribute type: Unsigned32

Time Base: Sync Send Factor * 31,25 μ s

Allowed values: 1 to 511

Default value: 0x00000003 for Sync Master, 0x00000002 for Sync Slave

PTCP Master Startup Time

This attribute shall be the duration of the startup phase of the sync master. During this time the sync master shall check if there is a sync master with a higher clock quality in the sync domain.

Attribute type: Unsigned16

Time Base: 1 s

Allowed values: 0 to 300 s

Default value: 60

Sync Properties

This attribute contains the following attributes:

Role

This attribute contains the role of the device related to time synchronization.

Allowed values shall be according to Table 245.

Table 245 – Sync Properties Role

Meaning
Local sync
External sync: Role Sync Slave
Sync Master

Sync Class

This attribute contains the value for the Sync Class. Allowed values shall be according to Table 246.

Table 246 – 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

PTCP Master Priority 1

This attribute is used in the execution of the best master clock algorithm. Lower values take precedence.

This attribute contains the following attributes:

Priority

This attribute contains the priority.

Attribute type: Unsigned8

Allowed Values: 0x00 (if role is sync slave), 0x01-0x02 (if role is sync master)

Level

This attribute contains the level.

Attribute type: Unsigned8

Allowed Values: 0x00 (highest) - 0x07 (lowest)

PTCP Master Priority 2

This attribute is used in the execution of the best master clock algorithm. Lower values take precedence.

Attribute type: Unsigned8

Allowed Values: 0xFF

PTCP Subdomain Name

This attribute contains the name of the PTCP subdomain which is configured by the engineering system. A PTCP subdomain is a set of IO controller and/or IO devices with synchronized send clocks.

Attribute type: see Chassis ID

7.3.3.3.5 Services

7.3.3.3.5.1 Write PD Sync Data

This service allows writing PD Sync Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 247 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 247 – Data elements of Write PD Sync Data

Parameter name	service specific elements	req	ind
Data	PTCP Subdomain 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(=)
	PTCP Timeout Factor	M	M(=)
	PTCP Takeover Timeout Factor	M	M(=)
	PTCP Master Startup Time	M	M(=)
	Sync Properties	M	M(=)
	PTCP Master Priority 1	M	M(=)
	PTCP Master Priority 2	M	M(=)
	PTCP Subdomain Name	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

PTCP Subdomain ID

This parameter contains the value of the corresponding attribute of the PTCP class.

Reserved Interval Begin

This parameter contains the value of the corresponding attribute of the PTCP class.

Reserved Interval End

This parameter contains the value of the corresponding attribute of the PTCP class.

PLL Window

This parameter contains the value of the corresponding attribute of the PTCP class.

Sync Send Factor

This parameter contains the value of the corresponding attribute of the PTCP class.

Send Clock Factor

This parameter contains the value of the corresponding attribute of the PTCP class.

PTCP Timeout Factor

This parameter contains the value of the corresponding attribute of the PTCP class.

PTCP Takeover Timeout Factor

This parameter contains the value of the corresponding attribute of the PTCP class.

PTCP Master Startup Time

This parameter contains the value of the corresponding attribute of the PTCP class.

Sync Properties

This parameter contains the value of the corresponding attribute of the PTCP class.

PTCP Master Priority 1

This parameter contains the value of the corresponding attribute of the PTCP class.

PTCP Master Priority 2

This parameter contains the value of the corresponding attribute of the PTCP class.

PTCP Subdomain Name

This parameter contains the value of the corresponding attribute of the PTCP class.

7.3.3.3.5.2 Read PD Sync Data

This service allows reading PD Sync Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 247 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Sync Data (7.3.3.3.5.1).

7.3.3.3.5.3 Local Sync State Info

This local service indication shall be used to indicate any change of the synchronization state of an interface. Table 248 shows the parameters of the service.

Table 248 – Local Sync State Info

Parameter name	ind
Argument	M
Sync Data	S
Slot Number	M
Subslot Number	M
PTCP Subdomain ID	M
IR Data UUID	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

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.

PTCP Subdomain ID

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

IR Data UUID

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.

PTCP Timeout Factor

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 of the corresponding attribute of the ASE object.

7.3.3.3.5.4 Local SYNCH Event

The service SYNCH indicates to the application of the IO device that a new isochronous send clock cycle has been started.

Table 249 shows the parameters of the service.

Table 249 – Local SYNCH Event

Parameter name	ind
Argument	M
Slot	M
Subslot	M
Global Cycle Counter	M
Phase	M
Status	M

Argument

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

Slot Number

This parameter contains the Slot Number addressing the initiators interface.

Subslot Number

This parameter contains the Subslot Number addressing the initiators interface.

Global Cycle Counter

This parameter shall contain the local value of the cycle counter.

Phase

This parameter contains the Phase to be used for the RTC frame.

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.

7.3.3.3.6 Behavior

7.3.3.3.6.1 Diagnosis and Alarms

PTCP synchronization may cause diagnosis events. There are some dependencies of sync errors, so not all errors occur at once.

If no synchronization mismatch is present we are synchronized. “Wrong PTCPSubDomainID” and “Wrong IRDataID” requires a sync message to be received. “Jitter out of Boundary” requires a sync message to be received and correct PTCPSubDomainID and IRDataID.

These dependencies are handled within PTCP “Local Sync State Info” service.

7.3.3.3.6.2 Primitive definitions

7.3.3.3.6.2.1 Primitives exchanged between remote machines

The service primitives including their associated parameters issued or received by synchronization diagnosis (SYNC_DIAG) devices are described in the service definition and shown in Table 250.

Table 250 – Remote primitives issued or received by SYNC_DIAG

Primitive	Source	Destination	Associated parameters	Functions
Local Diagnosis Event.req	SYNC_DIAG	FAL	see 7.3.4.2.5.5	—
Alarm.req	SYNC_DIAG	FAL	see 7.3.1.5.5	—
Alarm.cnf ()	FAL	SYNC_DIAG	see 7.3.1.5.5	—

7.3.3.3.6.2.2 Primitives exchanged between local machines

The local service primitives including their associated parameters issued or received by SYNC_DIAG are described in the service definition and shown in Table 251.

Table 251 – Local primitives issued or received by SYNC_DIAG

Primitive	Source	Destination	Associated parameters	Functions
Sync State Change.ind	local	SYNC_DIAG	see 6.3.11.3.7	—

7.3.3.3.6.3 State transition diagram

The state transition diagram of the SYNC_DIAG is shown in Figure 45

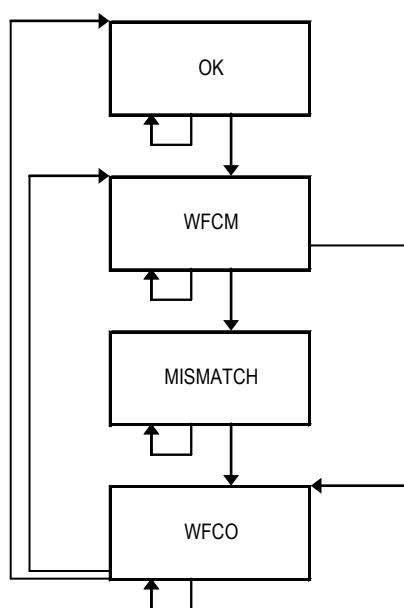


Figure 45 – State transition diagram SYNC_DIAG

States of the SYNC_DIAG

OK	Initial state and synchronization ok
WFCM	Wait for alarm confirm while going to mismatch
MISMATCH	synchronization diagnosis exists
WFCO	Wait for alarm confirm while going to ok

7.3.3.3.6.4 State machine description

This state machine cares about the diagnosis issues regarding synchronization. It is triggered by sync state changed indications and issues coming and going alarms.

7.3.3.3.6.5 State table

Table 252 defines the state table for SYNC_DIAG.

Table 252 – State table SYNC_DIAG

#	Current State	Event /Condition =>Action	Next State
1	OK	Sync State Change.ind () /SyncState == SYNCHRONIZED => ignore	OK
2	OK	Sync State Change.ind () /SyncState == NOT_SYNCHRONIZED => SYNC_MISMATCH (SyncID, ErrorType) NextState := MISMATCH Alarm.req ()	WFCM
3	OK	Alarm.cnf () => ignore	OK
4	WFCM	Sync State Change.ind () /SyncState == SYNCHRONIZED => NextState := OK	WFCM
5	WFCM	Sync State Change.ind () /SyncState == NOT_SYNCHRONIZED => NextState := MISMATCH	WFCM
6	WFCM	Alarm.cnf () /NextState == OK => SYNC_OK (SyncID) Alarm.req ()	WFCO
7	WFCM	Alarm.cnf () /NextState == MISMATCH => ignore	MISMATCH
8	MISMATCH	Alarm.cnf () => ignore	MISMATCH
9	MISMATCH	Sync State Change.ind () /SyncState == SYNCHRONIZED => SYNC_OK (SyncID) NextState := OK Alarm.req ()	WFCO
10	MISMATCH	Sync State Change.ind () /SyncState == NOT_SYNCHRONIZED => ignore	MISMATCH
11	WFCO	Sync State Change.ind () /SyncState == SYNCHRONIZED => NextState := OK	WFCO
12	WFCO	Sync State Change.ind () /SyncState == NOT_SYNCHRONIZED => NextState := MISMATCH	WFCO
13	WFCO	Alarm.cnf () /NextState == MISMATCH => SYNC_MISMATCH (SyncID, ErrorType) Alarm.req ()	WFCM

#	Current State	Event /Condition =>Action	Next State
14	WFCO	Alarm.cnf () /NextState == OK => ignore	OK

7.3.3.3.6.6 Functions, Macros, Timers and Variables

Table 253 contains the functions, macros, timers and variables used by the state table of SYNC_DIAG, their arguments and descriptions.

Table 253 – Functions, Macros, Timers and Variables used by SYNC_DIAG

Name	Type	Function/Meaning
SYNC_OK(SyncID)	Function	No Sync mismatch. Sync OK. Alarm Priority:= ALARM_LOW if (SyncID == 0) Alarm type:= Sync Data Change Notification else Alarm type:= Time Data Change Notification Local Remove Diagnosis Entry () if (SyncID == 0) Channel Error type := Sync mismatch else Channel Error type := Time mismatch Local Diagnosis Event_ind ()
SYNC_MISMATCH(SyncID,ErrorType)	Function	Sync mismatch. Issue an Alarm Notification and Diagnosis Event. Only changes of error states will cause an Alarm Notification and Diagnosis Event! Alarm Priority:= ALARM_LOW if (SyncID == 0) Alarm type:= Sync Data Change Notification else Alarm type:= Time Data Change Notification Local Add Diagnosis Entry () if (SyncID == 0) Channel Error type := Sync mismatch else Channel Error type := Time mismatch Ext Channel Error type := ErrorType Local Diagnosis Event_ind ()
NextState	Variable	This local variable contains the desired state

7.3.3.4 MAC Bridges class

7.3.3.4.1 Overview

The MAC Bridges ASE controls the bridging according to IEEE 802.1Q (MAC bridges) extended with the special requirements for RT_CLASS_3 and additionally uses the Virtual Bridges ASE for each port.

7.3.3.4.2 Relations

The services are bound to other services as shown in Table 254:

Table 254 – Binding MAC Bridge services

Service	Binding	Usage
Write PD IR Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD IR Data	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD IR Subframe Data	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD IR Subframe Data	AR:Record Data:Read	used by IO controller or IO supervisor

7.3.3.4.3 Template

A Communication Interface Management object is described by the following template:

ASE: Communication Interface Management ASE

CLASS: MAC Bridge

CLASS ID: not used

PARENT CLASS: MAC Bridge

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: List of Interfaces
- 2.1 (m) Attribute: API
- 2.2 (m) Attribute: Slot Number
- 2.3 (m) Attribute: Subslot Number
- 2.4 (o) Attribute: PD IR Data
 - 2.4.1 (m) Attribute: PD IR Global Data
 - 2.4.2 (m) Attribute: PD IR Frame Data
 - 2.4.3 (m) Attribute: PD IR Begin End Data
- 2.5 (m) Attribute: PD IR Subframe Data
 - 2.5.1 (m) Attribute: List of Subframes
 - 2.5.1.1 (m) Attribute: Frame ID
 - 2.5.1.2 (m) Attribute: SF IOCR Properties
 - 2.5.1.2.1 (m) Attribute: Distributed Watchdog Factor
 - 2.5.1.2.2 (m) Attribute: Restart Factor for Distributed Watchdog
 - 2.5.1.2.3 (m) Attribute: DFP Mode
 - 2.5.1.2.4 (m) Attribute: DFP Redundant Path Layout
 - 2.5.1.2.5 (m) Attribute: SFCRC16
 - 2.5.1.2.6 (m) Attribute: DFP Direction

SERVICES:

- 7 (o) OpsService: Write PD IR Data
- 8 (o) OpsService: Read PD IR Data
- 12 (o) OpsService: Write PD IR Subframe Data
- 13 (o) OpsService: Read PD IR Subframe Data

7.3.3.4.4 Attributes

Implicit

The key attribute Implicit indicates that the Communication Interface Management object is implicitly addressed by the service.

List of Interfaces

This attribute list lists all interfaces related to the Communication Interface Management object. It contains the following attributes:

API

This attribute contains the API containing the interface submodule.

The interface submodule is always assigned to the default API.

Attribute type: Unsigned32

Allowed Values: 0x00000000

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 227.

PD IR Data

This optional attribute contains the following attributes:

PD IR Global Data

This attribute contains the global IR data used by the engineering system as computation base. This attribute is inherited from parent class MAC Bridge (see 6.3.14.2).

PD IR Frame Data

This conditional list of elements shall be filled by the engineering system, if the device support IRT with IR Frame Data Elements only, as indicated by the GSD. This attribute is inherited from parent class MAC Bridge (see 6.3.14.2).

PD IR Begin End Data

This list contains the Red Guard and the bandwidth administration information for the traffic classes RED and GREEN. This attribute is inherited from parent class MAC Bridge (see 6.3.14.2).

PD IR Subframe Data

This optional attribute lists the DFP information. It consists of the following attributes:

List of Subframes

This attribute lists all DFP subframes for which the IO device is involved. A list element includes the following attributes:

Frame ID

This parameter contains the frame ID of the related DFP IO CR.

Attribute type: Unsigned16

Allowed values: according to the Frame ID range usable for DFP as defined in IEC 61158-6-10.

SF IOCR Properties

This attribute describes the properties for the DFP frame. It contains the following attributes:

Distributed Watchdog Factor

This attribute contains the distributed watchdog factor. For details to the distributed watchdog see Annex E.

The attribute shall be set to “0” only if the IO controller is able to work on subframe level.

Table 255 – Distributed Watchdog Factor

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	Distributed WatchDog shall be disabled
0x01 – 0x02	Optional	The value “n” leads to an AR termination for n missed frames. An expiration shall be signaled to the application as a hint for a possible communication disturbance.
0x03 – 0x1F	Mandatory	
0x20 – 0xFF	Optional	

Time base: $\text{SendClockFactor} * \text{ReductionRatio} * 31,25 \mu\text{s}$

Attribute type: Unsigned8

Allowed values: see Table 255

Restart Factor for Distributed Watchdog

This attribute describes the restart factor for the distributed watchdog.

The attribute is set to “0” only if the IO controller is able to work on subframe level.

Table 256 – Restart Factor For Distributed Watchdog

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	No restart delay necessary
0x01 – 0x09	Optional	Less than 1 s restart delay
0x0A – 0x50	Mandatory	1 s to 8 s restart delay
0x51 – 0xFF	Optional	More than 8 s restart delay

The `RestartFactorForDistributedWD` shall only be used if the `SFIOCRProperties.DistributedWatchDogFactor` is not set to zero.

NOTE The `RestartForDistributedWDTime` is defined as minimum time having an accuracy between ‘-0’ and ‘+1 s’.

Time base: 100 ms

Attribute type: Unsigned8

Allowed values: see Table 256

DFP Mode

This attribute describes the DFP mode of the frame.

Table 257 – DFP Mode

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	End node mode
0x01 – 0x3F	Mandatory	Concatenating mode Own Subframe position
0x40 – 0xFF	Optional	Concatenating mode Own Subframe position

Attribute type: Unsigned8

Allowed values: see Table 257

DFP Redundant Path Layout

This attributed describes the layout of the redundant DFP frame.

Table 258 – SFIOCRProperties.DFPRedundantPathLayout

Value (hexadecimal)	Meaning	Description
0x00	Mandatory	The Frame for the redundant path, coded by the FrameID with the least significant bit set to one, shall contain the ordering shown by SubframeData.
0x01	Mandatory	The Frame for the redundant path, coded by the FrameID with the least significant bit set to one, shall contain the inverse ordering of the SubframeData.

Attribute type: Boolean

Allowed values: see Table 258

SFCRC16

This attribute indicates the validity of the subframe checksum and the subframe's cycle counter.

Table 259 – SFCRC16

Value (hexadecimal)	Description
0x00	SFCRC16 and SFCycleCounter shall be created or set to zero by the sender and not checked by the receiver.
0x01	SFCRC16 and SFCycleCounter shall be created by the sender and checked by the receiver.

Attribute type: Boolean

Allowed values: see Table 259

DFP Direction

This attribute indicates the direction of the conveyance a DFP frame.

Attribute type: Boolean

Allowed values: DFP_INBOUND, DFP_OUTBOUND

7.3.3.4.5 Services

7.3.3.4.5.1 Write PD IR Data

This service allows writing PD IR Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 260 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 260 – Data elements of Write PD IR Data

Parameter name	service specific elements	req	ind
Data	Slot Number	M	M(=)
	Subslot Number	M	M(=)
	PD IR Global Data	M	M(=)
	IR Data UUID	M	M(=)
	Max Bridge Delay	M	M(=)
	Number of Ports	M	M(=)
	List of Port Delays	M	M(=)
	Max Port Tx Delay	M	M(=)
	Max Port Rx Delay	M	M(=)
	Max Line Rx Delay	M	M(=)
	Yellow Time	M	M(=)
	PD IR Frame Data	M	M(=)
	Frame Data Properties	M	M(=)
	List of PD 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(=)
	PD IR Begin End Data	M	M(=)
	Red Guard	M	M(=)
	Start of Red Frame ID	M	M(=)
	End of Red Frame ID	M	M(=)
	Number of Ports	M	M(=)
	List of Ports	M	M(=)
	Number of Assignments	M	M(=)
	List of Assignments	M	M(=)
	Tx Begin End Assignment	M	M(=)

Parameter name	service specific elements	req	ind
	Red Period Begin	M	M(=)
	Green Period Begin	M	M(=)
	Rx Begin End Assignment	M	M(=)
	Red Period Begin	M	M(=)
	Green Period Begin	M	M(=)
	Number of Phases	M	M(=)
	List of Phases	M	M(=)
	Tx Phase Assignment	M	M(=)
	Assigned Value For Reserved Begin	M	M(=)
	Assigned Value For Reserved End	M	M(=)
	Rx Phase Assignment	M	M(=)
	Assigned Value For Reserved Begin	M	M(=)
	Assigned Value For Reserved End	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

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.

PD IR Global Data

The parameter contains the attributes as in Table 260 (parameter description in 6.3.14.2.3).

PD IR Frame Data

The parameter contains the attributes as in Table 260 (parameter description in 6.3.14.2.3).

PD IR Begin End Data

The parameter contains the attributes as in Table 260 (parameter description in 6.3.14.2.3).

7.3.3.4.5.2 Read PD IR Data

This service allows reading PD IR Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 260 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD IR Data in 7.3.3.4.5.1.

7.3.3.4.5.3 Write PD IR Subframe Data

This service allows writing PD IR Subframe Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 261 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 261 – Data elements of Write PD IR Subframe Data

Parameter name	service specific elements	req	ind
Data	List of Subframes	M	M(=)
	Frame ID	M	M(=)
	SF IOCR Properties	M	M(=)
	Distributed Watchdog Factor	M	M(=)
	Restart Factor for Distributed Watchdog	M	M(=)
	DFP Mode	M	M(=)
	DFP Redundant Path Layout	M	M(=)
	SFCRC16	M	M(=)
	List of Subframe Data	M	M(=)
	Subframe Position	M	M(=)
	Subframe Length	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

List of Subframes

This parameter consists of the following elements:

Frame ID

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

SF IOCR Properties

This parameter consists of the following elements:

Distributed Watchdog Factor

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

Restart Factor for Distributed Watchdog

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

DFP Mode

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

DFP Redundant Path Layout

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

SFCRC16

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

List of Subframe Data

This parameter consists of the following elements:

Subframe Position

This parameter contains the value of the corresponding attribute of the Communication Relationship class (see 7.3.1.3.4).

Subframe Length

This parameter contains the value of the corresponding attribute of the Communication Relationship class (see 7.3.1.3.4).

7.3.3.4.5.4 Read PD IR Subframe Data

This service allows reading PD IR Subframe Data using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 261 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD IR Subframe Data in 7.3.3.4.5.3.

7.3.3.4.6 Behavior

The bridging mechanisms of IEEE 802.1Q shall be temporary disabled within the RED phase. In this case, the bridging of frames out of Frame ID range 3 (see IEC 61158-6-10) shall be done according to the attributes defined by this ASE with the behavior according to the RT_CLASS_3 Relay protocol machine.

During the green phase, normal forwarding applies.

7.3.3.5 Media Redundancy class

7.3.3.5.1 Overview

MRP is a recovery protocol based on a ring topology and the redundant interconnection of MRP rings. It is defined in IEC 62439-2 and used here for loop prevention and media redundancy.

MRP is designed to react deterministically on a single failure of an inter-switch link or switch in the ring or interconnection topology. Standard topology here is ring or ring with branches.

MRP interconnection protocol (MRPIC) is designed for redundant ring coupling of MRP rings.

7.3.3.5.2 Relations

The services are bound to other services as shown in Table 262:

Table 262 – Binding Media Redundancy services

Service	Binding	Usage
Read PD Interface MRP Data Real	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Interface MRP Data Check	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Interface MRP Data Check	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Interface MRP Data Adjust	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Interface MRP Data Adjust	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Port MRP Data Real	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Port MRP Data Adjust	AR:Record Data:Write	used by IO controller or IO supervisor

Service	Binding	Usage
Read PD Port MRP Data Adjust	AR:Record Data:Read	used by IO controller or IO supervisor
Read PD Port MRPIC Data Real	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Port MRPIC Data Check	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Port MRPIC Data Check	AR:Record Data:Read	used by IO controller or IO supervisor
Write PD Port MRPIC Data Adjust	AR:Record Data:Write	used by IO controller or IO supervisor
Read PD Port MRPIC Data Adjust	AR:Record Data:Read	used by IO controller or IO supervisor

7.3.3.5.3 Template

A MRP object is described by the following template:

ASE: Communication Interface Management ASE

CLASS: Media redundancy

CLASS ID: not used

PARENT CLASS: Media redundancy

ATTRIBUTES:

- 1 (m) Key Attribute: Implicit
- 2 (m) Attribute: List of Interfaces
- 2.1 (m) Attribute: API
- 2.1 (m) Attribute: Slot Number
- 2.2 (m) Attribute: Subslot Number
- 2.3 (m) Attribute: List of MRP Instances
- 2.4 (m) Attribute: List of MRPIC Ports

SERVICES:

- 1 (o) OpsService: Read PD Interface MRP Data Real
- 2 (o) OpsService: Write PD Interface MRP Data Check
- 3 (o) OpsService: Read PD Interface MRP Data Check
- 4 (o) OpsService: Write PD Interface MRP Data Adjust
- 5 (o) OpsService: Read PD Interface MRP Data Adjust
- 6 (o) OpsService: Read PD Port MRP Data Real
- 7 (o) OpsService: Write PD Port MRP Data Adjust
- 8 (o) OpsService: Read PD Port MRP Data Adjust
- 9 (o) OpsService: Read PD Port MRPIC Data Real
- 10 (o) OpsService: Write PD Port MRPIC Data Check
- 11 (o) OpsService: Read PD Port MRPIC Data Check
- 12 (o) OpsService: Write PD Port MRPIC Data Adjust
- 13 (o) OpsService: Read PD Port MRPIC Data Adjust

7.3.3.5.4 Attributes

Implicit

The key attribute Implicit indicates that the Communication Interface Management object is implicitly addressed by the service.

List of Interfaces

This attribute list lists all interfaces related to the Communication Interface Management object. It contains the following attributes:

API

This attribute contains the API containing the interface submodule.

The interface submodule is always assigned to the default API.

Attribute type: Unsigned32

Allowed Values: 0x00000000

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 227.

List of MRP Instances

This attribute is defined in the Media redundancy ASE; see 6.3.10.2.3 for details.

List of MRPIC Ports

This attribute is defined in the Media redundancy ASE; see 6.3.10.2.3 for details.

7.3.3.5.5 Services**7.3.3.5.5.1 Read PD Interface MRP Data Real**

This service allows reading PD Interface MRP Data Real using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 263 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 263 – Data elements of Read PD Interface MRP Data Real

Parameter name	service specific elements	req	ind
Data	List of MRP Instances	M	M(=)
	MRP Instance	M	M(=)
	MRP Domain UUID	M	M(=)
	MRP Role	M	M(=)
	MRP Domain Name	M	M(=)
	MRP Version	M	M(=)
	MRP Prio	C	C(=)
	MRP TOPchgT	C	C(=)
	MRP TOPNRmax	C	C(=)
	MRP TSTshortT	C	C(=)
	MRP TSTdefaultT	C	C(=)
	MRP TSTNRmax	C	C(=)
	MRP Ring State Data	C	C(=)

Parameter name	service specific elements	req	ind
	MRP LNKdownT	C	C(=)
	MRP LNKupT	C	C(=)
	MRP LNKNRmax	C	C(=)

Data

This parameter contains the values of the object which has been read.

List of MRP Instances

This list contains the MRP instances of this node.

MRP Instance

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

MRP Domain UUID

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

MRP Role

This parameter contains the value of the attribute operational MRP role of the ASE object.

MRP Domain Name

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

MRP Version

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

MRP Prio

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP TOPchgT

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP TOPNRmax

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP TSTshortT

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP TSTdefaultT

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP TSTNRmax

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP Ring State Data

This conditional attribute is present when the value of the operational MRP role attribute is "MANAGER" or "MANAGER_AUTO". It contains the value of the corresponding attribute of the ASE object.

MRP LNKdownT

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the value of the corresponding attribute of the ASE object.

MRP LNKupT

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the value of the corresponding attribute of the ASE object.

MRP LNKNRmax

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the value of the corresponding attribute of the ASE object.

7.3.3.5.5.2 Write PD Interface MRP Data Check

This service allows writing PD Interface MRP Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 264 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 264 – Data elements of Write PD Interface MRP Data Check

Parameter name	service specific elements	req	ind
Data	List of MRP Instances	M	M(=)
	MRP Instance	M	M(=)
	MRP Domain UUID	M	M(=)
	MRP Check	M	M(=)

Data

This parameter contains the values of the object which may be used to write.

List of MRP Instances

This list contains the MRP instances of this node.

MRP Instance

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

MRP Domain UUID

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

MRP Check

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

7.3.3.5.5.3 Read PD Interface MRP Data Check

This service allows reading PD Interface MRP Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 264 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Interface MRP Data Check in 7.3.3.5.5.2.

7.3.3.5.5.4 Write PD Interface MRP Data Adjust

This service allows writing PD Interface MRP Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 265 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 265 – Data elements of Write PD Interface MRP Data Adjust

Parameter name	service specific elements	req	ind
Data	List of MRP Instances	M	M(=)
	MRP Instance	M	M(=)
	MRP Domain UUID	M	M(=)
	MRP Role	M	M(=)
	MRP Domain Name	M	M(=)
	MRP Prio	C	C(=)
	MRP TOPchgT	C	C(=)
	MRP TOPNRmax	C	C(=)
	MRP TSTshortT	C	C(=)
	MRP TSTdefaultT	C	C(=)
	MRP TSTNRmax	C	C(=)
	MRP LNKdownT	C	C(=)
	MRP LNKupT	C	C(=)
	MRP LNKNRmax	C	C(=)

Data

This parameter contains the values of the object which may be used to write.

List of MRP Instances

This list contains the MRP instances of this node.

MRP Instance

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

MRP Domain UUID

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

MRP Role

This parameter contains the value of the attribute administrative MRP role of the ASE object.

MRP Domain Name

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

MRP Prio

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the value of the corresponding attribute of the ASE object.

MRP TOPchgT

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the value of the corresponding attribute of the ASE object.

MRP TOPNRmax

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the value of the corresponding attribute of the ASE object.

MRP TSTshortT

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the value of the corresponding attribute of the ASE object.

MRP TSTdefaultT

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the value of the corresponding attribute of the ASE object.

MRP TSTNRmax

This conditional attribute is present when the value of the operational MRP role attribute is “MANAGER” or “MANAGER_AUTO”. It contains the value of the corresponding attribute of the ASE object.

MRP LNKdownT

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the value of the corresponding attribute of the ASE object.

MRP LNKupT

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the value of the corresponding attribute of the ASE object.

MRP LKNRmax

This conditional attribute is present when the value of the operational MRP role attribute is “CLIENT”. It contains the value of the corresponding attribute of the ASE object.

7.3.3.5.5.5 Read PD Interface MRP Data Adjust

This service allows reading PD Interface MRP Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to interface submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 265 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Interface MRP Data Adjust in 7.3.3.5.5.4.

7.3.3.5.5.6 Read PD Port MRP Data Real

This service allows reading PD Port MRP Data Real using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 266 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 266 – Data elements of Read PD Port MRP Data Real

Parameter name	service specific elements	rsp	cnf
Data	MRP Instance	M	M(=)
	MRP Domain UUID	M	M(=)

Data

This parameter contains the value of the object which has been read.

MRP Instance

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

MRP Domain UUID

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

7.3.3.5.5.7 Write PD Port MRP Data Adjust

This service allows writing PD Port MRP Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 267 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 267 – Data elements of Write PD Port MRP Data Adjust

Parameter name	service specific elements	rsp	cnf
Data	MRP Instance	M	M(=)
	MRP Domain UUID	M	M(=)

Data

This parameter contains the value of the object which may be used to write.

MRP Instance

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

MRP Domain UUID

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

7.3.3.5.5.8 Read PD Port MRP Data Adjust

This service allows reading PD Port MRP Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 267 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Port MRP Data Adjust in 7.3.3.5.5.7.

7.3.3.5.5.9 Read PD Port MRPIC Data Real

This service allows reading PD Port MRPIC Data Real using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy, 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 268 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 268 – Data elements of Read PD Port MRPIC Data Real

Parameter name	service specific elements	rsp	cnf
Data	MRPIC Domain ID	M	M(=)
	MRPIC Role	M	M(=)
	MRPIC Domain Name	M	M(=)
	MRPIC MIM Params	C	C(=)
	MRPIC State Data	C	C(=)
	MRPIC MIC Params	C	C(=)

Data

This parameter contains the value of the object which has been read.

MRPIC Domain ID

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

MRPIC Role

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

MRPIC Domain Name

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

MRPIC MIM Params

This conditional attribute is present when the value of the MRPIC role attribute is “MIM”. It contains the value of the corresponding attribute of the ASE object.

MRPIC State Data

This conditional attribute is present when the value of the MRPIC role attribute is “MIM”. It contains the value of the corresponding attribute of the ASE object.

MRPIC MIC Params

This conditional attribute is present when the value of the MRPIC role attribute is “MIC”. It contains the value of the corresponding attribute of the ASE object.

7.3.3.5.5.10 Write PD Port MRPIC Data Check

This service allows writing PD Port MRPIC Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy, 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 269 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 269 – Data elements of Write PD Port MRPIC Data Check

Parameter name	service specific elements	rsp	cnf
Data	MRPIC Check	M	M(=)

Data

This parameter contains the value of the object which may be used to write.

MRPIC Check

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

7.3.3.5.5.11 Read PD Port MRPIC Data Check

This service allows reading PD Port MRPIC Data Check using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy, 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 267 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Port MRPIC Data Check in 7.3.3.2.5.18.

7.3.3.5.5.12 Write PD Port MRPIC Data Adjust

This service allows writing PD Port MRPIC Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the write service (see 7.3.1.6.5.2.1); for the error code hierarchy, 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning; see Table 270 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Table 270 – Data elements of Write PD Port MRPIC Data Adjust

Parameter name	service specific elements	rsp	cnf
Data	MRPIC Domain ID	M	M(=)
	MRPIC Role	M	M(=)
	MRPIC Domain Name	M	M(=)
	MRPIC MIM Params	C	C(=)
	MRPIC MIC Params	C	C(=)

Data

This parameter contains the value of the object which may be used to write.

MRPIC Domain ID

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

MRPIC Role

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

MRPIC Domain Name

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

MRPIC MIM Params

This conditional attribute is present when the value of the MRPIC role attribute is “MIM”. It contains the value of the corresponding attribute of the ASE object.

MRPIC MIC Params

This conditional attribute is present when the value of the MRPIC role attribute is “MIC”. It contains the value of the corresponding attribute of the ASE object.

7.3.3.5.5.13 Read PD Port MRPIC Data Adjust

This service allows reading PD Port MRPIC Data Adjust using a record data object with subslot scope (see 7.3.1.6.1). The service is applicable to port submodules only.

The parameter structure and the meaning of the parameters is the same as described in the read service (see 7.3.1.6.5.1.1); for the error code hierarchy, 7.3.1.6.6.1 applies. The parameter Data has a service specific structure and meaning, see Table 267 for its structure.

The AR context is of no meaning for this service; see 7.3.1.6.6.4 for details.

Data

See Write PD Port MRPIC Data Adjust in 7.3.3.2.5.20.

7.3.3.5.6 Behavior**7.3.3.5.6.1 MRP**

MRP defines different roles for the devices inside a ring topology.

For media redundancy the IEC 62439-2 applies. The support of the MRP Management Information Base (MIB) is optional.

The Media redundancy ASE specifies a uniform method of achieving media redundancy with deterministic recovery time in case of a single network failure (link failure, switch node failure causing link failures).

NOTE Media redundancy operates transparent to the application. No changes in the application are required.

The Media Redundancy Protocol (MRP) is designed to react deterministic on a single failure of a trunk link or node in the network.

A MRP compliant network shall have a ring topology with multiple nodes.

Each node in the ring requires an integrated switch with at least two ports (ringports) connected to the ring.

Each node in the ring shall be able to detect the failure or recovery of a trunk link or the failure or recovery of a neighbor node.

One of the nodes acts as media redundancy manager (MRM). The function of the MRM is to observe and to control the ring topology in order to react on network faults. The MRM does

this by sending frames on one ringport over the ring and receiving them from the ring over its other ringport, and viceversa in the other direction.

The other nodes in the ring have the role of Media Redundancy Clients (MRC). A MRC reacts on received reconfiguration frames from the MRM and can detect and signal link changes on its ringports.

A node with administrative role MANAGER will always act as MRM and will have operational role MANAGER.

By adjusting one, some, or even all devices in the ring to administrative role MANAGER_AUTO, these devices select one of them to act as MRM and all others to act as MRC. The one acting as MRM will have operational role MANAGER_AUTO, the others operational role CLIENT. All nodes in administrative role MANAGER_AUTO will recognize a fail of the node with operational role MANAGER_AUTO due to missing test frames passing by, and will start a new selection of the one acting as MRM.

A node with administrative role CLIENT will always act as MRC and will have operational role CLIENT.

A node with administrative role DISABLED will not participate in the MRP protocol and will have operational role DISABLED.

A device supporting MRP starts “out of the box” (i.e. without configuration) having MRP enabled using the role as follows (in falling priority):

- If the device supports role MANAGER_AUTO then the “Out of the box” role should be MANAGER_AUTO.
- If the device supports role CLIENT then the “Out of the box” role should be CLIENT.
- If the device supports only role MANAGER then the “Out of the box” role should be DISABLED.

This behavior applies also for ResetToFactory/FactoryReset.

If MRP needs to be disabled, engineering shall set the MRP Role explicitly to DISABLED.

The MRP consists of a service and a protocol entity. Both are described by IEC 62439-2.

7.3.3.5.6.2 MRPIC

It is possible to redundantly interconnect two or more MRP-rings via the MRP Interconnection Protocol. The principle of the MRP Interconnection Protocol is explained and defined in IEC 62439-2. To ease the introduction to the topic, a short overview is given hereunder.

An MRP Interconnection setup consists at a minimum of two rings and two redundant interconnection links between these rings. The redundant interconnection links are provided by four dedicated devices supporting the MRP Interconnection protocol. The roles of these four devices must be:

- one Media Redundancy Interconnection Manager (MIM),
- and three Media Redundancy Interconnection Clients (MIC).

All four devices must additionally take one of the basic operational MRP roles (MRC or MRM) as they are part of the interconnected MRP rings.