

Industrial communication networks — Fieldbus specifications —

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

ICS 25.040.40; 35.100.70

National foreword

This British Standard is the UK implementation of EN 61158-6-10:2008. It is identical with IEC 61158-6-10:2007. Together with all of the other sections of BS EN 61158-6, it supersedes BS EN 61158-6:2004 which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee AMT/7, Industrial communications — Process measurement and control, including Fieldbus.

A list of organizations represented on this committee can be obtained on request to its secretary.

This publication does not purport to include all the necessary provisions of a contract. Users are responsible for its correct application.

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 June 2008

© BSI 2008

ISBN 978 0 580 61630 3

Amendments/corrigenda issued since publication

Date	Comments

English version

**Industrial communication networks -
Fieldbus specifications -
Part 6-10: Application layer protocol specification -
Type 10 elements
(IEC 61158-6-10:2007)**

Réseaux de communication industriels -
Spécifications des bus de terrain -
Partie 6-10: Spécification des services
des couches d'application -
Éléments de type 10
(CEI 61158-6-10:2007)

Industrielle Kommunikationsnetze -
Feldbusse -
Teil 6-10: Protokollspezifikation
des Application Layer
(Anwendungsschicht) -
Typ 10-Elemente
(IEC 61158-6-10:2007)

This European Standard was approved by CENELEC on 2008-02-01. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the Central Secretariat or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the Central Secretariat has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

CENELEC

European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

Central Secretariat: rue de Stassart 35, B - 1050 Brussels

Foreword

The text of document 65C/476/FDIS, future edition 1 of IEC 61158-6-10, prepared by SC 65C, Industrial networks, of IEC TC 65, Industrial-process measurement, control and automation, was submitted to the IEC-CENELEC parallel vote and was approved by CENELEC as EN 61158-6-10 on 2008-02-01.

This and the other parts of the EN 61158-6 series supersede EN 61158-6:2004.

With respect to EN 61158-6:2004 the following changes were made:

- deletion of Type 6 fieldbus for lack of market relevance;
- addition of new fieldbus types;
- partition into multiple parts numbered 6-2, 6-3, ...6-20.

The following dates were fixed:

- latest date by which the EN has to be implemented
at national level by publication of an identical
national standard or by endorsement (dop) 2008-11-01
- latest date by which the national standards conflicting
with the EN have to be withdrawn (dow) 2011-02-01

NOTE Use of some of the associated protocol types is restricted by their intellectual-property-right holders. In all cases, the commitment to limited release of intellectual-property-rights made by the holders of those rights permits a particular data-link layer protocol type to be used with physical layer and application layer protocols in type combinations as specified explicitly in the EN 61784 series. Use of the various protocol types in other combinations may require permission from their respective intellectual-property-right holders.

IEC and CENELEC draw attention to the fact that it is claimed that compliance with this standard may involve the use of patents as follows, where the [xx] notation indicates the holder of the patent right:

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

WO publication	Title (WO)
WO 99/046908	Local network, especially ethernet network, with redundancy properties and redundancy manager

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

WO publication	Title (WO)
WO 00/026731	Automation system and method for accessing the functionality of hardware components
WO 02/043336	System and method for the parallel transmission of real-time critical and non real-time critical data via switched data networks especially ethernet
WO 02/076033	Synchronous, clocked communication system with local input/output components and method for integrating local input/output components into such a system
WO 03/028258	Method for synchronising nodes of a communication system
WO 03/028259	Communications system and method for synchronising a communications cycle
WO 04/030284	Method for permanent redundant transmission of data telegrams in communication systems
EP 1453230	Synchronisation in einem schaltbaren Datennetz

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

The holders of these patent rights have assured IEC that they are willing to negotiate licences under reasonable and non-discriminatory terms and conditions with applicants throughout the world. In this respect, the statement of the holders of these patent rights are registered with IEC. Information may be obtained from:

[HI]: Hirschmann Automation and Control GmbH
Stuttgarter Straße 45-51
D-72654 Neckartenzlingen
Germany

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

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

Annex ZA has been added by CENELEC.

Endorsement notice

The text of the International Standard IEC 61158-6-10:2007 was approved by CENELEC as a European Standard without any modification.

CONTENTS

INTRODUCTION.....	19
1 Scope.....	20
1.1 General.....	20
1.2 Specifications.....	20
1.3 Conformance.....	20
2 Normative references	21
3 Terms, definitions, abbreviations, symbols and conventions	23
3.1 Referenced terms and definitions	23
3.2 Additional terms and definitions for distributed automation	23
3.3 Additional terms and definitions for decentralized periphery	25
3.4 Additional abbreviations and symbols for distributed automation	32
3.5 Additional abbreviations and symbols for decentralized periphery	33
3.6 Additional abbreviations and symbols for media redundancy	34
3.7 Conventions	34
3.8 Conventions used in state machines	41
4 Application layer protocol specification for common protocols.....	44
4.1 FAL syntax description	44
4.2 Transfer syntax	47
4.3 Discovery and basic configuration	56
4.4 Precision time control.....	78
4.5 Media redundancy.....	146
4.6 Real-time cyclic.....	174
4.7 Real-time acyclic.....	191
4.8 Remote procedure call	202
4.9 Link layer discovery.....	217
4.10 MAC bridges	225
4.11 Virtual bridges	248
4.12 IP suite.....	255
4.13 Domain name system	258
4.14 Dynamic host configuration	258
4.15 Simple network management.....	258
4.16 Common DLL mapping protocol machines (DMPM)	268
5 Application layer protocol specification for distributed automation	278
5.1 FAL syntax description	278
5.2 Transfer syntax	302
5.3 FAL protocol state machines	304
5.4 AP context state machine	305
5.5 FAL service protocol machines (FSPM)	305
5.6 Application relationship protocol machine (ARPM).....	389
5.7 DLL mapping protocol machines (DMPMs)	393
5.8 Protocol options	397
6 Application layer protocol specification for decentralized periphery.....	398
6.1 FAL syntax description	398
6.2 Transfer syntax	407
6.3 FAL protocol state machines	525
6.4 AP-Context state machine	526
6.5 FAL service protocol machines (FSPMs)	526

6.6	Application relationship protocol machines (ARPMs)	591
6.7	DLL mapping protocol machines (DMPMs)	682
Annex A (informative)	Filtering data base (FDB)	683
Annex B (informative)	Establishing of a companion AR	686
Annex C (informative)	Establishing of a device access AR	687
Annex D (informative)	Establishing of an AR (simple procedure)	688
Annex E (informative)	Establishing of an AR (accelerated procedure)	689
Annex F (informative)	Establishing of an AR (fast startup procedure)	691
Annex G (informative)	Example of the upload, storage, and retrieval procedure	692
Annex ZA (normative)	Normative references to international publications with their corresponding European publications	695
Bibliography		694
Figure 1	Common structure of specific fields	38
Figure 2	Common structure of specific fields for octet 1 (high)	39
Figure 3	Common structure of specific fields for octet 2 (low)	39
Figure 4	Common structure of specific fields for octet 1 (high)	40
Figure 5	Common structure of specific fields for octet 2	40
Figure 6	Common structure of specific fields for octet 3	40
Figure 7	Common structure of specific fields for octet 4 (low)	41
Figure 8	Coding of the data type BinaryDate	48
Figure 9	Encoding of Time Of Day value	49
Figure 10	Encoding of Time Difference value	49
Figure 11	Encoding of Network Time value	50
Figure 12	Encoding of Network Time Difference value	50
Figure 13	Timescale correspondence between PTCP_Time and CycleCounter	84
Figure 14	Message timestamp point	87
Figure 15	Four message timestamps	87
Figure 16	Line delay protocol with follow up	88
Figure 17	Line delay protocol without follow up	88
Figure 18	Line delay measurement	90
Figure 19	Model parameter for GSDML usage	91
Figure 20	Bridge delay measurement	92
Figure 21	Delay accumulation	92
Figure 22	Worst case accumulated time deviation of synchronization	93
Figure 23	Scheme for measurement of deviation	93
Figure 24	Measurement of deviation	94
Figure 25	Sending Sync-Frame without Follow Up-Frame	94
Figure 26	Sending Sync-Frame with FollowUp-Frame	95
Figure 27	Forwarding Sync- and FollowUp-Frame	95
Figure 28	Transition between Synchronization Variants	96
Figure 29	State transition diagram of delay request	97
Figure 30	State transition diagram of delay response	105
Figure 31	Overview of PTCP	

Figure 32 – State transition diagram of BMA	112
Figure 33 – State transition diagram of MSM	129
Figure 34 – State transition diagram of BRC	139
Figure 35 – State transition diagram of SRX	143
Figure 36 – MRM protocol machine for MRP	151
Figure 37 – MRC protocol machine	157
Figure 38 – MRM protocol machine.....	166
Figure 39 – MRC protocol machine for MRRT	172
Figure 40 – Structure of the CycleCounter	176
Figure 41 – Structuring of the protocol machines within the DMPM (bridge)	269
Figure 42 – Error message structure	278
Figure 43 – Coding scheme of ITEMQUALITYDEF	283
Figure 44 – Relationship among protocol machines	305
Figure 45 – State transition diagram of FSPM	319
Figure 46 – State transition diagram of ARPM	391
Figure 47 – State transition diagram of DMPM	396
Figure 48 – Classification of diagnosis, maintenance and qualified	472
Figure 49 – Definition of PLL window	485
Figure 50 – Detection of dropped frames — appear	494
Figure 51 – Detection of dropped frames — disappear.....	494
Figure 52 – Relationship among Protocol Machines	525
Figure B.1 – Establishing of a companion AR	686
Figure C.1 – Establishing of a device access AR	687
Figure D.1 – Accelerated establishing of an IOAR (simple procedure).....	688
Figure E.1 – Accelerated establishing of an IOAR without error	689
Figure E.2 – Accelerated establishing of an IOAR with “late” error	690
Figure F.1 – Establishing of an IOAR using fast startup	691
Figure G.1 – Example of upload with storage	692
Figure G.2 – Example of retrieval with storage.....	693
Table 1 – State machine description elements	41
Table 2 – Description of state machine elements	42
Table 3 – Conventions used in state machines	42
Table 4 – IEEE 802.3 DLPDU syntax	44
Table 5 – IEEE 802.11 DLPDU syntax	45
Table 6 – IEEE 802.15.1 DLPDU syntax	45
Table 7 – SourceAddress.....	51
Table 8 – DCP_MulticastMACAdd	51
Table 9 – PTCP_MulticastMACAdd	52
Table 10 – MRP OUI.....	52
Table 11 – MRPMulticastMACAdd	53
Table 12 – MRRTMulticastMACAdd	53
Table 13 – LT (Length/Type).....	53

Table 14 – TagControllInformation.Priority	54
Table 15 – FrameID range 1	54
Table 16 – FrameID range 2	54
Table 17 – FrameID range 3	55
Table 18 – FrameID range 4	55
Table 19 – FrameID range 5	55
Table 20 – FrameID range 6	55
Table 21 – FrameID range 7	56
Table 22 – FrameID range 8	56
Table 23 – FrameID range 9	56
Table 24 – DCP APDU syntax	57
Table 25 – DCP substitutions	58
Table 26 – ServiceID	60
Table 27 – ServiceType for request	60
Table 28 – ServiceType for response	60
Table 29 – List of options	61
Table 30 – List of suboptions	62
Table 31 – SuboptionDHCP	64
Table 32 – BlockQualifier with option IP	65
Table 33 – BlockQualifier with option DeviceProperties, DHCP, and ManufacturerSpecific	65
Table 34 – BlockError	65
Table 35 – BlockInfo for SuboptionIPParameter	66
Table 36 – Bit 1 and Bit 0 of BlockInfo for SuboptionIPParameter	66
Table 37 – Bit 7 of BlockInfo for SuboptionIPParameter	66
Table 38 – BlockInfo for all other suboptions	66
Table 39 – DeviceInitiativeValue	66
Table 40 – SignalValue	67
Table 41 – DeviceRoleDetails	68
Table 42 – IPAddress	68
Table 43 – Subnetmask	69
Table 44 – StandardGateway	69
Table 45 – DCPUCS state table	72
Table 46 – DCPUCR state table	74
Table 47 – DCPMCS state table	76
Table 48 – DCPMCR state table	78
Table 49 – PTCP APDU syntax	79
Table 50 – PTCP substitutions	79
Table 51 – PTCP_TLVHeader.Type	80
Table 52 – PTCP_Delay10ns	80
Table 53 – PTCP_Delay1ns.Value	81
Table 54 – PTCP_Delay1ns.Sign	81
Table 55 – PTCP_CumulativeFrequencyOffset	81

Table 56 – PTCP_SequenceID	82
Table 57 – PTCP_SubType for OUI (=00-0E-CF)	82
Table 58 – PTCP_NanoSeconds	82
Table 59 – PTCP_Flags.LeapSecond	83
Table 60 – Timescale correspondence between MJD, UTC, and PTCP_EPOCHNumber	83
Table 61 – Timescale correspondence between PTCP_EPOCHNumber, PTCP_Second, PTCP_Nanosecond, CycleCounter, and SendClockFactor	83
Table 62 – PTCP_ClockStratum	85
Table 63 – PTCP_ClockRole	85
Table 64 – PTCP_RequestPortID	86
Table 65 – PTCP_SyncID	86
Table 66 – PTCP_T2TimeStamp	86
Table 67 – DelayRequest state table	99
Table 68 – Macros used by DelayRequest	103
Table 69 – Functions used by DelayRequest	104
Table 70 – DelayResponse state table	106
Table 71 – Macros used by DelayResponse	109
Table 72 – Functions used by DelayResponse	110
Table 73 – BMA state table	114
Table 74 – BMA state table – check remote	121
Table 75 – BMA state table – check local vs. remote	122
Table 76 – Macros used by BMA	124
Table 77 – Functions used by BMA	126
Table 78 – BMA_Sync_Ind	126
Table 79 – BMA_Announce_Ind	127
Table 80 – MSM state table	130
Table 81 – Macros used by MSM	135
Table 82 – Functions used by MSM	136
Table 83 – MSM_Role_Ind	136
Table 84 – MSM_Sync	137
Table 85 – MSM_Announce	138
Table 86 – BRC state table	140
Table 87 – Macros used by BRC	142
Table 88 – Macros used by the BRC	142
Table 89 – BRC_Sync_Ind	142
Table 90 – SRX state table	144
Table 91 – Functions used by SRX	145
Table 92 – MRP APDU syntax	146
Table 93 – MRP substitutions	146
Table 94 – MRP_TLVHeader.Type	147
Table 95 – MRP_Prio	147
Table 96 – MRP_PortRole	147
Table 97 – MRP_RingState	148

Table 98 – MRP_Interval	148
Table 99 – MRP_Transition.....	148
Table 100 – MRP_TimeStamp	148
Table 101 – MRP_DomainUUID	149
Table 102 – MRRT APDU syntax	149
Table 103 – MRRT substitutions	149
Table 104 – MRRT_TLVHeader.Type.....	150
Table 105 – MRRT_DomainUUID.....	150
Table 106 – Local variables of MRM protocol machine.....	152
Table 107 – MRM state machine	152
Table 108 – Local variables of MRC protocol machine	158
Table 109 – MRC state machine	158
Table 110 – Functions	163
Table 111 – FDB Clear Timer	165
Table 112 – Topology Change Timer	165
Table 113 – Local variables of MRM Protocol Machine for MRRT Activation	167
Table 114 – MRM state machine for MRRT Activation.....	167
Table 115 – MRC state machine for MRRT Activation	173
Table 116 – MRM and MRC functions	174
Table 117 – RTC APDU syntax	175
Table 118 – RTC substitutions	175
Table 119 – CycleCounter Difference.....	176
Table 120 – DataStatus.State	177
Table 121 – DataStatus.DataValid	177
Table 122 – DataStatus.ProviderState	177
Table 123 – DataStatus.StationProblemIndicator	177
Table 124 – TransferStatus for RT_CLASS_3	178
Table 125 – IOxS.Extension.....	179
Table 126 – IOCS.Instance	179
Table 127 – IOxS.DataState	179
Table 128 – PPM state table	181
Table 129 – Functions used by the PPM	184
Table 130 – CPM state table	185
Table 131 – Functions used by the CPM	191
Table 132 – RTA APDU syntax	191
Table 133 – RTA substitutions	192
Table 134 – PDUType.Type	192
Table 135 – PDUType.Version.....	193
Table 136 – APMS state table.....	195
Table 137 – Functions used by the APMS and APMR	198
Table 138 – A_Timer_add.....	198
Table 139 – A_Timer_event	199
Table 140 – A_Timer_remove	199

Table 141 – APMR state table	200
Table 142 – RPC APDU syntax	202
Table 143 – RPC substitutions	203
Table 144 – RPCPacketType	204
Table 145 – RPCFlags	204
Table 146 – RPCFlags2	205
Table 147 – RPCDRep.Character- and IntegerEncoding	205
Table 148 – RPCDRep Octet 2 – Floating Point Representation	205
Table 149 – RPCObjectUUID.Data4	206
Table 150 – RPCObjectUUID – defined values	206
Table 151 – RPCInterfaceUUID – defined values	207
Table 152 – RPCOperationNmb (IO device, controller and supervisor)	208
Table 153 – RPCOperationNmb for endpoint mapper	208
Table 154 – RPCDataRepresentationUUID – defined values	210
Table 155 – RPCInquiryType	211
Table 156 – RPCEPMapStatus	213
Table 157 – Values of NCAFaultStatus	215
Table 158 – Values of NCAResjectStatus	216
Table 159 – LLDP APDU syntax	217
Table 160 – LLDP substitutions	218
Table 161 – LLDP_PNIO_SubType	219
Table 162 – PTCP_PortRxDelayLocal	220
Table 163 – PTCP_PortRxDelayRemote	220
Table 164 – PTCP_PortTxDelayLocal	220
Table 165 – PTCP_PortTxDelayRemote	220
Table 166 – CableDelayLocal	221
Table 167 – RTClass2_PortStatus.State	221
Table 168 – RTClass3_PortStatus.State	221
Table 169 – RTClass3_PortStatus.Mode	222
Table 170 – MRRT_PortStatus.State	222
Table 171 – LLDP_RedPeriodBegin.Offset	222
Table 172 – LLDP_RedPeriodBegin.Valid	222
Table 173 – LLDP_OrangePeriodBegin.Offset	223
Table 174 – LLDP_OrangePeriodBegin.Valid	223
Table 175 – LLDP_GreenPeriodBegin.Offset	223
Table 176 – LLDP_GreenPeriodBegin.Valid	223
Table 177 – LLDP_LengthOfPeriod.Length	224
Table 178 – LLDP_LengthOfPeriod.Valid	224
Table 179 – IFW state table	226
Table 180 – IFW function table	232
Table 181 – S_FW state table	234
Table 182 – S_FW function table	237
Table 183 – S_FU_FW state table	240

Table 184 – S_FU_FW function table	244
Table 185 – Primitives issued by LMPM to MMAC	248
Table 186 – Primitives issued by MMAC to LMPM	248
Table 187 – Primitives issued by MMAC to MAC	248
Table 188 – Primitives issued by MAC to MMAC	249
Table 189 – MMAC state table	250
Table 190 – MMAC function table	254
Table 191 – IP/UDP APDU syntax	256
Table 192 – IP/UDP substitutions	256
Table 193 – UDP_SrcPort	257
Table 194 – UDP_DstPort	257
Table 195 – IP_DstIPAddress	257
Table 196 – IP Multicast DstIPAddress according to RFC 2365	257
Table 197 – Enterprise number	258
Table 198 – LMPM state table	270
Table 199 – LMPM macros table	274
Table 200 – LMPM function table	278
Table 201 – Error messages	279
Table 202 – VARTYPE values	282
Table 203 – ITEMQUALITYDEF values	283
Table 204 – STATEDEF values	286
Table 205 – GROUPEXCEPTIONDEF values	286
Table 206 – ACCESSRIGHTSDEF values	287
Table 207 – PERSISTDEF values	287
Table 208 – UUID values	290
Table 209 – Data format for serialized connection data	303
Table 210 – Calculation of the RT reference data size	304
Table 211 – Primitives issued by FAL User to FSPM	306
Table 212 – Primitives issued by FSPM to FAL User	313
Table 213 – FSPM state descriptions	319
Table 214 – FSPM state table	320
Table 215 – Primitives issued by FSPM to ARPM	390
Table 216 – Primitives issued by ARPM to FSPM	390
Table 217 – Parameters used with primitives exchanged between FSPM and ARPM	391
Table 218 – ARPM state descriptions	391
Table 219 – ARPM state table	392
Table 220 – Primitives issued by ARPM to DMPM	393
Table 221 – Primitives issued by DMPM to ARPM	394
Table 222 – Parameters used with primitives exchanged between ARPM and DMPM	394
Table 223 – Primitives issued by DMPM to ORPC model	395
Table 224 – Primitives issued by ORPC model to DMPM	395
Table 225 – Parameters used with primitives exchanged between DMPM and ORPC model	396

Table 226 – DMPM state descriptions	396
Table 227 – DMPM state table	397
Table 228 – IO APDU substitutions	398
Table 229 – BlockType	408
Table 230 – AlarmType	412
Table 231 – AlarmSpecifier.ChannelDiagnosis	413
Table 232 – AlarmSpecifier.ManufacturerSpecificDiagnosis	413
Table 233 – AlarmSpecifier.SubmoduleDiagnosisState	413
Table 234 – AlarmSpecifier.ARDiagnosisState	414
Table 235 – SlotNumber	414
Table 236 – SubslotNumber	414
Table 237 – Grouping of DiagnosisData	417
Table 238 – Index (user specific)	417
Table 239 – Index (subslot specific)	418
Table 240 – Index (slot specific)	419
Table 241 – Index (AR specific)	420
Table 242 – Index (API specific)	421
Table 243 – Index (device specific)	422
Table 244 – ARTYPE	422
Table 245 – IOCRMulticastMACAdd	423
Table 246 – Type 10 OUI	423
Table 247 – ARProperties.State	424
Table 248 – ARProperties.SupervisorTakeoverAllowed	424
Table 249 – ARProperties.ParametrizationServer	424
Table 250 – ARProperties.DataRate	424
Table 251 – ARProperties.DeviceAccess	425
Table 252 – ARProperties.CompanionAR	425
Table 253 – ARProperties.AcknowledgeCompanionAR	425
Table 254 – ARProperties.PullModuleAlarmAllowed	425
Table 255 – IOCRProperties.RTClass	426
Table 256 – IOCRProperties.MediaRedundancy	426
Table 257 – IOCRTagHeader.IOCRVLANID	427
Table 258 – IOCRTagHeader.IOUserPriority	427
Table 259 – IOCRTYPE	427
Table 260 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=0	427
Table 261 – CMInitiatorActivityTimeoutFactor with ARProperties.DeviceAccess:=1	428
Table 262 – LengthIOCS	428
Table 263 – LengthIOPS	429
Table 264 – AlarmCRProperties.Priority	429
Table 265 – AlarmCRProperties.Transport	429
Table 266 – AlarmCRTagHeaderHigh.AlarmCRVLANID	430
Table 267 – AlarmCRTagHeaderHigh.AlarmUserPriority	430
Table 268 – AlarmCRTagHeaderLow.AlarmCRVLANID	430

Table 269 – AlarmCRTagHeaderLow.AlarmUserPriority	430
Table 270 – AlarmSequenceNumber	431
Table 271 – AlarmCRType	431
Table 272 – RTATimeoutFactor	431
Table 273 – RTARetries	431
Table 274 – AddressResolutionProperties.Protocol	432
Table 275 – AddressResolutionProperties.Factor	432
Table 276 – MCITimeoutFactor	433
Table 277 – ModuleIdentNumber	433
Table 278 – SubmoduleIdentNumber	434
Table 279 – ControlBlockProperties in conjunction with ControlCommand.ApplicationReady	435
Table 280 – ControlBlockProperties in conjunction with the other values of the field ControlCommand	435
Table 281 – ControlCommand.PrmEnd	435
Table 282 – ControlCommand.ApplicationReady	435
Table 283 – ControlCommand.Release	436
Table 284 – ControlCommand.Done	436
Table 285 – ControlCommand.ReadyForCompanion	436
Table 286 – DataDescription.Type	436
Table 287 – Values of ReductionRatio	438
Table 288 – Values of Phase	438
Table 289 – Values of Sequence	439
Table 290 – DataHoldFactor	439
Table 291 – WatchdogFactor	440
Table 292 – Values of FrameSendOffset	440
Table 293 – Values of ErrorCode for negative responses	441
Table 294 – Values of ErrorDecode	441
Table 295 – Coding of ErrorCode1 with ErrorDecode PNIORW	442
Table 296 – Values of ErrorCode1 and ErrorCode2 for ErrorDecode with the value PNIO	443
Table 297 – Values of ErrorCode2 for ErrorCode1 = RPC	449
Table 298 – ModuleState	449
Table 299 – SubmoduleState.AddInfo	449
Table 300 – SubmoduleState.QualifiedInfo	450
Table 301 – SubmoduleState.MaintenanceRequired	450
Table 302 – SubmoduleState.MaintenanceDemanded	450
Table 303 – SubmoduleState.DiagInfo	450
Table 304 – SubmoduleState.ARInfo	451
Table 305 – SubmoduleState.IdentInfo	451
Table 306 – SubmoduleState.FormatIndicator	451
Table 307 – SubmoduleState.Detail	452
Table 308 – SubmoduleProperties.Type	452
Table 309 – SubmoduleProperties.SharedInput	

Table 310 – SubmoduleProperties.ReduceInputSubmoduleDataLength	453
Table 311 – SubmoduleProperties.ReduceOutputSubmoduleDataLength	453
Table 312 – SubmoduleProperties.DiscardIOXS	453
Table 313 – SubstitutionMode	454
Table 314 – SubstituteActiveFlag	454
Table 315 – InitiatorUDPRTPort	454
Table 316 – ResponderUDPRTPort	455
Table 317 – InitiatorRPCServerPort	455
Table 318 – ResponderRPCServerPort	455
Table 319 – IM_Hardware_Revision	456
Table 320 – IM_SWRevision_Functional_Enhancement	456
Table 321 – IM_SWRevision_Bug_Fix	456
Table 322 – IM_SWRevision_Internal_Change	456
Table 323 – IM_Revision_Counter	456
Table 324 – IM_Profile_ID	457
Table 325 – IM_Profile_Specific_Type	457
Table 326 – IM_Version_Major	457
Table 327 – IM_Version_Minor	457
Table 328 – IM_Date	459
Table 329 – UserStructureIdentifier	459
Table 330 – ChannelErrorType	461
Table 331 – ChannelNumber	462
Table 332 – ChannelProperties.Type	463
Table 333 – Valid combinations within ChannelProperties	464
Table 334 – Valid combinations for Alarmnotification and RecordDataRead(DiagnosisData)	465
Table 335 – ChannelProperties.Specifier	466
Table 336 – ChannelProperties.Direction	466
Table 337 – ExtChannelErrorType	466
Table 338 – ExtChannelErrorType for ChannelErrorType 0 – 0x7FFF	466
Table 339 – ExtChannelErrorType for ChannelErrorType “Data transmission impossible”	467
Table 340 – ExtChannelErrorType for ChannelErrorType “Remote mismatch”	467
Table 341 – ExtChannelErrorType for ChannelErrorType “Media redundancy mismatch”	467
Table 342 – ExtChannelErrorType for ChannelErrorType “Sync mismatch” and for ChannelErrorType “Time mismatch”	468
Table 343 – ExtChannelErrorType for ChannelErrorType “Isochronous mode mismatch”	468
Table 344 – ExtChannelErrorType for ChannelErrorType “Multicast CR mismatch”	468
Table 345 – ExtChannelErrorType for ChannelErrorType “Fiber optic mismatch”	469
Table 346 – ExtChannelErrorType for ChannelErrorType “Network component function mismatch”	469
Table 347 – Values for Accumulative Info	469
Table 348 – Values for “Fiber optic mismatch” – “Power Budget”	469

Table 349 – Values for “Network component function mismatch” – “Frame dropped”	470
Table 350 – Values for “Remote mismatch” – “Peer CableDelay mismatch”	470
Table 351 – Values for QualifiedChannelQualifier	471
Table 352 – Values for MaintenanceStatus	471
Table 353 – URRecordIndex	472
Table 354 – URRecordLength	472
Table 355 – LineDelay.Value with LineDelay.FormatIndicator == 0	473
Table 356 – LineDelay.Value with LineDelay.FormatIndicator == 1	473
Table 357 – LineDelay.FormatIndicator	474
Table 358 – RxPort	474
Table 359 – NumberOfTxPortGroups	474
Table 360 – TxPortEntry	475
Table 361 – FrameDetails.SyncFrame	476
Table 362 – FrameDetails.MeaningFrameSendOffset	476
Table 363 – MAUType	477
Table 364 – Valid combinations between MAUType and PortState	478
Table 365 – CheckSyncMode.CableDelay	478
Table 366 – CheckSyncMode.SyncMaster	479
Table 367 – MAUTypeMode.Check	479
Table 368 – DomainBoundary	479
Table 369 – MulticastBoundary	480
Table 370 – PortState	480
Table 371 – MediaType	480
Table 372 – MaxBridgeDelay	481
Table 373 – NumberOfPorts	481
Table 374 – MaxPortTxDelay	481
Table 375 – MaxPortRxDelay	481
Table 376 – SyncProperties.Role	482
Table 377 – SyncProperties.SyncClass in conjunction with Clock Master	482
Table 378 – SyncProperties.SyncClass in conjunction with Clock Slave	482
Table 379 – SyncSendFactor	483
Table 380 – SyncFrameAddress.MulticastSelection for RTASyncPDU	483
Table 381 – SyncFrameAddress.MulticastSelection for RTCSyncPDU	483
Table 382 – SyncFrameAddress.PDUType	484
Table 383 – PTCPTimeoutFactor	484
Table 384 – PLLWindow	484
Table 385 – TimeDataCycle	485
Table 386 – TimeIOInput	485
Table 387 – TimeIOOutput	486
Table 388 – TimeIOInputValid	486
Table 389 – TimeIOOutputValid	486
Table 390 – ControllerApplicationCycleFactor	486
Table 391 – MRP_Role	487

Table 392 – MRP_RTMode.RTClass1_2	487
Table 393 – MRP_RTMode.RTClass3	487
Table 394 – MRRT_TSTdefaultT	488
Table 395 – MRP_TOPchgT	488
Table 396 – MRP_TOPNRmax	488
Table 397 – MRP_TSTshortT	489
Table 398 – MRP_TSTdefaultT	489
Table 399 – MRP_TSTNRmax	489
Table 400 – MRRT_TSTNRmax	490
Table 401 – MRP_LNKdownT	490
Table 402 – MRP_LNKupT	490
Table 403 – MRP_LNKNRmax	491
Table 404 – MRP_RTState	491
Table 405 – MRP_Check.MediaRedundancyManager	491
Table 406 – MRP_Check.MRP_DomainUUID	491
Table 407 – VendorBlockType	492
Table 408 – FiberOpticType	492
Table 409 – FiberOpticCableType	492
Table 410 – FiberOpticPowerBudgetType.Value	493
Table 411 – FiberOpticPowerBudgetType.CheckEnable	493
Table 412 – NCDropBudgetType.Value	493
Table 413 – NCDropBudgetType.CheckEnable	493
Table 414 – FSHelloMode.Mode	494
Table 415 – FSHelloInterval	495
Table 416 – FSHelloRetry	495
Table 417 – FSHelloDelay	496
Table 418 – FSPParameterMode.Mode	496
Table 419 – FSPParameterUUID	496
Table 420 – FSMMode.Mode	497
Table 421 – ArgsLength check	498
Table 422 – ARBlockReq – request check	499
Table 423 – IOCRBlockReq – request check	500
Table 424 – AlarmCRBlockReq – request check	503
Table 425 – ExpectedSubmoduleBlockReq – request check	504
Table 426 – PrmServerBlock – request check	505
Table 427 – MCRBlockReq – request check	506
Table 428 – ARRPCBlockReq – request check	506
Table 429 – ArgsLength check	507
Table 430 – ARBlockRes – response check	507
Table 431 – IOCRBlockRes – response check	508
Table 432 – AlarmCRBlockRes – response check	508
Table 433 – ModuleDiffBlock – response check	509
Table 434 – ArgsLength check	510

Table 435 – ControlBlockConnect – request check	510
Table 436 – ControlBlockPlug – request check	511
Table 437 – ArgsLength check.....	511
Table 438 – ControlBlockConnect – response check.....	512
Table 439 – ControlBlockPlug – response check.....	513
Table 440 – ArgsLength check.....	513
Table 441 – ControlBlockConnect – request check	514
Table 442 – ControlBlockPlug – request check	514
Table 443 – ArgsLength check.....	515
Table 444 – ControlBlockConnect – response check.....	515
Table 445 – ControlBlockPlug – response check.....	516
Table 446 – ArgsLength check.....	516
Table 447 – ReleaseBlock – request check.....	517
Table 448 – ArgsLength check.....	517
Table 449 – ReleaseBlock – response check	518
Table 450 – ArgsLength check.....	518
Table 451 – IODWriteReqHeader – request check	519
Table 452 – ArgsLength check.....	519
Table 453 – IODWriteResHeader – response check.....	520
Table 454 – ArgsLength check.....	520
Table 455 – ArgsLength check.....	521
Table 456 – ArgsLength check.....	522
Table 457 – IODReadReqHeader – request check	523
Table 458 – RecordDataReadQuery – request check	523
Table 459 – ArgsLength check.....	524
Table 460 – IODReadResHeader – response check.....	524
Table 461 – Primitives issued by AP-Context (FAL user) to FSPMDEV	527
Table 462 – Primitives issued by FSPMDEV to AP-Context (FAL user)	535
Table 463 – FSPMDEV protocol machine for multicast communication	542
Table 464 – Functions used by AP-Context (FAL user) to FSPMDEV	549
Table 465 – Function used by FSPMDEV to AP-Context (FAL user)	553
Table 466 – Primitives issued by AP-Context (FAL user) to FSPMCTL.....	560
Table 467 – Primitives issued by FSPMCTL to AP-Context (FAL user).....	565
Table 468 – Function used by AP-Context (FAL user) to FSPMCTL	573
Table 469 – Functions used by FSPMCTL to AP-Context (FAL user)	583
Table 470 – Primitives issued by FSPMDEV or FSPMCTL to ALPMI.....	591
Table 471 – Primitives issued by ALPMI to FSPMDEV or FSPMCTL	591
Table 472 – Primitives issued by CMDEV or CMCTL to ALPMI	592
Table 473 – Primitives issued by ALPMI to CMCTL or CMDEV	592
Table 474 – Primitives issued by APMR to ALPMI.....	593
Table 475 – Primitives issued by ALPMI to APMR.....	593
Table 476 – Primitives issued by APMS to ALPMI.....	594
Table 477 – Primitives issued by ALPMI to APMS.....	594

Table 478 – ALPMI state table	595
Table 479 – Primitives issued by FSPMDEV or FSPMCTL to ALPMR	598
Table 480 – Primitives issued by ALPMR to FSPMDEV or FSPMCTL	598
Table 481 – Primitives issued by CMDEV or CMCTL to ALPMR	598
Table 482 – Primitives issued by ALPMR to CMCTL or CMDEV	599
Table 483 – Primitives issued by APMR to ALPMR	599
Table 484 – Primitives issued by ALPMR to APMR	599
Table 485 – Primitives issued by APMS to ALPMR	600
Table 486 – Primitives issued by ALPMR to APMS	600
Table 487 – ALPMR state table	601
Table 488 – Primitives issued by CMCTL to NRPM	605
Table 489 – Primitives issued by NRPM to CMCTL	606
Table 490 – Primitives issued by other machines to NRPM	607
Table 491 – Primitives issued by NRPM to other machines	608
Table 492 – NRPM state table	609
Table 493 – Functions used by the NRPM and RMPM	615
Table 494 – Primitives issued by CMDEV to RMPM	616
Table 495 – Primitives issued by RMPM to CMDEV	617
Table 496 – Primitives issued by RPC to RMPM	617
Table 497 – Primitives issued by RMPM to RPC	618
Table 498 – Primitives issued by other machines to RMPM	618
Table 499 – Primitives issued by RMPM to other machines	619
Table 500 – RMPM state table	620
Table 501 – Primitives issued by FSPMDEV to CMDEV	630
Table 502 – Primitives issued by CMDEV to FSPMDEV	631
Table 503 – CMDEV state table	632
Table 504 – Macros used by CMDEV	651
Table 505 – Primitives issued by CMDEV to NRMC	652
Table 506 – Primitives issued by NRMC to CMDEV	652
Table 507 – Primitives issued by CPM to NRMC	652
Table 508 – Primitives issued by NRMC to CPM	653
Table 509 – Primitives issued by other machines to NRMC	653
Table 510 – Primitives issued by NRMC to other machines	654
Table 511 – NRMC state table	654
Table 512 – Primitives issued by FSPMCTL to CMCTL	659
Table 513 – Primitives issued by CMCTL to FSPMCTL	660
Table 514 – CMCTL state table	662
Table A-1 – Unicast FDB entries	683
Table A-2 – Multicast FDB entries	683
Table A-3 – Broadcast FDB entry	685

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/TR 61158–1.

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this standard is to provide a set of rules for communication expressed in terms of the procedures to be carried out by peer application entities (AEs) at the time of communication. These rules for communication are intended to provide a sound basis for development in order to serve a variety of purposes:

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

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

1 Scope

1.1 General

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

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

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

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

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

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

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

1.2 Specifications

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

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

1.3 Conformance

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

2 Normative references

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

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

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

IEC 61784-3-3, *Industrial communication networks – Profiles – Part 3-3: Functional safety fieldbuses – Additional specifications for CPF 3*

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

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

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

ISO/IEC 8824, *Information technology – Open Systems Interconnection – Specification of Abstract Syntax Notation One (ASN.1)*

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

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

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

IEEE 802-2001, *IEEE Standards for local and metropolitan area networks: overview and architecture*

IEEE 802.1AB-2005, *IEEE Standards for Local and Metropolitan Area Networks: Station and Media Access Control Connectivity Discovery*

IEEE 802.1D-2004, *IEEE Standards for local and metropolitan area networks – Media access Control (MAC) Bridges*

IEEE 802.1Q-2005, *IEEE Standards for Local and metropolitan area networks – Virtual Bridged Local Area Networks*

IEEE 802.3-2005, *IEEE Standards for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer specifications*

IEEE 802.11-1999, *IEEE Standards for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*

IEEE 802.15.1-2005, *IEEE Standard for Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 15.1: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications for wireless personal area networks (WPANs)*

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

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

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

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

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

IETF RFC 1112, *Host Extensions for IP Multicasting*; available at <<http://www.ietf.org>>

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

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

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

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

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

IETF RFC 2737, *Entity MIB (Version 2)*, available at <<http://www.ietf.org>>

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

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

IETF RFC 3490, *Internationalizing Domain Names in Applications (IDNA)*; available at <<http://www.ietf.org>>

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

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

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

3 Terms, definitions, abbreviations, symbols and conventions

3.1 Referenced terms and definitions

3.1.1 ISO/IEC 7498-1 terms

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

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

3.1.2 ISO/IEC 8822 terms

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

- a) abstract syntax
- b) presentation context

3.1.3 ISO/IEC 8824 terms

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

- a) object identifier
- b) type

3.1.4 ISO/IEC 9545 terms

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

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

3.2 Additional terms and definitions for distributed automation

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

3.2.1

active connection control object

instance of a certain FAL class that abstracts the interconnection facility (as Consumer and Provider) of an automation device

3.2.2

configuration data base

interconnection information maintained by the ACCO ASE

3.2.3**connection**

the logical link between sink and source of attributes and services at different custom interfaces of Custom RT-Auto objects

3.2.4**connection channel**

description of a connection between a sink and a source of data items

3.2.5**consumer**

node or sink that is receiving data from a producer

3.2.6**consumerID**

unambiguous identifier within the scope of the ACCO assigned by the consumer to recognize the internal data of a configured interconnection sink.

3.2.7**data marshaling**

the encoding of parameters of the FAL service primitives with respect to their interface definition

NOTE This is part of the abstract ORPC model.

3.2.8**engineering**

abstract term that characterizes the client application or device responsible for configuring an automation system via interconnecting data items

3.2.9**event**

an instance of a change of conditions

3.2.10**interface**

collection of FAL class attributes and services that represents a specific view on the FAL class

3.2.11**interface definition language**

syntax and semantic of describing service parameters in a formal way

NOTE This description is the input for the ORPC model, especially for the ORPC wire protocol.

3.2.12**interface pointer**

key attribute that unambiguously addresses an object interface instance

3.2.13**logical device**

a certain FAL class that abstracts a software component or a firmware component as an autonomous self-contained facility of an automation device

3.2.14**method**

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

3.2.15**object remote procedure call**

model for object oriented or component based remote method invocation

3.2.16

physical device

a certain FAL class that abstracts the hardware facilities of an automation device

3.2.17

property

a synonym for ASE attributes which are readable or writeable via operational ASE services

NOTE These services are generally named “get_<Attribute Name>” or “set_<Attribute Name>” and correspond with the IDL keywords “propget” and “propput”.

3.2.18

provider

source of a data connection.

3.2.19

providerID

an unambiguous identifier within the scope of the ACCO assigned by the provider to recognize the internal data of a configured connection source

3.2.20

quality code

additional status information of a data item

3.2.21

quality code aware

attribute of the RT-Auto class that indicates that an RT-Auto object uses a status code for its data items

3.2.22

quality code unaware

opposite of quality code aware

3.2.23

RT-Auto

an FAL class that abstracts the automation function as a process-related component of an automation device

3.2.24

runtime object model

objects that exist in a device together with their interfaces and methods that are accessible

3.3 Additional terms and definitions for decentralized periphery

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

3.3.1

alarm

activation of an event that shows a critical state

3.3.2

alarm ack

acknowledgment of an event that shows a critical state

3.3.3

alarm data object

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

3.3.4

allocate

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

3.3.5**application**

function or data structure for which data is consumed or produced

3.3.6**application layer interoperability**

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

3.3.7**application objects**

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

3.3.8**application process**

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

3.3.9**application process identifier**

distinguishes multiple application processes used in a device

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

3.3.10**application process object**

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

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

3.3.11**application process object class**

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

3.3.12**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

3.3.13**application relationship application service element**

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

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

3.3.15**attribute**

description of an externally visible characteristic or feature of an object

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

3.3.16

backup

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

3.3.17

behavior

indication of how an object responds to particular events

3.3.18

channel

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

NOTE 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.

3.3.19

channel related diagnosis

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

EXAMPLE open loop

3.3.20

class

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

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

3.3.21

class attributes

attribute that is shared by all objects within the same class

3.3.22

class code

unique identifier assigned to each object class

3.3.23

class specific service

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

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

3.3.24

clear

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

3.3.25

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

3.3.26

common profile

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

3.3.27

communication data object

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

3.3.28**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

3.3.29**configuration fault**

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

3.3.30**configuration identifier**

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

3.3.31**consume**

act of receiving data from a provider

3.3.32**consumer**

node or sink receiving data from a provider

3.3.33**context management**

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

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

3.3.34**conveyance path**

unidirectional flow of APDUs across an application relationship

3.3.35**cyclic**

repetitive in a regular manner

3.3.36**data consistency**

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

3.3.37**device**

physical hardware connected to the link

NOTE A device may contain more than one node.

3.3.38**device ID**

a vendor assigned device type identification

3.3.39**device profile**

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

3.3.40**diagnosis data object**

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

3.3.41**diagnosis information**

all data available at the server for maintenance purposes

3.3.42

dynamic reconfiguration

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

3.3.43

endpoint

one of the communicating entities involved in a connection

3.3.44

engineering

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

3.3.45

error

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

3.3.46

error class

general grouping for related error definitions and corresponding error codes

3.3.47

error code

identification of a specific type of error within an error class

3.3.48

event

instance of a change of conditions

3.3.49

extended channel related diagnosis

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

EXAMPLE Link Fail

3.3.50

frame

unit of data transfer on a link

3.3.51

identification data object

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

3.3.52

implicit AR endpoint

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

3.3.53

index

address of a record data object within an application process

3.3.54

instance

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

3.3.55

instance attributes

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

3.3.56**instantiated**

object that has been created in a device

3.3.57**invocation**

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

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

3.3.58**IO controller**

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

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

3.3.59**IO data object**

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

3.3.60**IO device**

field device which acts as server for IO operation

3.3.61**IO parameter server**

server for application parameter of IO devices (client)

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

3.3.62**IO subsystem**

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

3.3.63**IO supervisor**

engineering device which manages commissioning and diagnosis of an IO system

3.3.64**IO system**

system composed of all its IO subsystems

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

3.3.65**Isochronous mode**

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

3.3.66**member**

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

3.3.67**message**

synonym for frame

3.3.68**method**

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

3.3.69

module

hardware or logical component of a physical device

3.3.70

network

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

3.3.71

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 clearly defined interface and behavior

3.3.72

object specific service

service unique to the object class which defines it

3.3.73

operate

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

3.3.74

packet

frame

3.3.75

peer

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

3.3.76

physical device

automation or other network device

3.3.77

point-to-point connection

connection that exists between exactly two application objects

3.3.78

primary

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

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

3.3.79

provider

node or source sending data to one or many consumer

3.3.80

PTCP domain

certain number of PTCP subdomains in one IP subnet

3.3.81

PTCP subdomain

certain amount of DTEs with synchronized clocks

3.3.82

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

3.3.83**resource**

processing or information capability

3.3.84**run**

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

3.3.85**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

3.3.86**service**

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

3.3.87**slot**

address of a structural unit within an IO device

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

3.3.88**stop**

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

3.3.89**submodule**

hardware or logical component of a module

3.3.90**subslot**

address of a structural unit within a slot

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

3.3.91**vendor ID**

central administrative number used as manufacturer identification

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

3.4 Additional abbreviations and symbols for distributed automation

ACCO	Active Connection Control Object
IDL	Interface Definition Language
IP	Internet Protocol
DNS	Domain Name Service
LDev	Logical Device
ORPC	Object Remote Procedure Call
PDev	Physical Device
QoS	Quality of Service
QC	Quality Code
RT	Runtime
RT-Auto	Runtime Automation Object
TCP	Transmission Control Protocol

3.5 Additional abbreviations and symbols for decentralized periphery

AE	Application Entity
AL	Application Layer
ALME	Application Layer Management Entity
ALP	Application Layer Protocol
ALPMI	Alarm Protocol Machine Initiator
ALPMR	Alarm Protocol Machine Responder
AP	Application Process
APDU	Application Protocol Data Unit
API	Application Process Identifier
APO	Application Object
AR	Application Relationship
AREP	Application Relationship End Point
ARP	Address Resolution Protocol
ASCII	American Standard Code for Information Interchange
ASE	Application Service Element
BMC	Best Master Clock
CM	Context Management
Cnf	Confirmation
CR	Communication Relationship
CREP	Communication Relationship End Point
DCE	OSF Distributed Computing Environment
DCP	Discovery and basic Configuration Protocol
DCPMCR	DCP Multicast Receiver
DCPMCS	DCP Multicast Sender
DCPUCR	DCP Unicast Receiver
DCPUCS	DCP Unicast Sender
DHCP	Dynamic Host Configuration Protocol
DIM	Device Interface Module
DL-	(as a prefix) Data Link-
DLC	Data Link Connection
DLL	Data Link Layer
DLPDU	Data Link-Protocol Data Unit
DLSDU	DL-service-data-unit
DNS	Domain Name Service
DTE	Data Terminal Equipment
FAL	Fieldbus Application Layer
FIFO	First In First Out
GSDML	Generic Station Description Markup Language
I&M	Identification and Maintenance Profile
IANA	Internet Assigned Numbers Authority
ICMP	Internet Control Message Protocol
ID	Identifier
IEC	International Electrotechnical Commission
IFW	RT_CLASS_3 Forwarding Protocol Machine
Ind	Indication
IOCS	Input Output Object Consumer Status
IOPS	Input Output Object Provider Status
IP	Internet Protocol
IR	Isochronous Relay
IRT	Isochronous Real Time Protocol
ISO	International Organization for Standardization
IsoM	Isochronous Mode
LED	Light Emitting Diode
LLDP	Link Layer Discovery Protocol
LME	Layer Management Entity
lsb	Least Significant Bit
LT	Length/Type
MAC	Medium Access Control
msb	Most Significant Bit
NCA	Network Computing Architecture

OSI	Open Systems Interconnect
PDU	Protocol Data Unit
PI	PROFIBUS International see < http://www.profinet.com >
PL	Physical Layer
PTCP	Precision Transparent Clock Protocol
QoS	Quality of Service
Req	Request
RPC	Remote Procedure Call
Rsp	Response
RT	Real Time Protocol
RTA	Real Time Protocol Acyclic
RTC	Real Time Protocol Cyclic
RTE	Real Time Ethernet
SDU	Service Data Unit
TLV	Type Length Value (coding rule)
UDP	User Datagram Protocol
UUID	Universal Unique Identifier
VLAN	Virtual Local Area Network

3.6 Additional abbreviations and symbols for media redundancy

MRC	Media Redundancy Client
MRM	Media Redundancy Master
MRP	Media Redundancy Protocol

3.7 Conventions

3.7.1 General concept

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

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

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

3.7.2 Conventions for distributed automation

3.7.2.1 IDL – Interface Definition Language

3.7.2.1.1 Basic information

The Interface Definition Language (IDL) is a language for specifying interfaces, operations (procedures or functions), parameters to these operations, and data types.

Notation

The IDL is a case-sensitive language. The syntax of IDL is described using an extended BNF (Backus-Naur Form) notation. The meaning of the BNF notation is as follows:

- Brackets ([]) enclose an optional part of the syntax.
- Ellipsis points (...) indicate that the left clause can be repeated either zero or more times if it is optional or one or more times if it is required. The vertical bar (|) indicates alternative productions; it is read as "or."
- Language punctuation that does not conflict with punctuation characters used in the BNF notation appears in a production in the appropriate position. Language punctuation that does conflict with punctuation characters used in the BNF notation is enclosed in less-than and greater-than symbols; for example, <[>. Note particularly that when ' (single quotation) or " (double quotation) appear in a production, they are a part of the language and shall appear in IDL source.

Elements in the grammar that are capitalized are terminals of the grammar. For example, <Identifier> is not further expanded. Also, keywords of the language are terminals of the grammar. For example, the keyword boolean is not further expanded.

3.7.2.1.2 Keywords and reserved words

Some keywords are reserved words, and shall not be used as identifiers. Keywords that are not reserved may be used as identifiers, except when used as attributes (that is, within [] (brackets)). Reserved words are keywords of the language that shall not be used as user identifiers.

IDL reserved words are given in the following list:

boolean	byte	case	char
const	default	double	enum
FALSE	float	handle_t	hyper
import	int	interface	long
NULL	pipe	short	small
struct	switch	TRUE	typedef
union	unsigned	void	

The following list gives the IDL keywords that are reserved when in the context of an attribute: that is, between [] (brackets):

align	broadcast	broadcast	comm_status
context_handle	endpoint	first_is	handle
idempotent	ignore	implicit_handle	in
last_is	length_is	local	max_is
maybe	out	ptr	ref
size_is	string	switch_is	switch_type
transmit_as	uuid	version	

3.7.2.1.3 Identifiers

Each object is named with an unique identifier. The maximum length of an identifier is 31 characters.

Some identifiers are used as a base from which the compiler constructs other identifiers. These identifiers have further restrictions on their length. The character set for identifiers contains the alphabetic characters A to Z and a to z, the digits 0 to 9, and the _ (underbar) character. An identifier shall start with an alphabetic character or the _ (underbar) character.

3.7.2.1.4 IDL punctuation

The punctuation used in IDL consists of the following characters:

- The . (dot)
- The , (comma)
- The pair ()(parentheses)
- The pair [](brackets)
- The pair {}(braces)
- The ; (semicolon)
- The : (colon)
- The * (asterisk)
- The ' (single quote)
- The " (double quote)
- The = (equal sign)

3.7.2.1.5 Interface definition structure

An interface definition written in IDL has the following structure:

```
<interface> ::= <interface_header> { <interface_body> }
```

3.7.2.1.6 Interface header

The structure of the interface header is as follows:

```
<interface_header> ::= [ <interface_attributes> ] > interface <Identifier>
```

where:

```
<interface_attributes> ::= <interface_attribute> [ , <interface_attribute> ] ...
<interface_attribute> ::= uuid ( <Uuid_rep> )
    | version ( <Integer_literal> [ , <Integer_literal> ] )
    | endpoint ( <port_spec> [ , <port_spec> ] ... )
    | local
    | pointer_default ( <ptr_attr> )
<port_spec> ::= <Family_string> : <Port_string>
```

If an interface defines any operations, exactly one of the uuid attribute or the local attribute shall be specified. Whichever is specified shall appear exactly once.

It is permissible to have neither the uuid nor the local attribute if the interface defines no operations. The version attribute may occur at most once.

3.7.2.1.7 Interface body

The structure of the interface body is as follows:

```
<interface_body> ::= [ <import> ... ] <interface_component>
[ <interface_component> ... ]
```

where:

```
<import> ::= import <import_list>;
<interface_component> ::= <export>
    | <op_declarator>;
<export> ::= <type_declarator>;
    | <const_declarator>;
    | <tagged_declarator>;
```

3.7.2.1.8 Further possibilities

It is possible to use comments. The IDL allows to declare constants. Type declarations are subdivided into base types and the fundamental integer, char, boolean, byte, void, handle_t types, constructed types, structures, unions, enumerated types, pipes and arrays.

3.7.3 Conventions for decentralized periphery

3.7.3.1 Abstract syntax conventions

3.7.3.1.1 PDUs are described as octets or groups of octets

- a) Groups of octets separated by a comma appear in the order they are transferred
If optional octets are not present the following octets appear without a gap
- b) If octets or groups of octets are grouped within “{ }” the order is arbitrary
- c) If octets or groups of octets are marked with “*” they may appear more than once.
If it is used within a “{ }” section they may appear mixed with other octets or group of octets of this section.
- d) Octets can be grouped or values can be assigned within “()”
- e) If octets or groups of octets are grouped within “[]” the group can be omitted
- f) Complex APDUs may be built by means of substitutions (sub-structures)
- g) Exclusive selections of octets or groups of octets are separated by “^”

NOTE 1 The formal PDU example

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

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

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

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

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

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

Variant 5: Octet1, OctetGroup1, Octet3, Octet4

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

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

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

3.7.3.2 Convention for the encoding of reserved bits and octets

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

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

3.7.3.3 Conventions for the common codings of specific field octets

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

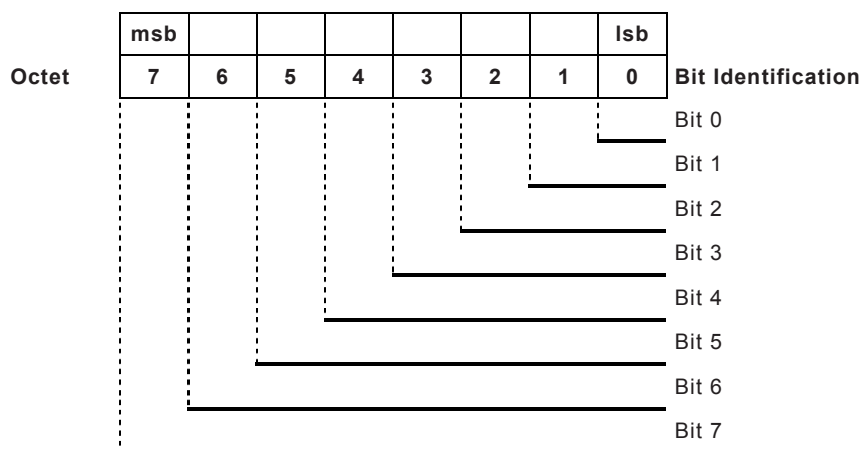


Figure 1 – Common structure of specific fields

Several bit may be grouped as group of bit. Each bit or group of bits shall be addressed by its Bit Identification (e.g. Bit 0, Bit 1 to 4). The position within the octet shall be according to the figure above. Alias names may be used for each bit or group of bits or they may be marked as reserved. The grouping of individual bits shall be in ascending order without gaps. The values for a group of bits may be represented as binary, decimal or hexadecimal values. This value shall only be valid for the grouped bits and can only represent the whole octet if all 8 bits are grouped. Decimal or hexadecimal values shall be transferred in binary values so that the bit with the highest number of the group represents the msb concerning the grouped bit.

EXAMPLE 1 Description and relation for the specific field octet

Bit 0: reserved.

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

Bit 4-7: shall always set to one.

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

(msb) Bit 7 = 1,

Bit 6 = 1,

Bit 5 = 1,

Bit 4 = 1,

Bit 3 = 0,

Bit 2 = 1,

Bit 1 = 0,

(lsb) Bit 0 = 0.

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

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

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

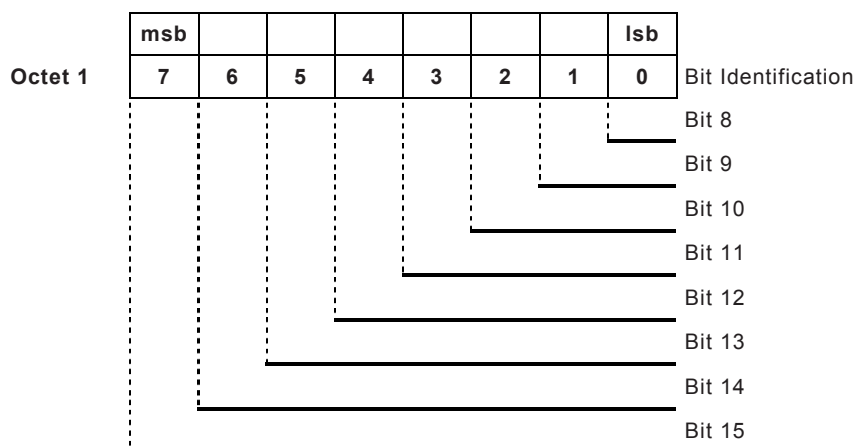


Figure 2 – Common structure of specific fields for octet 1 (high)

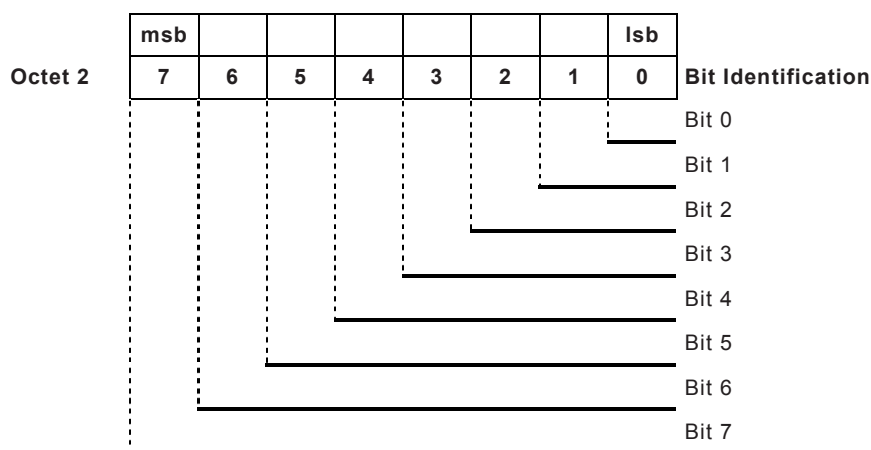


Figure 3 – Common structure of specific fields for octet 2 (low)

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

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

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

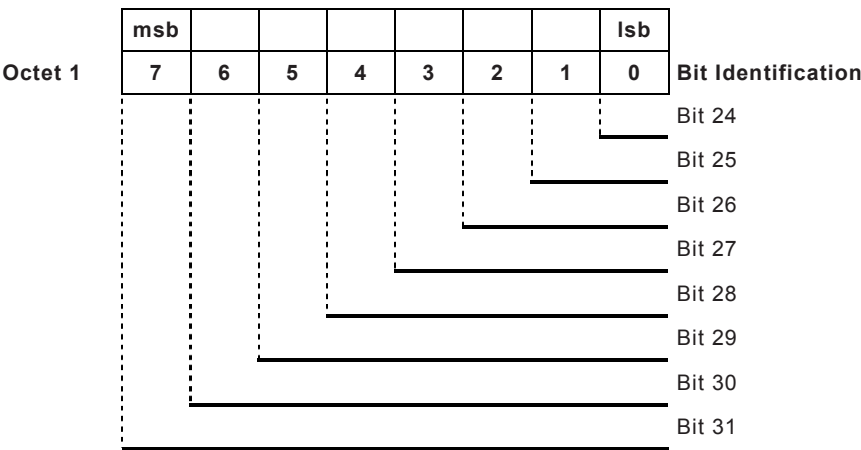


Figure 4 – Common structure of specific fields for octet 1 (high)

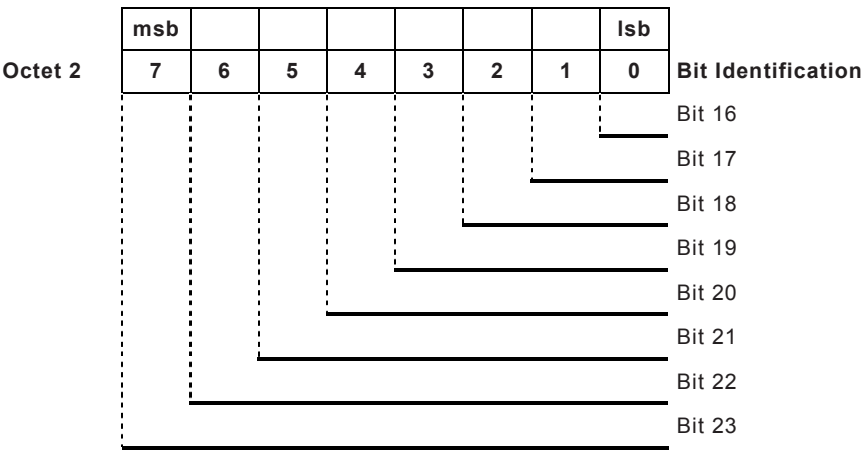


Figure 5 – Common structure of specific fields for octet 2

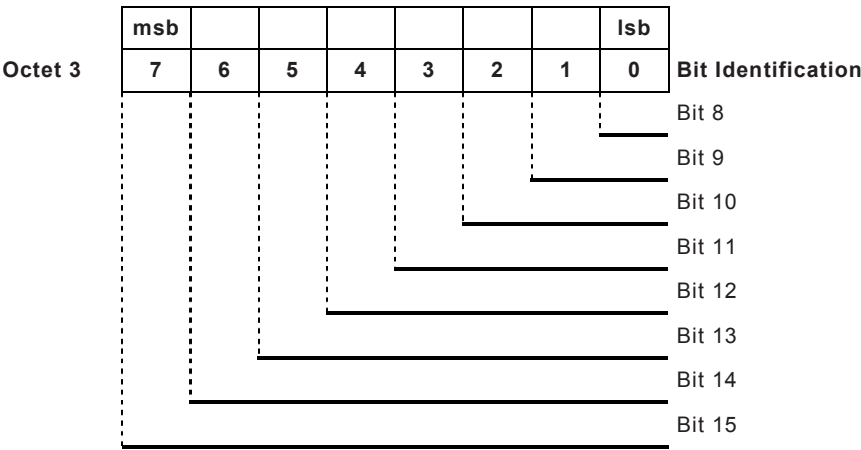


Figure 6 – Common structure of specific fields for octet 3

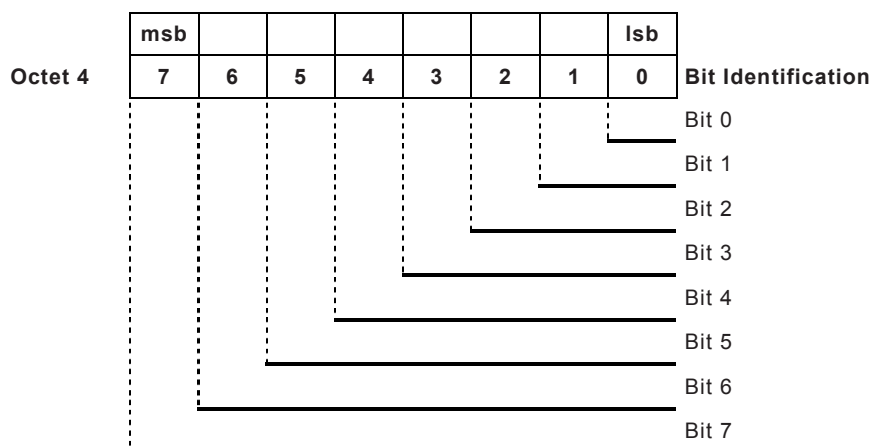


Figure 7 – Common structure of specific fields for octet 4 (low)

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

3.8 Conventions used in state machines

The protocol sequences are described by means of State Machines.

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

The 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 are shown in Table 2.

Table 1 – State machine description elements

#	Current state	Event or condition => action	Next state

Table 2 – Description of state machine elements

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

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

Table 3 – Conventions used in state machines

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

Readers are strongly recommended to refer to the 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 Application layer protocol specification for common protocols

4.1 FAL syntax description

4.1.1 DLPDU abstract syntax reference

4.1.1.1 General

Table 4, Table 5, and Table 6 give an outline of the abstract syntax of the DLPDU according to IEEE 802.3, IEEE 802.11 and IEEE 802.15.1.

4.1.1.2 IEEE 802.3

The encoding and decoding of the fields in Table 4 shall be according to IEEE 802.3 for the DLPDU.

Table 4 – IEEE 802.3 DLPDU syntax

DLPDU name	DLPDU structure
DLPDU	Preamble ^a , StartFrameDelimiter, DestinationAddress, SourceAddress, DLSDU ^b , DLPDU_Padding* ^c , FrameCheckSequence
DLSDU	[VLAN] ^d , LT, SAPDU ^e FIDAPDU
SAPDU	UDP-RTC-PDU ^e UDP-RTA-PDU ^e MRP-PDU ^e CL-RPC-PDU ^e LLDP-PDU ^e ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^e RTA-PDU ^e DCP-PDU ^e PTCP-PDU ^e MRRT-PDU
VLAN	LT(=0x8100), TagControlInformation
NOTE 1 According to IEEE 802.3 the DLPDUs have a minimum length of 64 octets (excluded Preamble, Start Frame Delimiter).	
NOTE 2 For the IEEE 802.3 frames with VLAN tag is the minimum frame size increased to 68 octets in order to guaranty the minimum frame size of 64 octets after removing the VLAN tag by a bridge.	
^a The field contains at least 7 octets	
^b The minimum DLSDU size is 2 octets.	
^c The number of padding octets shall be in the range of 0..46 depending on the DLSDU size. The value shall be set to zero.	
^d The VLAN field can be omitted in case of optimized RT transportation. The field VLAN may be set by the encoder but it may be discarded by intermediate bridges. The decoder shall accept DLPDUs with or without VLAN fields. The VLAN field shall be omitted for RT_CLASS_3 PDUs.	
^e The VLAN field should be omitted.	

4.1.1.3 IEEE 802.11

The encoding and decoding of the fields in Table 5 shall be according to IEEE 802.11 for the DLPDU.

Table 5 – IEEE 802.11 DLPDU syntax

DLPDU name	DLPDU structure
DLPDU	DLPDU_1 ^ DLPDU_2 ^ DLPDU_3 ^ DLPDU_4
DLPDU_1	FrameControl, DestinationAddress, SourceAddress, BSSID, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_2	FrameControl, DestinationAddress, BSSID, SourceAddress, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_3	FrameControl, BSSID, SourceAddress, DestinationAddress, SequenceControl, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLPDU_4	FrameControl, ReceiverAddress, TransmitterAddress, DestinationAddress, SequenceControl, SourceAddress, [QoSControl] ^b , DLSDU, DLPDU_Padding* ^a , FrameCheckSequence
DLSDU	LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ MRP-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU ^ MRRT-PDU
NOTE 1 For definition of FrameControl see IEEE 802.11.	
NOTE 2 For definition of BSSID see IEEE 802.11.	
NOTE 3 For definition of SequenceControl see IEEE 802.11.	
NOTE 4 For definition of QoSControl see IEEE 802.11.	
NOTE 5 For definition of ReceiverAddress see IEEE 802.11.	
NOTE 6 For definition of TransmitterAddress see IEEE 802.11.	
^a The number of padding octets shall be in the range of 0..46 depending on the DLSDU size. The value shall be set to zero.	
^b This field should exist for UDP-RTC-PDU and UDP-RTA-PDU and shall exist for RTC-PDU and RTA-PDU.	

4.1.1.4 IEEE 802.15.1

The encoding and decoding of the fields in Table 6 shall be according to IEEE 802.15.1 for the DLPDU.

Table 6 – IEEE 802.15.1 DLPDU syntax

DLPDU name	DLPDU structure
DLPDUHEADER	Access Code, Packet Header, Payload Header, L2CAP Header, BNEPType(0)
DLPDU	DLPDUHEADER, DestinationAddress, SourceAddress, DLSDU ^a , FrameCheckSequence
DLSDU	[VLAN] ^b , LT, SAPDU ^ FIDAPDU
SAPDU	UDP-RTC-PDU ^ UDP-RTA-PDU ^ MRP-PDU ^ CL-RPC-PDU ^ LLDP-PDU ^c ^ ICMP-PDU
FIDAPDU	FrameID, RTC-PDU ^ RTA-PDU ^ DCP-PDU ^ PTCP-PDU ^c ^ MRRT-PDU
VLAN	LT(=0x8100), TagControlInformation
NOTE 1 For definition of Access Code see IEEE 802.15.1-2005, 8.6.3.	
NOTE 2 For definition of Packet Header see IEEE 802.15.1-2005, 8.6.4.	
NOTE 3 For definition of Payload Header see IEEE 802.15.1-2005, 8.6.6.2.	
NOTE 4 For definition of L2CAP Header see IEEE 802.15.1-2005, 14.3.1.	
NOTE 5 For definition of BNEPType see IEEE 802.15.1-2005.	

DLPDU name	DLPDU structure
^a The minimum DLSDU size is 2 octets. ^b This field should exist for UDP-RTC-PDU and UDP-RTA-PDU and shall exist for RTC-PDU and RTA-PDU. ^c The VLAN field should be omitted.	

4.1.2 Data types

4.1.2.1 Notation for the Boolean type

Boolean ::= BOOLEAN

-- TRUE if the value is non-zero.

-- FALSE if the value is zero.

4.1.2.2 Notation for the Integer type

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

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

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

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

4.1.2.3 Notation for the Unsigned type

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

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

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

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

4.1.2.4 Notation for the Floating Point type

Floating32 ::= BIT STRING SIZE (4) -- IEC 60559 Single precision

Floating64 ::= BIT STRING SIZE (8) -- IEC 60559 Double precision

4.1.2.5 Notation for the OctetString type

OctetString ::= OCTET STRING -- For generic use

4.1.2.6 Notation for VisibleString type

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

4.1.2.7 Notation for BinaryDate type

BinaryDate ::= OctetString SIZE (7)

4.1.2.8 Notation for TimeOfDay type

TimeOfDay with date indication ::= OctetString SIZE (6)

TimeOfDay without date indication ::= OctetString SIZE (4)

4.1.2.9 Notation for TimeDifference type

TimeDifference with date indication ::= OctetString SIZE (6)

TimeDifference without date indication ::= OctetString SIZE (4)

4.1.2.10 Notation for Network Time type

Network Time ::= OctetString SIZE (8)

4.1.2.11 Notation for Network Time Difference type

Network Time Difference ::= OctetString SIZE (8)

4.2 Transfer syntax

4.2.1 Coding of basic data types

4.2.1.1 General Encoding

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

4.2.1.2 Encoding of a Boolean value

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

4.2.1.3 Encoding of an Integer value

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

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

4.2.1.4 Encoding of an Unsigned value

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

NOTE The value of a binary number is derived by numbering the bits in the ContentsOctets, starting with bit 0 of the last octet as bit zero and ending the numbering with bit 7 of the first octet. Each bit is assigned a numerical

value of 2^N , where N is its position in the above numbering sequence. The value of the binary number is obtained by adding the numerical values assigned to each bit for those bits which are set to one.

4.2.1.5 Encoding of a Floating-Point value

- The encoding of a fixed-length Floating-Point value of Floating32 and Floating64 types shall be primitive, and the ContentsOctets shall consist of exactly four, or eight octets, respectively.
- The ContentsOctets shall contain floating-point values defined in conformance with IEC 60559. The sign is encoded in bit 7 of the first octet. It is followed by the exponent starting from bit 6 of the first octet, and then the mantissa starting from bit 6 of the second octet for Floating32 or Floating64.

4.2.1.6 Encoding of a Visible String value

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

4.2.1.7 Encoding of an OctetString value

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

4.2.1.8 Encoding of a BinaryDate value

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

bits octets	7	6	5	4	3	2	1	0	
1	2 ¹⁵	2 ¹⁴	2 ¹³	2 ¹²	2 ¹¹	2 ¹⁰	2 ⁹	2 ⁸	0...59 999 ms
2	2 ⁷	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	
3	RSV	RSV	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0...59 min
4	SU	RSV	RSV	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0...23 hours
5	day of week			day of month					1...7 day of week
	2 ²	2 ¹	2 ⁰	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	1...31 day of month
6	RSV	RSV	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	1...12 months
7	RSV	2 ⁶	2 ⁵	2 ⁴	2 ³	2 ²	2 ¹	2 ⁰	0 ... 50 years 2000 to 2050 51 ... 99 years 1951 to 1999
msb					lsb				
NOTE To avoid the Y2K problem the values of the year are from 0 to 50 for the years 2000 to 2050 and from 51 to 99 for the years 1951 to 1999. This is according to the British Standards Institution (BSi) DISC PD2000-1:1998 "A Definition of Year 2000 Conformance Requirements" Rule 3 (b).									

Figure 8 – Coding of the data type BinaryDate