



BSI Standards Publication

Industrial communication networks — Fieldbus specifications

Part 6-2: Application layer protocol specification — Type 2 elements

National foreword

This British Standard is the UK implementation of EN IEC 61158-6-2:2019. It is identical to IEC 61158-6-2:2019. It supersedes BS EN 61158-6-2:2014, which will be withdrawn on 25 July 2022.

The UK participation in its preparation was entrusted to Technical Committee GEL/65/3, 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.

© The British Standards Institution 2019
Published by BSI Standards Limited 2019

ISBN 978 0 539 05638 9

ICS 35.110; 35.100.70; 25.040.40

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 September 2019.

Amendments/corrigenda issued since publication

Date	Text affected
------	---------------

English Version

Industrial communication networks - Fieldbus specifications -
Part 6-2: Application layer protocol specification - Type 2
elements
(IEC 61158-6-2:2019)

Réseaux de communication industriels - Spécifications des
bus de terrain - Partie 6-2: Spécification du protocole de la
couche application - Éléments de type 2
(IEC 61158-6-2:2019)

Industrielle Kommunikationsnetze - Feldbusse - Teil 6-2:
Protokollspezifikation des Application Layer
(Anwendungsschicht) - Typ 2-Elemente
(IEC 61158-6-2:2019)

This European Standard was approved by CENELEC on 2019-07-25. 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 CEN-CENELEC Management Centre 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 CEN-CENELEC Management Centre has the same status as the official versions.

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



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

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

The text of document 65C/948/FDIS, future edition 4 of IEC 61158-6-2, 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 approved by CENELEC as EN IEC 61158-6-2:2019.

The following dates are fixed:

- latest date by which the document has to be implemented at national level by publication of an identical national standard or by endorsement (dop) 2020-04-25
- latest date by which the national standards conflicting with the document have to be withdrawn (dow) 2022-07-25

This document supersedes EN 61158-6-2:2014 and all of its amendments and corrigenda (if any).

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

Endorsement notice

The text of the International Standard IEC 61158-6-2:2019 was approved by CENELEC as a European Standard without any modification.

In the official version, for Bibliography, the following notes have to be added for the standards indicated:

IEC 61131-3 NOTE Harmonized as EN 61131-3
IEC 61784-1:2019 NOTE Harmonized as EN IEC 61784-1:2019 (not modified)
IEC 61784-2:2019 NOTE Harmonized as EN IEC 61784-2:2019 (not modified)

Annex ZA

(normative)

Normative references to international publications with their corresponding European publications

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

NOTE 1 Where an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 61158-1	2019	Industrial communication networks Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series	-EN IEC 61158-1	2019
IEC 61158-3-2	2014	Industrial communication networks Fieldbus specifications - Part 3-2: Data- link layer service definition - Type 2 elements	-EN 61158-3-2	2014
IEC 61158-3-2019 2:2014/Amd 1			+ A1	2019
IEC 61158-4-2	2019	Industrial communication networks Fieldbus specifications - Part 4-2: Data-link layer protocol specification - Type 2 elements	-EN IEC 61158-4-2	2019
IEC 61158-5-2	2019	Industrial communication networks Fieldbus specifications - Part 5-2: Application layer service definition - Type 2 elements	-EN IEC 61158-5-2	2019
IEC 61588	2009	Precision clock synchronization protocol for- networked measurement and control systems		-
IEC 61784-3-2	-	Industrial communication networks -- Profiles - Part 3-2: Functional safety fieldbuses - Additional specifications for CPF 2	--	-
IEC 61800-7-202	-	Adjustable speed electrical power drive systems - Part 7-202: Generic interface and use of profiles for power drive systems - Profile type 2 specification	EN 61800-7-202	-
IEC 62026-3	2014	Low-voltage switchgear and controlgear -- Controller-device interfaces (CDIs) - Part 3: DeviceNet	--	-
ISO 639-2	-	Codes for the representation of names of- languages - Part-2: Alpha-3 code		-
ISO 11898	1993	Road vehicles - Interchange of digital- information - Controller area network (CAN) for high-speed communication		-

ISO/IEC 7498-1	-	Information technology - Open Systems-Interconnection - Basic reference model: The basic model	-
ISO/IEC 8824-1	-	Information technology - Abstract Syntax-Notation One (ASN.1): Specification of basic notation	-
ISO/IEC 8825-1	-	Information technology - ASN.1 encoding-rules: Specification of Basic Encoding Rules (BER), Canonical Encoding Rules (CER) and Distinguished Encoding Rules (DER)	-
ISO/IEC 9545	-	Information technology - Open Systems-Interconnection - Application layer structure	-
ISO/IEC 10646	-	Information technology - Universal Coded-Character Set (UCS)	-
ISO/IEC 10731	-	Information technology - Open Systems-Interconnection - Basic Reference Model - Conventions for the definition of OSI services	-
IEEE Std 802.1D	2004	IEEE standard for local and metropolitan area networks – Media Access Control (MAC) bridges	-
IEEE Std 802.1Q	2005	IEEE standard for local and metropolitan area networks – Virtual bridged local area networks	-
IEEE Std 802.3	2015	IEEE Standard for Ethernet	-
IETF RFC 1035	-	Domain Names - Implementation and-Specification	-
IETF RFC 1112	-	Host Extensions for IP Multicasting	-
IETF RFC 1117	-	Internet numbers	-
IETF RFC 1122	-	Requirements for Internet Hosts --Communication Layers	-
IETF RFC 1759	-	Printer MIB	-
IETF RFC 2236	-	Internet Group Management Protocol,-Version 2	-
IETF RFC 2474	-	Definition of the Differentiated Services-Field (DS Field) in the IPv4 and IPv6 Headers	-
IETF RFC 2475	-	An Architecture for Differentiated Services	-
IETF RFC 2597	-	Assured Forwarding PHB Group	-
IETF RFC 2873	-	TCP Processing of the IPv4 Precedence-Field	-
IETF RFC 3140	-	Per Hop Behavior Identification Codes	-
IETF RFC 3246	-	An Expedited Forwarding PHB (Per-Hop-Behavior)	-
IETF RFC 3376	-	Internet Group Management Protocol,-Version 3	-
IETF RFC 4594	-	Configuration Guidelines for DiffServ-Service Classes	-
IETF RFC 791	-	Internet protocol darpa internet program-protocol specification	-
ISO/IEC/IEEE 8802--3	-	Standard for Ethernet	-

CONTENTS

FOREWORD.....	14
INTRODUCTION.....	16
1 Scope.....	17
1.1 General.....	17
1.2 Specifications	17
1.3 Conformance	18
2 Normative references	18
3 Terms, definitions, symbols, abbreviated terms and conventions	20
3.1 Terms and definitions from other ISO/IEC standards.....	20
3.1.1 Terms and definitions from ISO/IEC 7498-1	20
3.1.2 Terms and definitions from ISO/IEC 9545	21
3.1.3 Terms and definitions from ISO/IEC 8824-1	21
3.1.4 Terms and definitions from ISO/IEC 8825-1	22
3.2 Terms and definitions from IEC 61158-5-2	22
3.3 Additional terms and definitions	22
3.4 Abbreviated terms and symbols	29
3.5 Conventions.....	29
3.5.1 General concept	29
3.5.2 Attribute specification	30
3.5.3 Common services	30
3.5.4 State machine conventions	34
4 Abstract syntax.....	35
4.1 FAL PDU abstract syntax	35
4.1.1 General	35
4.1.2 PDU structure.....	36
4.1.3 UCMM_PDUs	38
4.1.4 Transport_Headers.....	40
4.1.5 CM_PDUs.....	43
4.1.6 CM PDU components	57
4.1.7 MR headers	66
4.1.8 OM_Service_PDU.....	67
4.1.9 Message and connection paths.....	122
4.1.10 Class, attribute and service codes	137
4.1.11 Error codes.....	143
4.2 Data abstract syntax specification.....	157
4.2.1 Transport format specification.....	157
4.2.2 Abstract syntax notation	157
4.2.3 Control network data specification	158
4.2.4 Data type specification / dictionaries.....	160
4.3 Encapsulation abstract syntax.....	162
4.3.1 Encapsulation protocol	162
4.3.2 Command descriptions	165
4.3.3 Common packet format.....	177
5 Transfer syntax.....	181
5.1 Compact encoding	181
5.1.1 Encoding rules.....	181

5.1.2	Encoding constraints	182
5.1.3	Examples.....	182
5.2	Data type reporting	188
5.2.1	Object data representation.....	188
5.2.2	Elementary data type reporting	189
5.2.3	Constructed data type reporting.....	190
6	Structure of FAL protocol state machines	196
7	AP-Context state machine	196
7.1	Overview.....	196
7.2	Connection object state machine	196
7.2.1	I/O Connection instance behavior	196
7.2.2	Bridged Connection instance behavior	200
7.2.3	Explicit Messaging Connection instance behavior	201
8	FAL service protocol machine (FSPM).....	204
8.1	General.....	204
8.2	Primitive definitions.....	204
8.3	Parameters of primitives	209
8.4	FSPM state machines	209
9	Application relationship protocol machines (ARPMs)	210
9.1	General.....	210
9.2	Connection-less ARPM (UCMM)	210
9.2.1	General	210
9.2.2	Primitive definitions	210
9.2.3	Parameters of primitives	211
9.2.4	UCMM state machines.....	212
9.2.5	Examples of UCMM sequences	217
9.2.6	Management UCMM	219
9.3	Connection-oriented ARPMs (transports)	220
9.3.1	Transport PDU buffer.....	220
9.3.2	Transport classes	220
9.3.3	Common primitive definitions.....	221
9.3.4	Parameters of common primitives	222
9.3.5	Transport state machines – class 0.....	222
9.3.6	Transport state machines – class 1.....	226
9.3.7	Transport state machines – class 2.....	231
9.3.8	Transport state machines – class 3.....	239
9.3.9	Transport state machines – classes 4, 5, 6	249
9.3.10	Transport state machines – class 4.....	249
9.3.11	Transport state machines – class 5.....	249
9.3.12	Transport state machines – class 6.....	249
10	DLL mapping protocol machine 1 (DMPM 1)	249
10.1	General.....	249
10.2	Link producer.....	250
10.3	Link consumer	250
10.4	Primitive definitions.....	250
10.4.1	Primitives exchanged between DMPM and ARPM.....	250
10.4.2	Parameters of ARPM/DMPM primitives	250
10.4.3	Primitives exchanged between data-link layer and DMPM.....	251

10.4.4	Parameters of DMPM/Data-link Layer primitives	251
10.4.5	Network connection ID	252
10.5	DMPM state machine	253
10.5.1	DMPM states	253
10.5.2	Functions used by DMPM	254
10.6	Data-link Layer service selection	254
11	DLL mapping protocol machine 2 (DMPM 2)	254
11.1	General	254
11.2	Mapping of UCMM PDUs	255
11.2.1	General	255
11.2.2	Common requirements for Connection Manager PDU's	256
11.2.3	Forward_open PDU for class 2 and class 3 connections	258
11.2.4	Forward_open for class 0 and class 1 connections	258
11.2.5	Forward_close	262
11.3	Mapping of transport class 0 and class 1 PDUs	263
11.3.1	Class 0 and class 1 PDUs	263
11.3.2	No dependency on TCP connections	263
11.3.3	Class 0 and class 1 packet ordering	263
11.3.4	Screening incoming connected data	264
11.4	Mapping of transport class 2 and class 3 PDU's	264
11.5	Mapping of transport classes 4 to 6	265
11.6	IGMP Usage	265
11.6.1	Background (informative)	265
11.6.2	IGMP Membership Report messages	266
11.6.3	IGMP Leave Group messages	266
11.7	Quality of Service (QoS) for CP 2/2 messages	267
11.7.1	Overview	267
11.7.2	DSCP format	267
11.7.3	IEEE 802.1D/IEEE 802.Q format	268
11.7.4	Mapping CPF 2 traffic to DSCP and IEEE 802.1D	268
11.7.5	CP 2/2 usage of DSCP	269
11.7.6	CP 2/2 usage of IEEE 802.1D/IEEE 802.1Q	269
11.7.7	User considerations with IEEE 802.1D/IEEE 802.1Q	269
11.8	Management of an encapsulation session	270
11.8.1	Phases of an encapsulation session	270
11.8.2	Establishing a session	270
11.8.3	Terminating a session	270
11.8.4	Maintaining a session	270
11.8.5	TCP connection management	271
12	DLL mapping protocol machine 3 (DMPM 3)	271
	Bibliography	272
	Figure 1 – Attribute table format and terms	30
	Figure 2 – Service request/response parameter	30
	Figure 3 – Example of an STD	34
	Figure 4 – Network connection parameters	58
	Figure 5 – Priority/Tick_time bit definition	61
	Figure 6 – Member ID/EX description (WORD)	70

Figure 7 – Transport Class Trigger attribute	110
Figure 8 – CP2/3_initial_comm_characteristics attribute format	114
Figure 9 – Segment type	123
Figure 10 – Port segment	124
Figure 11 – Logical segment encoding	126
Figure 12 – Extended network segment	132
Figure 13 – Symbolic segment encoding	133
Figure 14 – Encapsulation message	162
Figure 15 – FixedLengthBitString compact encoding bit placement rules	185
Figure 16 – Example compact encoding of a SWORD FixedLengthBitString	186
Figure 17 – Example compact encoding of a WORD FixedLengthBitString	186
Figure 18 – Example compact encoding of a DWORD FixedLengthBitString	186
Figure 19 – Example compact encoding of a LWORD FixedLengthBitString	186
Figure 20 – Example 1 of formal encoding of a structure type specification	191
Figure 21 – Example 2 of formal encoding of a structure type specification	191
Figure 22 – Example 3 of formal encoding of a handle structure type specification	192
Figure 23 – Example 4 of formal encoding of a handle structure type specification	192
Figure 24 – Example 5 of abbreviated encoding of a structure type specification	193
Figure 25 – Example 1 of formal encoding of an array type specification	194
Figure 26 – Example 2 of formal encoding of an array type specification	194
Figure 27 – Example 1 of abbreviated encoding of an array type specification	195
Figure 28 – Example 2 of abbreviated encoding of an array type specification	195
Figure 29 – I/O Connection object state transition diagram	196
Figure 30 – Bridged Connection object state transition diagram	200
Figure 31 – Explicit Messaging Connection object state transition diagram	202
Figure 32 – State transition diagram of UCMM client9	212
Figure 33 – State transition diagram of high-end UCMM server	214
Figure 34 – State transition diagram of low-end UCMM server	216
Figure 35 – Sequence diagram for a UCMM with one outstanding message	218
Figure 36 – Sequence diagram for a UCMM with multiple outstanding messages	219
Figure 37 – TPDU buffer	220
Figure 38 – Data flow diagram using a client transport class 0 and server transport class 0	223
Figure 39 – Sequence diagram of data transfer using transport class 0	223
Figure 40 – Class 0 client STD	224
Figure 41 – Class 0 server STD	225
Figure 42 – Data flow diagram using client transport class 1 and server transport class 1	226
Figure 43 – Sequence diagram of data transfer using client transport class 1 and server transport class 1	227
Figure 44 – Class 1 client STD	229
Figure 45 – Class 1 server STD	230
Figure 46 – Data flow diagram using client transport class 2 and server transport class 2	232

Figure 47 – Diagram of data transfer using client transport class 2 and server transport class 2 without returned data	233
Figure 48 – Sequence diagram of data transfer using client transport class 2 and server transport class 2 with returned data	234
Figure 49 – Class 2 client STD	235
Figure 50 – Class 2 server STD	237
Figure 51 – Data flow diagram using client transport class 3 and server transport class 3	240
Figure 52 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 without returned data.....	241
Figure 53 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 with returned data	242
Figure 54 – Class 3 client STD	244
Figure 55 – Class 3 server STD	247
Figure 56 – Data flow diagram for a link producer and consumer	249
Figure 57 – State transition diagram for a link producer	253
Figure 58 – State transition diagram for a link consumer.....	254
Figure 59 – DS field in the IP header	268
Figure 60 – IEEE 802.1Q tagged frame.....	268
Table 1 – Get_Attributes_All response service rules	31
Table 2 – Example class level object/service specific response data of Get_Attributes_All.....	31
Table 3 – Example Get_Attributes_All data array method.....	32
Table 4 – Set_Attributes_All request service rules	33
Table 5 – Example Set_Attributes_All attribute ordering method	33
Table 6 – Example Set_Attributes_All data array method	33
Table 7 – State event matrix format	35
Table 8 – Example state event matrix	35
Table 9 – UCMM_PDU header format	39
Table 10 – UCMM command codes.....	39
Table 11 – Transport class 0 header	40
Table 12 – Transport class 1 header	40
Table 13 – Transport class 2 header	40
Table 14 – Transport class 3 header	40
Table 15 – Real-time data header – exclusive owner	41
Table 16 – Real-time data header– redundant owner	41
Table 17 – Forward_Open request format.....	46
Table 18 – Forward_Open_Good response format	46
Table 19 – Forward_Open_Bad response format	47
Table 20 – Large_Forward_Open request format	48
Table 21 – Large_Forward_Open_Good response format	48
Table 22 – Large_Forward_Open_Bad response format.....	49
Table 23 – Forward_Close request format.....	50
Table 24 – Forward_Close_Good response format.....	50

Table 25 – Forward_Close_Bad response format	51
Table 26 – Unconnected_Send request format	51
Table 27 – Unconnected_Send_Good response format	52
Table 28 – Unconnected_Send_Bad response format	53
Table 29 – Unconnected_Send request format (modified)	54
Table 30 – Unconnected_Send_Good response format (modified)	54
Table 31 – Unconnected_Send_Bad response format (modified)	55
Table 32 – Get_Connection_Data request format	55
Table 33 – Get_Connection_Data response format	55
Table 34 – Search_Connection_Data request format	56
Table 35 – Get_Connection_Owner request format	57
Table 36 – Get_Connection_Owner response format	57
Table 37 – Time-out multiplier	60
Table 38 – Tick time units	61
Table 39 – Encoded application path ordering	65
Table 40 – Transport class, trigger and Is_Server format	66
Table 41 – MR_Request_Header format	66
Table 42 – MR_Response_Header format	67
Table 43 – Structure of Get_Attributes_All_ResponsePDU body	67
Table 44 – Structure of Set_Attributes_All_RequestPDU body	68
Table 45 – Structure of Get_Attribute_List_RequestPDU body	68
Table 46 – Structure of Get_Attribute_List_ResponsePDU body	68
Table 47 – Structure of Set_Attribute_List_RequestPDU body	68
Table 48 – Structure of Set_Attribute_List_ResponsePDU body	69
Table 49 – Structure of Reset_RequestPDU body	69
Table 50 – Structure of Reset_ResponsePDU body	69
Table 51 – Structure of Start_RequestPDU body	69
Table 52 – Structure of Start_ResponsePDU body	69
Table 53 – Structure of Stop_RequestPDU body	70
Table 54 – Structure of Stop_ResponsePDU body	70
Table 55 – Structure of Create_RequestPDU body	70
Table 56 – Structure of Create_ResponsePDU body	70
Table 57 – Structure of Delete_RequestPDU body	70
Table 58 – Structure of Delete_ResponsePDU body	70
Table 59 – Structure of Get_Attribute_Single_ResponsePDU body	71
Table 60 – Structure of Set_Attribute_Single_RequestPDU body	71
Table 61 – Structure of Set_Attribute_Single_ResponsePDU body	71
Table 62 – Structure of Find_Next_Object_Instance_RequestPDU body	71
Table 63 – Structure of Find_Next_Object_Instance_ResponsePDU body	72
Table 64 – Structure of Apply_Attributes_RequestPDU body	72
Table 65 – Structure of Apply_Attributes_ResponsePDU body	72
Table 66 – Structure of Save_RequestPDU body	72
Table 67 – Structure of Save_ResponsePDU body	72

Table 68 – Structure of Restore_RequestPDU body	73
Table 69 – Structure of Restore_ResponsePDU body	73
Table 70 – Structure of Get_Member_ResponsePDU body	73
Table 71 – Structure of Set_Member_RequestPDU body	73
Table 72 – Structure of Set_Member_ResponsePDU body	73
Table 73 – Structure of Insert_Member_RequestPDU body	74
Table 74 – Structure of Insert_Member_ResponsePDU body	74
Table 75 – Structure of Remove_Member_ResponsePDU body	74
Table 76 – Common structure of _Member_RequestPDU body (basic format)	75
Table 77 – Common structure of _Member_ResponsePDU body (basic format)	75
Table 78 – Common structure of _Member_RequestPDU body (extended format)	76
Table 79 – Common structure of _Member_ResponsePDU body (extended format)	76
Table 80 – Extended Protocol ID	76
Table 81 – Structure of _Member_RequestPDU body (Multiple Sequential Members)	77
Table 82 – Structure of _Member_ResponsePDU body (Multiple Sequential Members)	77
Table 83 – Structure of _Member_RequestPDU body (International String Selection)	78
Table 84 – Structure of _Member_ResponsePDU body (International String Selection)	78
Table 85 – Structure of Group_Sync_RequestPDU body	78
Table 86 – Structure of Group_Sync_ResponsePDU body	78
Table 87 – Structure of Multiple_Service_Packet_RequestPDU body	79
Table 88 – Structure of Multiple_Service_Packet_ResponsePDU body	79
Table 89 – Identity object class attributes	80
Table 90 – Identity object instance attributes	80
Table 91 – Identity object bit definitions for status instance attribute	81
Table 92 – Default values for extended device status field (bits 4 to 7) of status instance attribute	82
Table 93 – Identity object bit definitions for protection mode instance attribute	82
Table 94 – Class level object/service specific response data of Get_Attributes_All	82
Table 95 – Instance level object/service specific response data of Get_Attributes_All	83
Table 96 – Object-specific request parameter for Reset	83
Table 97 – Reset service parameter values	84
Table 98 – Communication link attributes that shall be preserved	84
Table 99 – Structure of Flash_LEDs_RequestPDU body	84
Table 100 – Message Router object class attributes	85
Table 101 – Message Router object instance attributes	85
Table 102 – Class level object/service specific response data of Get_Attributes_All	85
Table 103 – Instance level object/service specific response data of Get_Attributes_All	86
Table 104 – Structure of Symbolic_Translation_RequestPDU body	86
Table 105 – Structure of Symbolic_Translation_ResponsePDU body	86
Table 106 – Object specific status for Symbolic_Translation service	86
Table 107 – Assembly object class attributes	87
Table 108 – Assembly object instance attributes	87
Table 109 – Assembly Instance ID ranges	88

Table 110 – Acknowledge Handler object class attributes	88
Table 111 – Acknowledge Handler object instance attributes	89
Table 112 – Structure of Add_AckData_Path_RequestPDU body	89
Table 113 – Structure of Remove_AckData_Path_RequestPDU body	89
Table 114 – Time Sync object class attributes	90
Table 115 – Time Sync object instance attributes	90
Table 116 – ClockIdentity encoding for different network implementations	94
Table 117 – ClockClass values	94
Table 118 – TimeAccuracy values	94
Table 119 – TimePropertyFlags bit values	95
Table 120 – TimeSource values	95
Table 121 – Types of Clock	96
Table 122 – Network protocol to PortPhysicalAddressInfo mapping	96
Table 123 – Parameter object class attributes	97
Table 124 – Parameter Class Descriptor bit values	97
Table 125 – Parameter object instance attributes	98
Table 126 – Semantics of Descriptor Instance attribute	99
Table 127 – Descriptor Scaling bits usage	99
Table 128 – Minimum and Maximum Value semantics	100
Table 129 – Scaling Formula attributes	101
Table 130 – Scaling links	101
Table 131 – Class level object/service specific response data of Get_Attributes_All	102
Table 132 – Instance level object/service specific response data of Get_Attributes_All (Parameter object stub)	102
Table 133 – Instance level object/service specific response data of Get_Attributes_All (full Parameter object)	103
Table 134 – Structure of Get_Enum_String_RequestPDU body	104
Table 135 – Structure of Get_Enum_String_ResponsePDU body	104
Table 136 – Enumerated strings Type versus Parameter data type	104
Table 137 – Connection Manager object class attributes	105
Table 138 – Connection Manager object instance attributes	105
Table 139 – Class level object/service specific response data of Get_Attributes_All	106
Table 140 – Instance level object/service specific response data of Get_Attributes_All	106
Table 141 – Instance level object/service specific request data of Set_Attributes_All	107
Table 142 – Connection object class attributes	107
Table 143 – Connection object instance attributes	108
Table 144 – Values assigned to the state attribute	109
Table 145 – Values assigned to the instance_type attribute	110
Table 146 – Possible values within Direction Bit	111
Table 147 – Possible values within Production Trigger Bits	111
Table 148 – Possible values within Transport Class Bits	112
Table 149 – TransportClass_Trigger attribute values summary	112
Table 150 – Transport Class 0 client behavior summary	113
Table 151 – Transport Class 1, 2 and 3 client behavior summary	113

Table 152 – Values defined for the CP2/3_produced_connection_id attribute	114
Table 153 – Values defined for the CP2/3_consumed_connection_id attribute	114
Table 154 – Values for the Initial Production Characteristics nibble	115
Table 155 – Values for the Initial Consumption Characteristics nibble.....	116
Table 156 – Values for the watchdog_timeout_action.....	119
Table 157 – Structure of Connection_Bind_RequestPDU body.....	121
Table 158 – Object specific status for Connection_Bind service	121
Table 159 – Structure of Producing_Application_Lookup_RequestPDU body	121
Table 160 – Structure of Producing_Application_Lookup_ResponsePDU body.....	122
Table 161 – Producing_Application_Lookup Service status codes.....	122
Table 162 – Possible port segment examples	124
Table 163 – TCP/IP link address examples	125
Table 164 – Extended Logical Type	126
Table 165 – Electronic key segment format.....	128
Table 166 – Logical segments examples	129
Table 167 – Network segments	130
Table 168 – Extended subtype definitions	132
Table 169 – Symbolic segment examples	133
Table 170 – Data segment	134
Table 171 – ANSI_Extended_Symbol segment	134
Table 172 – Addressing categories	137
Table 173 – Class code ID ranges	137
Table 174 – Attribute ID ranges	138
Table 175 – Service code ranges	138
Table 176 – Class codes	139
Table 177 – Reserved class attributes for all object class definitions	140
Table 178 – Common services list	140
Table 179 – Identity object specific services list.....	141
Table 180 – Message Router object specific services list.....	141
Table 181 – Acknowledge Handler object specific services list.....	141
Table 182 – Parameter object specific services list	141
Table 183 – Services specific to Connection Manager	142
Table 184 – Services specific to Connection object.....	142
Table 185 – Device type numbering	143
Table 186 – Connection Manager service request error codes	144
Table 187 – General status codes.....	153
Table 188 – Extended status code for a general status of "Key Failure in path.....	155
Table 189 – Identity object status codes	156
Table 190 – Encapsulation header	163
Table 191 – Encapsulation command codes	163
Table 192 – Encapsulation status codes	165
Table 193 – Nop request encapsulation header	166
Table 194 – RegisterSession request encapsulation header	166

Table 195 – RegisterSession request data portion	167
Table 196 – RegisterSession reply encapsulation header	167
Table 197 – RegisterSession reply data portion (successful)	168
Table 198 – UnRegisterSession request encapsulation header	168
Table 199 – ListServices request encapsulation header	169
Table 200 – ListServices reply encapsulation header	169
Table 201 – ListServices reply data portion (successful)	170
Table 202 – Communications capability flags	170
Table 203 – ListIdentity request encapsulation header	171
Table 204 – ListIdentity reply encapsulation header	172
Table 205 – ListIdentity reply data portion (successful)	172
Table 206 – CPF 2 identity item	173
Table 207 – ListInterfaces request encapsulation header	174
Table 208 – ListInterfaces reply encapsulation header	174
Table 209 – SendRRData request encapsulation header	175
Table 210 – SendRRData request data portion	175
Table 211 – SendRRData reply encapsulation header	176
Table 212 – SendUnitData request encapsulation header	176
Table 213 – SendUnitData request data portion	176
Table 214 – Common packet format	177
Table 215 – CPF item format	177
Table 216 – Item Type ID numbers	177
Table 217 – Null address item	178
Table 218 – Connected address item	178
Table 219 – Sequenced address item	179
Table 220 – Unconnected data item	179
Table 221 – Connected data item	179
Table 222 – Sockaddr info items	180
Table 223 – Usage of CPF items	181
Table 224 – BOOLEAN encoding	182
Table 225 – Example compact encoding of a BOOL value	182
Table 226 – Encoding of SignedInteger values	182
Table 227 – Example compact encoding of a SignedInteger value	182
Table 228 – UnsignedInteger values	183
Table 229 – Example compact encoding of an UnsignedInteger	183
Table 230 – FixedLengthReal values	183
Table 231 – Example compact encoding of a REAL value	183
Table 232 – Example compact encoding of a LREAL value	183
Table 233 – FixedLengthReal values	184
Table 234 – STRING value	184
Table 235 – STRING2 value	184
Table 236 – STRINGN value	184
Table 237 – SHORT_STRING value	185

Table 238 – Example compact encoding of a STRING value	185
Table 239 – Example compact encoding of STRING2 value	185
Table 240 – SHORT_STRING type	185
Table 241 – Example compact encoding of a single dimensional ARRAY	187
Table 242 – Example compact encoding of a multi-dimensional ARRAY	187
Table 243 – Example compact encoding of a STRUCTURE	188
Table 244 – Identification codes and descriptions of elementary data types	189
Table 245 – Identification codes and descriptions of constructed data types	190
Table 246 – Formal structure encoding definition	190
Table 247 – Formal structure with handles encoding definition	191
Table 248 – Abbreviated structure encoding definition	192
Table 249 – Formal array encoding definition	193
Table 250 – Abbreviated array encoding definition	195
Table 251 – I/O Connection state event matrix	197
Table 252 – Bridged Connection state event matrix	200
Table 253 – Explicit Messaging Connection state event matrix	202
Table 254 – Primitives issued by FAL user to FSPM	205
Table 255 – Primitives issued by FAL user to FSPM	205
Table 256 – Primitives issued by FSPM to FAL user	208
Table 257 – Parameters used with primitives exchanged between FAL user and FSPM	209
Table 258 – Primitives issued by FSPM to ARPM	211
Table 259 – Primitives issued by ARPM to FSPM	211
Table 260 – Parameters used with primitives exchanged between FSPM and ARPM	211
Table 261 – UCMM client states	212
Table 262 – State event matrix of UCMM client	213
Table 263 – High-end UCMM server states	214
Table 264 – State event matrix of high-end UCMM server	215
Table 265 – Low-end UCMM server states	216
Table 266 – State event matrix of low-end UCMM server	217
Table 267 – Notification	220
Table 268 – Transport classes	221
Table 269 – Primitives issued by FSPM to ARPM	221
Table 270 – Primitives issued by ARPM to FSPM	222
Table 271 – Parameters used with primitives exchanged between FSPM and ARPM	222
Table 272 – Class 0 transport client states	224
Table 273 – Class 0 client SEM	224
Table 274 – Class 0 transport server states	225
Table 275 – Class 0 server SEM	225
Table 276 – Class 1 transport client states	228
Table 277 – Class 1 client SEM	229
Table 278 – Class 1 transport server states	230
Table 279 – Class 1 server SEM	231
Table 280 – Class 2 transport client states	235

Table 281 – Class 2 client SEM	236
Table 282 – Class 2 transport server states	237
Table 283 – Class 2 server SEM	238
Table 284 – Class 3 transport client states	243
Table 285 – Class 3 client SEM	244
Table 286 – Class 3 transport server states	246
Table 287 – Class 3 server SEM	248
Table 288 – Primitives issued by ARPM to DMPM	250
Table 289 – Primitives issued by DMPM to ARPM	250
Table 290 – Parameters used with primitives exchanged between ARPM and DMPM	251
Table 291 – Primitives exchanged between data-link layer and DMPM	251
Table 292 – Parameters used with primitives exchanged between DMPM and Data-link	251
Table 293 – Selection of connection ID	252
Table 294 – Link producer states	253
Table 295 – State event matrix of link producer	253
Table 296 – Link consumer states	254
Table 297 – State event matrix of link consumer	254
Table 298 – UCMM request	255
Table 299 – UCMM reply	256
Table 300 – Network Connection ID selection	257
Table 301 – Sockaddr Info usage	259
Table 302 – Example multicast assignments	262
Table 303 – UDP data format for class 0 and class 1	263
Table 304 – Transport class 2 and class 3 connected data	265
Table 305 – Default DSCP and IEEE 802.1D mapping	268

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**INDUSTRIAL COMMUNICATION NETWORKS –
FIELDBUS SPECIFICATIONS –****Part 6-2: Application layer protocol specification –
Type 2 elements****FOREWORD**

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

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

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

International Standard IEC 61158-6-2 has been prepared by subcommittee 65C: Industrial networks, of IEC technical committee 65: Industrial-process measurement, control and automation.

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

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

- clarifications of response PDU formats in case of failure in 4.1.5;
- clarifications of connection request priority definition and handling in 4.1.6.5 and 4.1.6.6;
- clarification of connection remaining path in 4.1.6.12;
- extensions of general syntax in 4.1.8.1;
- extensions and clarifications of Identity object PDUs in 4.1.8.2;
- updates of Assembly object PDUs in 4.1.8.4;
- extensions and clarification of Time sync object PDUs in 4.1.8.6;
- updates of Parameter object PDUs in 4.1.8.7;
- extensions of Connection Manager object PDUs in 4.1.8.8;
- clarifications of Connection object PDUs in 4.1.8.9;
- extensions and clarifications of message and connection paths in 4.1.9;
- updates of object and service class codes in 4.1.10 and error codes in 4.1.11;
- updates of data types in 4.2.4, 5.1.3 and 5.2.3;
- updates of the encapsulation abstract syntax in 4.3;
- updates to the DLL mapping protocol machine 2 in Clause 11;
- miscellaneous editorial corrections.

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

FDIS	Report on voting
65C/948/FDIS	65C/956/RVD

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

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

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

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

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

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

INTRODUCTION

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

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

INDUSTRIAL COMMUNICATION NETWORKS – FIELDBUS SPECIFICATIONS –

Part 6-2: Application layer protocol specification – Type 2 elements

1 Scope

1.1 General

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

This part of IEC 61158 provides common elements for basic time-critical and non-time-critical messaging communications between application programs in an automation environment and material specific to Type 2 fieldbus. The term “time-critical” is used to represent the presence of a time-window, within which one or more specified actions are required to be completed with some defined level of certainty. Failure to complete specified actions within the time window risks failure of the applications requesting the actions, with attendant risk to equipment, plant and possibly human life.

This International Standard specifies interactions between remote applications and defines the externally visible behavior provided by the Type 2 fieldbus application layer in terms of

- a) the formal abstract syntax defining the application layer protocol data units conveyed between communicating application entities;
- b) the transfer syntax defining encoding rules that are applied to the application layer protocol data units;
- c) the application context state machine defining the application service behavior visible between communicating application entities;
- d) the application relationship state machines defining the communication behavior visible between communicating application entities.

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

- a) define the wire-representation of the service primitives defined in IEC 61158-5-2, and
- b) define the externally visible behavior associated with their transfer.

This document specifies the protocol of the Type 2 fieldbus application layer, in conformance with the OSI Basic Reference Model (ISO/IEC 7498-1) and the OSI application layer structure (ISO/IEC 9545).

1.2 Specifications

The principal objective of this document is to specify the syntax and behavior of the application layer protocol that conveys the application layer services defined in IEC 61158-5-2.

A secondary objective is to provide migration paths from previously-existing industrial communications protocols.

1.3 Conformance

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

2 Normative references

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

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

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

IEC 61158-3-2:2014, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*
IEC 61158-3-2:2014/AMD1:2019

IEC 61158-4-2:2019, *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

IEC 61158-5-2:2019, *Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements*

IEC 61588:2009, *Precision clock synchronization protocol for networked measurement and control systems*

IEC 61784-3-2, *Industrial communication networks – Profiles – Part 3-2: Functional safety fieldbuses – Additional specifications for CPF 2*

IEC 61800-7-202, *Adjustable speed electrical power drive systems – Part 7-202: Generic interface and use of profiles for power drive systems – Profile type 2 specification*

IEC 62026-3:2014, *Low-voltage switchgear and controlgear – Controller-device interfaces (CDIs) – Part 3: DeviceNet*

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

ISO/IEC/IEEE 8802-3, *Information technology – Telecommunications and information exchange between systems – Local and metropolitan area networks – Specific requirements – Part 3: Standard for Ethernet*

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

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

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

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

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

ISO 639-2, *Codes for the representation of names of languages – Part 2: Alpha-3 code*

ISO 11898:1993¹, *Road vehicles – Interchange of digital information – Controller area network (CAN) for high-speed communication*

IEEE Std 802.1D-2004, *IEEE standard for local and metropolitan area networks – Media Access Control (MAC) bridges*, available at <<http://www.ieee.org>> [viewed 2018-09-04]

IEEE Std 802.1Q-2005², *IEEE standard for local and metropolitan area networks – Virtual bridged local area networks*, available at <<http://www.ieee.org>> [viewed 2018-09-04]

IEEE Std 802.3-2015, *IEEE Standard for Ethernet*, available at <<http://www.ieee.org>> [viewed 2018-09-04]

IETF RFC 791, *Internet Protocol*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 1035, *Domain Names – Implementation and Specification*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 1112, *Host Extensions for IP Multicasting*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 1117, *Internet Numbers*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 1122, *Requirements for Internet Hosts – Communication Layers*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 1759, *Printer MIB*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 2236, *Internet Group Management Protocol, Version 2*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 2474, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 2475, *An Architecture for Differentiated Services*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 2597, *Assured Forwarding PHB Group*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

¹ A newer edition of this document has been published, but only the cited edition applies.

² A newer edition of this document has been published, but only the cited edition applies.

IETF RFC 2873, *TCP Processing of the IPv4 Precedence Field*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 3140, *Per Hop Behavior Identification Codes*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 3246, *An Expedited Forwarding PHB (Per-Hop Behavior)*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 3376, *Internet Group Management Protocol, Version 3*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

IETF RFC 4594, *Configuration Guidelines for DiffServ Service Classes*, available at <<http://www.ietf.org>> [viewed 2018-09-04]

3 Terms, definitions, symbols, abbreviated terms and conventions

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

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

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

3.1 Terms and definitions from other ISO/IEC standards

3.1.1 Terms and definitions from ISO/IEC 7498-1

- abstract syntax
- application entity
- application process
- application protocol data unit
- application service element
- application entity invocation
- application process invocation
- application transaction
- presentation context
- real open system
- transfer syntax

3.1.2 Terms and definitions from ISO/IEC 9545

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

3.1.3 Terms and definitions from ISO/IEC 8824-1

- a) object identifier
- b) type
- c) value
- d) simple type
- e) structured type
- f) component type
- g) tag
- h) Boolean type
- i) true
- j) false
- k) integer type
- l) bitstring type
- m) octetstring type
- n) null type
- o) sequence type
- p) sequence of type
- q) choice type
- r) tagged type
- s) any type
- t) module
- u) production

3.1.4 Terms and definitions from ISO/IEC 8825-1

- a) encoding (of a data value)
- b) data value
- c) identifier octets (the singular form is used in this document)
- d) length octet(s) (both singular and plural forms are used in this document)
- e) contents octets

3.2 Terms and definitions from IEC 61158-5-2

- a) application relationship
- b) client
- c) peer
- d) server

3.3 Additional terms and definitions

3.3.1

allocate

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

3.3.2

application

function or data structure for which data is consumed or produced

3.3.3

application objects

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

3.3.4

attribute

description of an externally visible characteristic or feature of an object

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

3.3.5

behavior

indication of how an object responds to particular events

3.3.6

Best Master Clock Algorithm

BMCA

algorithm performed by each node to determine the clock that will become the master clock on a subnet and the grandmaster clock for the domain

Note 1 to entry: The algorithm primarily compares priority1, clock quality, priority2, and source identity to determine the best master among available candidates.

3.3.7**boundary clock**

clock that has multiple Precision Time Protocol (PTP) ports in a domain and maintains the timescale used in the domain

Note 1 to entry: It may serve as the source of time, i.e., be a master clock, and may synchronize to another clock, i.e., be a slave clock.

[SOURCE: IEC 61588:2009, 3.1.3, modified – second sentence changed to a Note]

3.3.8**called**

service user or a service provider that receives an indication primitive or a request APDU

3.3.9**calling**

service user or a service provider that initiates a request primitive or a request APDU

3.3.10**class**

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

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

3.3.11**class attribute**

attribute that is shared by all objects within the same class

3.3.12**class code**

unique identifier assigned to each object class

3.3.13**class specific service**

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

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

3.3.14**client**

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

3.3.15**clock**

node participating in the Precision Time Protocol (PTP) that is capable of providing a measurement of the passage of time since a defined epoch

Note 1 to entry: There are three types of clocks in IEC 61588:2009, boundary, transparent and ordinary clocks.

[SOURCE: IEC 61588:2009, 3.1.4, modified – different Note]

3.3.16**communication objects**

components that manage and provide a run time exchange of messages across the network

EXAMPLES Connection Manager object, Unconnected Message Manager (UCMM) object, and Message Router object.

3.3.17

connection

logical binding between application objects that may be within the same or different devices

Note 1 to entry: Connections may be either point-to-point or multipoint.

3.3.18

connection ID

CID

identifier assigned to a transmission that is associated with a particular connection between producers and consumers, providing a name for a specific piece of application information

3.3.19

connection path

octet stream that defines the application object to which a connection instance applies

3.3.20

connection point

buffer which is represented as a subinstance of an Assembly object

3.3.21

consume

act of receiving data from a producer

3.3.22

consumer

node or sink that is receiving data from a producer

3.3.23

consuming application

application that consumes data

3.3.24

cyclic

repetitive in a regular manner

3.3.25

device

physical hardware connected to the link

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

3.3.26

device profile

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

3.3.27

domain

logical grouping of clocks that synchronize to each other using the protocol, but that are not necessarily synchronized to clocks in another domain

[SOURCE: IEC 61588:2009, 3.1.7]

3.3.28

end node

producing or consuming node

3.3.29**end point**

one of the communicating entities involved in a connection

3.3.30**epoch**

origin of a time scale

[SOURCE: IEC 61588:2009, 3.1.9]

3.3.31**error**

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

3.3.32**frame**

denigrated synonym for DLPDU

3.3.33**grandmaster clock**

within a domain, clock that is the ultimate source of time for clock synchronization using the PTP protocol

[SOURCE: IEC 61588:2009, 3.1.13]

3.3.34**instance**

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

EXAMPLE California is an instance of the object class state.

Note 1 to entry: The terms object, instance, and object instance are used to refer to a specific instance.

3.3.35**instance attribute**

attribute whose value is unique to an object instance and whose definition is shared by all instances of an object

3.3.36**instantiated**

object that has been created in a device

3.3.37**interoperability**

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

3.3.38**Keeper**

object responsible for distributing link configuration data to all nodes on the link

3.3.39**little endian**

model of memory organization which stores the least significant octet at the lowest address, or for transfer, which transfers the lowest order octet first

Note 1 to entry: Native Type 2 data types are sent in little endian order.

3.3.40

Lpacket

Link packet

piece of application information that contains a size, control octet, tag, and link data

Note 1 to entry: Peer data-link layers use Lpackets to send and receive service data units from higher layers in the OSI stack.

3.3.41

management information

network accessible information that supports managing the operation of the fieldbus system, including the application layer

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

3.3.42

master clock

in the context of a single Precision Time Protocol (PTP) communication path, clock that is the source of time to which all other clocks on that path synchronize

[SOURCE: IEC 61588:2009, 3.1.17]

3.3.43

member

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

3.3.44

Message Router

object within a node that distributes messaging requests to appropriate application objects

3.3.45

multipoint connection

connection from one node to many

Note 1 to entry: Multipoint connections allow messages from a single producer to be received by many consumer nodes.

3.3.46

network

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

3.3.47

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

3.3.48

object specific service

service unique to the object class which defines it

3.3.49

ordinary clock

clock that has a single Precision Time Protocol (PTP) port in a domain and maintains the timescale used in the domain

Note 1 to entry: It may serve as a source of time, i.e., be a master clock, or may synchronize to another clock, i.e., be a slave clock.

[SOURCE: IEC 61588:2009, 3.1.22, modified – second sentence changed to a Note]

3.3.50**originator**

client responsible for establishing a connection path to the target

3.3.51**parent clock**

master clock to which a clock is synchronized

[SOURCE: IEC 61588:2009, 3.1.23]

3.3.52**point-to-point connection**

connection that exists between exactly two application objects

3.3.53**Precision Time Protocol****PTP**

protocol defined by IEC 61588:2009

Note 1 to entry: As an adjective, it indicates that the modified noun is specified in or interpreted in the context of IEC 61588:2009.

[SOURCE: IEC 61588:2009, 3.1.28, modified – second sentence changed to a Note]

3.3.54**produce**

act of sending data to be received by a consumer

3.3.55**producer**

node that is responsible for sending data

3.3.56**PTP message**

one of the message types defined in IEC 61588:2009

[SOURCE: IEC 61588:2009, 3.1.33]

3.3.57**PTP port**

logical access point of a clock for PTP communications to the communications network

[SOURCE: IEC 61588:2009, 3.1.35]

3.3.58**receiving**

service user that receives a confirmed primitive or an unconfirmed primitive, or a service provider that receives a confirmed APDU or an unconfirmed APDU

3.3.59**resource**

processing or information capability of a subsystem

3.3.60 sending

service user that sends a confirmed primitive or an unconfirmed primitive, or a service provider that sends a confirmed APDU or an unconfirmed APDU

3.3.61 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

3.3.62 service

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

3.3.63 synchronized clocks

(to a specified uncertainty) two clocks which have the same epoch and for which measurements of the time of a single event at an arbitrary time differ by no more than the specified uncertainty

[SOURCE: IEC 61588:2009, 3.1.40, modified – reworded]

3.3.64 system time

absolute time value as defined by CPF2 time synchronization in the context of a distributed time system where all devices have a local clock that is synchronized with a common master clock

Note 1 to entry: In the context of CPF2, System Time is a 64-bit integer value in units of nanoseconds with a value of 0 corresponding to the date 1970-01-01.

3.3.65 target

end-node to which a connection is established

3.3.66 temporary node

transient node

3.3.67 transaction id

field within a UCMM header that matches a response with the associated request

3.3.68 transparent clock

device that measures the time taken for a Precision Time Protocol (PTP) event message to transit the device and provides this information to clocks receiving this PTP event message

[SOURCE: IEC 61588:2009, 3.1.46, modified – truncated]

3.3.69 unconnected message manager UCMM

component within a node that transmits and receives unconnected explicit messages and sends them directly to the Message Router object

3.3.70**unconnected service**

messaging service which does not rely on the set up of a connection between devices before allowing information exchanges

3.3.71**vendor ID**

identification of each product manufacturer/vendor by a unique number

Note 1 to entry: Vendor IDs are assigned by ODVA, Inc.

3.4 Abbreviated terms and symbols

ASCII	American Standard Code for Information Interchange
CID	connection ID
DLL	data-link layer
DSCP	DiffServ Codepoint
IGMP	Internet Group Management Protocol (see IETF RFC 1112, IETF RFC 2236)
IP	Internet Protocol (see IETF RFC 791)
IPv4	Internet Protocol version 4 (see IETF RFC 791)
IPv6	Internet Protocol version 6 (see IETF RFC 791)
OSI	open systems interconnection (see ISO/IEC 7498-1)
PDU	protocol data unit
PHB	per-hop behavior
PTP	Precision Time Protocol (see IEC 61588:2009)
QoS	quality of service
Rcv	receive
Rx	receive
SDU	service data unit
SEM	state event matrix
STD	state transition diagram, used to describe object behavior
TCP	Terminal Control Protocol (see IETF RFC 793)
ToS	type of service
TPDU	transport protocol data unit
Tx	transmit
UDP	User Datagram Protocol (see IETF RFC 768)
Xmit	transmit
CM_API	actual packet interval
O2T or O⇒T	originator to target (connection parameters)
CM_RPI	requested packet interval
TUI	table unique identifier
T2O or T⇒O	target to originator (connection parameters)

3.5 Conventions**3.5.1 General concept**

The FAL is defined as a set of object-oriented ASEs. Each ASE is specified in a separate subclause. Each ASE specification is composed of three parts: its class definitions, its services, and its protocol specification. The first two are contained in IEC 61158-5-2. The protocol specification for each of the ASEs is defined in this document.

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 61158-5-2. The service specification defines the services that are provided by the ASE.

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

Bold font is used in this document to highlight parameter names or important requirement elements from surrounding text. Italic font is also used in Tables and Notes for that purpose for better visibility.

3.5.2 Attribute specification

Attributes are defined in an Attribute Table using the format and terms defined in Figure 1.

Attribute ID	Name	Data type	Semantics of values
--------------	------	-----------	---------------------

Figure 1 – Attribute table format and terms

The **Attribute ID** shall be a unique integer identification value assigned to an attribute. The valid ranges for Attribute IDs shall be as specified in 4.1.10.1.3. The Attribute ID shall identify the particular attribute being accessed.

Name shall refer to the name assigned to the attribute. An attribute may contain sub-elements, which may also have names, as is the case with the **STRUCT** of data type. The attribute name shall be the name which appears first, or at the top row in the name column (the row that contains the Attribute ID).

Every attribute shall be assigned a **Data type** which shall be either an elementary or a derived data type. The data types specified for the defined attributes shall be used in all implementations.

NOTE The elementary data types are defined in IEC 61158-5-2.

Semantics of values shall specify the meaning of the value(s) of the attribute. If this information needs more room than can fit in the table it will immediately follow the Class Attribute table. Included in the Class Attribute table will be an appropriate reference to this information.

If a Class Attribute is optional, then a default value or a special case processing method shall be defined such that the Client (Requester) can process the error message that occurs when accessing those objects that choose not to implement the class attribute.

3.5.3 Common services

3.5.3.1 Service_PDU definitions

Each service has unique parameters for request and response. The service request and response parameters shall be defined using service Request/Response parameter tables as defined in Figure 2.

Name	Type	Semantics of values
------	------	---------------------

Figure 2 – Service request/response parameter

Name shall refer to the name given to the service request/response parameter.

Type shall specify the data type of the service request/response parameter.

Semantics of values shall specify the meaning of the values of the service request/response parameter, e.g. “the value is counts of microseconds.”

3.5.3.2 Get_Attributes_All response

3.5.3.2.1 General definition

When the Get_Attributes_All common service is included in the list of supported System/Object Management services for an object class, then the **Get_Attributes_All** response shall be detailed for this class: the sequence or order of the data returned in the Service_ResponsePDU shall be specified. There are three ways in which the Get_Attributes_All Service_ResponsePDU may be specified:

- list the ordering of the attributes in the response message;
- specify the actual data array of the response message;
- combine the above two methods such that the actual data array also contains the attribute numbers.

Whichever method is used, the rules specified in Table 1 shall be adhered to when specifying the Get_Attributes_All Service_ResponsePDU of an object class for both the Class Attributes and the Instance Attributes.

Table 1 – Get_Attributes_All response service rules

Rule number	Rule
1	<p>If the definition of the Get_Attributes_All response includes optional attributes, then default values shall be specified in the response description. Optional attributes at the end of the list that are not implemented may be omitted in the response data. Optional attributes in the middle of the list that are not implemented shall be included in the response data, and set to specified default values.</p> <p>If any of the following optional attributes are included in an object specification, but not supported in the implementation of the object, then the following shall be adhered to:</p> <ul style="list-style-type: none"> – if the class attribute “Optional attribute list” is not supported, the default value of zero shall be inserted into the response array and no optional attribute numbers shall follow; – if the class attribute “optional service list” is not supported, the default value of zero shall be inserted into the response array and no optional service numbers shall follow.
2	If new attributes are added to an existing object, those attributes shall be added to the end of the response attribute list or data array to ensure compatibility with different object revisions.
3	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
4	The Get_Attributes_All response for objects specified in this document shall include only the open attributes; it shall not include any vendor specific attributes.

Table 2 is an example of the attribute ordering method of specifying the service data portion of a Get_Attributes_All response for class level attributes of an object which supports **optional gettable** class attributes 1, 2, 3 and 4.

Table 2 – Example class level object/service specific response data of Get_Attributes_All

Class attribute ID	Attribute name and default value
1	Revision, default = 0x0001
2	Max Instance, default = 0x0000
3	Number of Instances, default = 0x0000
4	Attribute list, number of attributes, default = 0x0000

Table 3 is an example of the data array method of specifying the service data portion of a Get_Attributes_All response for class level attributes of an object which supports **optional gettable** class attributes 1, 2, 3 and 4.

Table 3 – Example Get_Attributes_All data array method

Octet	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Revision (low octet) Default = 1							
1	Revision (high octet) Default = 0							
2	Max Instance (low octet) Default = 0							
3	Max Instance (high octet) Default = 0							
4	Number of Instances (low octet) Default = 0							
5	Number of Instances (high octet) Default = 0							
6	Optional Attribute List: number of attributes (low octet) Default = 0							
7	Optional Attribute List: number of attributes (high octet) Default = 0							
8	Optional Attribute List: optional attribute #1 (low octet)							
9	Optional Attribute List: optional attribute #1 (high octet)							
2n + 6	Optional Attribute List: optional attribute #n (low octet)							
2n + 7	Optional Attribute List: optional attribute #n (high octet)							

3.5.3.2.2 Revisions

The defined Get_Attributes_All response for an object may increase in size with each revision of the object; however, to insure interoperability, the format of the first part of the response shall remain parseable by a client of the older revision of the object. Clients (Requesters) need not make use of this compatibility requirement.

NOTE The Revision class attribute is not the revision of an implementation (which is reflected in the Identity object, Minor/Major revision status bits), but the revision of the class definition.

3.5.3.3 Set_Attributes_All request

3.5.3.3.1 General definition

When the Set_Attributes_All common service is included in the list of supported common services, then the format of the **Set_Attributes_All request** shall be included in the object specification. The sequence or order of the data supplied in the Service Data portion of the request shall be specified. There are three ways in which the Set_Attributes_All request may be specified:

- list the order of the attributes in the request message;
- specify the actual data array of the request message;
- combine the above two methods such that the actual data array also contains the attribute numbers.

Whichever method is used, the rules specified in Table 4 shall be adhered to when specifying an object's Set_Attributes_All request in the object specification for both the Class Attributes and the Instance Attributes.

Table 4 – Set_Attributes_All request service rules

Rule number	Rule
1	An object shall support the Set_Attributes_All service only if all settable attributes shown in the Set_Attributes_All request are implemented as settable.
2	Default values shall be specified for all optional attributes that are not implemented. If an implementation does not support an optional attribute, it shall accept the specified default value for the unsupported attribute.
3	If new settable attributes are added to an existing object, those attributes shall be added to the end of the request attribute list or data array.
4	Whichever method is used to specify the response, it shall be done in such a way as to be unambiguous, including rules to deal with variable length fields and padding.
5	The Set_Attributes_All request response for objects specified in this document shall include only the open attributes. It shall not include any vendor specific attributes.

Table 5 is an example of the attribute ordering method of specifying the service data portion of a Set_Attributes_All request for instance level attributes of an object which supports required settable instance attributes 7, 8, 9, 10, 11 and 12.

Table 5 – Example Set_Attributes_All attribute ordering method

Class Attribute ID	Attribute name
7	Output Range
8	Value Data Type
9	Fault State
10	Idle State
11	Fault Value
12	Idle Value

Table 6 is an example of the data array method of specifying the service data portion of a Set_Attributes_All request for instance level attributes of an object which supports required settable instance attributes 7, 8, 9, 10, 11 and 12.

Table 6 – Example Set_Attributes_All data array method

Octet	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
0	Output Range							
1	Value Data Type							
2	Fault State							
3	Idle State							
4	Fault Value (low octet)							
5	Fault Value (high octet)							
6	Idle Value (low octet)							
7	Idle Value (high octet)							

3.5.3.3.2 Revisions

A server processing a Set_Attributes_All request may need to process more data than is expected by the server if the client has implemented a newer revision of this object while the server an older revision. As a result, the server object may have to process a Set_Attributes_All request that contains more data because the additional attributes were not

recognized. If the server receives more data in a Set_Attributes_All request than it expects the server shall respond with a general status code equal to 0x15 (too much data).

NOTE The Revision class attribute is not the revision of an implementation (which is reflected in the Identity object, Minor/Major revision status bits), but the revision of the class definition.

3.5.4 State machine conventions

3.5.4.1 General

State changes may be triggered by events (internal or external) or service invocations. Reaction to service invocations may depend on the value(s) of the attribute(s) accessed by the service.

Behaviour of the state machine shall be defined for combinations of:

- the **event** the machine receives;
- the **state** the machine is in when it receives notification of a state-changing event.

To define behaviour in these terms, a State Transition Diagram (STD) and a State Event Matrix (SEM) are used when applicable in the state machine specification.

3.5.4.2 State Transition Diagram (STD)

An **event** is an external stimulus that may cause a state transition. An STD graphically illustrates the states of an object and includes events, service calls and changes of attributes that cause it to transition to another state. Figure 3 shows an example of an STD.

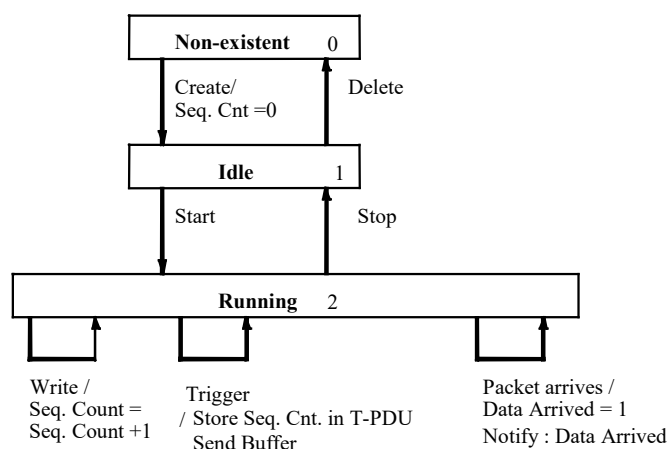


Figure 3 – Example of an STD

NOTE Event that is the primary cause of the State Transition is followed by "/" mark. Notification to upper layer is marked by "Notify:"

3.5.4.3 State Event Matrix

A **state** is the current active mode of operation of the state machine object (e.g. Running, Idle).

A state event matrix is a table that lists all possible events, services and changes in attribute values that initiate a state change, and indicates the response by the object to the event based on the state of the object when it receives notification of that event.

State event matrix format is as shown in Table 7.

Table 7 – State event matrix format

Event	State		
	State 1		State n
Event A description	Function triggered by event A in State 1 (if any) Notification to FAL user (if any) Transition to another state (if any)		Function triggered by event A in State n (if any) Notification to FAL user (if any) Transition to another state (if any)
.....			
Event X description	Function triggered by event X in State 1 (if any) Notification to FAL user (if any) Transition to another state (if any)		Function triggered by event X in State n (if any) Notification to FAL user (if any) Transition to another state (if any)
In absence of function, notification or transition, the corresponding entry shall not be made in the table.			

3.5.4.4 Example state event matrix

Table 8 shows an example of a state machine with three states:

- **Non-existent:** the object has not yet been created; objects transition to the existent state via the create service (if the object may be dynamically created) or at power-up (if the object is fixed by design/implementation);
- **Idle:** the object accepts services (e.g. Get_Attribute_Single), but does not produce or consume data onto or from the link;
- **Running:** the object is performing all its specified functions.

Table 8 – Example state event matrix

Event	State		
	Non-existent	Idle	Running
Create	Transition to Idle	Error: Object already exists.	Error: Object already exists.
Delete	Error: Object does not exist. (General Error Code 0x16)	Transition to Non-existent	Error: Object State Conflict (General Error Code 0x0C)
Start	Error: Object does not exist. (General Error Code 0x16)	Transition to Running	Error: Object State Conflict (General Error Code 0x0C)
Stop	Error: Object does not exist. (General Error Code 0x16)	Error: Object State Conflict (General Error Code 0x0C)	Transition to Idle
Get_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Validate/service the request. Return response
Set_Attribute_Single	Error: Object does not exist. (General Error Code 0x16)	Validate/service the request. Return response	Error: Object State Conflict (General Error Code 0x0C)

4 Abstract syntax

4.1 FAL PDU abstract syntax

4.1.1 General

FAL_PDU shall be either a UCMM_PDU or a Transport_PDU.

UCMM_PDU is used to convey information for connection-less services.

Transport_PDU is used to convey information for connection-oriented services.

A connected message assumes previously negotiated resources and parameters at its source, its destination(s), and any intermediate transit points. These resources are referenced by a unique connection identifier and do not need to be contained in each message, only the connection ID is needed to identify the message and refer to its related parameters, thus giving significant savings in message efficiency.

An unconnected message provides a means to communicate on the local link without previously negotiated resources at the destination so it shall carry full destination ID details, internal data descriptors and full source ID details if a reply is requested. Unconnected messages are used mainly to create connections.

The unconnected service is provided by the Unconnected Message Manager (UCMM). Messages received through the UCMM are forwarded to the Message Router (MR), which direct them to the appropriate internal object for execution. Connections are established by specific unconnected messages sent through the Connection Manager (CM) using UCMM services. Connections may be established either to the Message Router (for messaging purpose), or directly to an application object. Connection target is specified using a connection path, and may be either on the local or a remote link, through several intermediate router nodes. Once a connection is established with an application object, the UCMM, MR and CM are no longer required, since data will be exchanged directly with the connected object, based on the corresponding connection ID. Connected messages sent to the Message Router will be forwarded to the appropriate internal object for execution.

4.1.2 PDU structure

4.1.2.1 UCMM_PDU structure

UCMM_PDU is a sequence of a UCMM_Header, followed by either OM_Service (Object Management ASE) or CM_Service (Connection Manager ASE).

OM_Service shall be either OM_Request or OM_Response.

OM_Request shall consist of MR_Request_Header and a Service_RequestPDU.

Service_RequestPDU shall be one of the following:

- Get_Attribute_Single_RequestPDU
- Get_Attributes_All_RequestPDU
- Get_Attribute_List_RequestPDU
- Set_Attribute_Single_RequestPDU
- Set_Attributes_All_RequestPDU
- Set_Attribute_List_RequestPDU
- Reset_RequestPDU
- Create_RequestPDU
- Delete_RequestPDU
- Start_RequestPDU
- Stop_RequestPDU
- Find_Next_Object_Instance_RequestPDU
- NOP_RequestPDU
- Apply_Attributes_RequestPDU

- Save_RequestPDU
- Restore_RequestPDU
- Get_Member_RequestPDU
- Set_Member_RequestPDU
- Insert_Member_RequestPDU
- Remove_Member_RequestPDU
- Group_Sync_RequestPDU
- Add_AckData_Path_RequestPDU
- Remove_AckData_Path_RequestPDU
- Get_Enum_String_RequestPDU
- Symbolic_Translation_RequestPDU
- Connection_Bind_RequestPDU
- Producing_Application_Lookup_RequestPDU
- Flash_LEDs_RequestPDU
- Multiple_Service_Packet_RequestPDU

OM_Response shall consist of MR_Response_Header and a Service_ResponsePDU.

Service_ResponsePDU shall be one of the following:

- Get_Attribute_Single_ResponsePDU
- Get_Attributes_All_ResponsePDU
- Get_Attribute_List_ResponsePDU
- Set_Attribute_Single_ResponsePDU
- Set_Attributes_All_ResponsePDU
- Set_Attribute_List_ResponsePDU
- Reset_ResponsePDU
- Create_ResponsePDU
- Delete_ResponsePDU
- Start_ResponsePDU
- Stop_ResponsePDU
- Find_Next_Object_Instance_ResponsePDU
- NOP_ResponsePDU
- Apply_Attributes_ResponsePDU
- Save_ResponsePDU
- Restore_ResponsePDU
- Get_Member_ResponsePDU
- Set_Member_ResponsePDU
- Insert_Member_ResponsePDU
- Remove_Member_ResponsePDU
- Group_Sync_ResponsePDU
- Add_AckData_Path_ResponsePDU
- Remove_AckData_Path_ResponsePDU
- Get_Enum_String_ResponsePDU
- Symbolic_Translation_ResponsePDU

- Connection_Bind_ResponsePDU
- Producing_Application_Lookup_ResponsePDU
- Flash_LEDs_ResponsePDU
- Multiple_Service_Packet_ResponsePDU

CM_Service shall be either CM_Request or CM_Response.

CM_Request shall consist of MR_Request_Header and a CM_RequestPDU.

CM_RequestPDU shall be one of the following:

- Forward_Open_RequestPDU
- Forward_Close_RequestPDU
- Large_Forward_Open_RequestPDU
- Unconnected_Send_RequestPDU
- Get_Connection_Data_RequestPDU
- Search_Connection_Data_RequestPDU
- Get_Connection_Owner_RequestPDU

CM_Response consists of MR_Response_Header and a CM_ResponsePDU.

CM_ResponsePDU shall be one of the following:

- Forward_Open_ResponsePDU
- Forward_Close_ResponsePDU
- Large_Forward_Open_ResponsePDU
- Unconnected_Send_ResponsePDU
- Get_Connection_Data_ResponsePDU
- Search_Connection_Data_ResponsePDU
- Get_Connection_Owner_ResponsePDU

4.1.2.2 Transport_PDU structure

Transport_PDU is a sequence of a Transport_Header, followed by either OM_Service (Object Management ASE) or Application Data.

OM_Service is defined in 4.1.2.1 above.

Application Data comes from the source to which the connection has been made.

4.1.3 UCMM_PDUs

Subclause 4.1.3 defines the requirements for UCMM PDU's when using a Type 2 data-link layer.

All UCMM_PDUs shall be sent and received using fixed tag 0x83 or Management tag 0x88. When a device becomes powered, the UCMM shall invoke the `DLL_enable_fixed_request` service of the data-link layer to request that fixed tag 0x83 or 0x88 Lpackets be routed to the UCMM. All sending and receiving of TPDUs shall use the `DLL_fixed_request.req` and `DLL_fixed_request.ind` service primitives of the data-link layer, respectively. The packet parameter of these services shall be prefixed with the following header to form the Transport PDU before sending to the data-link layer. The format of the UCMM_PDU Header is shown in Table 9.

Table 9 – UCMM_PDU header format

Parameter name	Format
command_code	USINT
Timeout	USINT
TransactionID	UINT

The `command_code` shall specify the type of UCMM_PDU as shown in Table 10. UCMM packets received with a reserved `command_code` shall be discarded without acknowledgement.

Table 10 – UCMM command codes

Command_code	Description
0	Reserved
1	acknowledge a request
2	request with retry until acknowledged
3	response with retry until acknowledged
4	request with no acknowledge and no response
5	acknowledge a response
6	response which shall not retry (no acknowledge)
7	request with retry until response (no acknowledge)
8	Request which shall not retry and shall cause a code 6 response
9 to 255	Reserved

The timeout field shall specify the duration of the transaction in an 8-bit floating-point format. The most significant 5 bits of the field shall be an unsigned exponent that is not biased. The least significant 3 bits shall be the least significant 3 bits of a 4 bit unsigned mantissa. The most significant bit of the mantissa shall be 1 and shall not appear explicitly in the 8-bit representation. The binary point shall be positioned between the implied 1 and the rest of the mantissa. The units of the transaction duration computed using the timeout field shall be in milliseconds.

NOTE Using ANSI C precedence rules, the number of 0,125 ms ticks is $(8 \mid \text{timeout} \& 7) \ll (\text{timeout} \gg 3 \& 31)$.

The transactionID field shall be composed of two sub-fields:

- the least significant 10 bits, RECORD, shall identify a specific transaction;
- the most significant 6 bits, SEQUENCE, shall be used for duplicate detection. It shall be incremented for every unique message on this RECORD; however it shall not be incremented on a retry.

Only one message shall be outstanding for a given RECORD.

If multiple outstanding messages are required then multiple RECORDs are required.

When an I'm alive fixed tag packet is received from a node (see IEC 61158-4-2, 6.10 and 8.1), all UCMM transactions with that node shall be aborted.

4.1.4 Transport_Headers

4.1.4.1 General

Contents of Transport headers varies depending on the class of transport selected during the connection establishment.

4.1.4.2 Class 0 (null or base)

The class 0 transport header shall be an unsigned 16-bit number. This header shall be written to the TPDU buffer by the transport and shall be simply discarded by the transport from the packet coming from the consumer. The format of the Header is shown in Table 11.

Table 11 – Transport class 0 header

Parameter name	Format
don't_care	UINT

4.1.4.3 Class 1 (duplicate detection)

The class 1 transport header shall be an unsigned 16-bit sequence count. This header shall be written to the TPDU buffer by the transport and shall be read by the transport from the packet coming from the consumer. The format of the Header is shown in Table 12.

Table 12 – Transport class 1 header

Parameter name	Format
sequence_count	UINT

4.1.4.4 Class 2 (acknowledged)

Like the class 1 transport header, the class 2 transport header shall be a 16-bit sequence count. This header shall be written to the TPDU buffer by the transport and shall be read by the transport from the packet coming from the consumer. The format of the Header is shown in Table 13.

Table 13 – Transport class 2 header

Parameter name	Format
sequence_count	UINT

4.1.4.5 Class 3 (verified)

Like the class 1 transport header, the class 3 transport header shall be a 16-bit sequence count. This header shall be written to the TPDU buffer by the transport and shall be read by the transport from the packet coming from the consumer. The format of the Header is shown in Table 14.

Table 14 – Transport class 3 header

Parameter name	Format
sequence_count	UINT

4.1.4.6 Class 4 (non-blocking), class 5 (non-blocking, fragmenting) and class 6 (multipoint connection, fragmenting)

NOTE Subclause contents from the previous edition have been obsoleted.

4.1.4.7 Listen only O⇒T data format

The O⇒T connection shall use a specific heartbeat format (no 32-bit header, 0 octets of application data).

4.1.4.8 Input only O⇒T data format

The O⇒T connection shall use a specific heartbeat format (no 32-bit header, 0 octets of application data).

4.1.4.9 Exclusive owner O⇒T data format

Exclusive owner connections shall have either of two real-time transfer formats:

- 32-bit header, fixed size;
- no header, variable size.

The 32-bit header prefixed to the real-time data shall be in the form shown in Table 15.

Table 15 – Real-time data header – exclusive owner

Parameter	Format	Size (bits)
Run_idle	RUN_IDLE	1
Reserved	UDINT	31

The `run_idle` flag (bit 0) shall be set (1 = RUN) to indicate that the following data shall be sent to the target application. It shall be clear (0 = IDLE) to indicate that the idle event shall be sent to the target application. The `reserved` field (bits 1 to 31) shall be reserved and set to 0.

If the `no_header` transfer format is used, the reception of a packet with data beyond the transport header shall indicate the RUN mode. If the packet is truncated after the transport header, the target shall be sent an idle event.

4.1.4.10 Redundant owner O⇒T data format

4.1.4.10.1 General format

Redundant owner shall have a 32-bit header prefixed to the real-time data. This header shall be in the form shown in Table 16.

Table 16 – Real-time data header– redundant owner

Parameter	Format	Size (bits)
Run_idle	RUN_IDLE	1
COO	BOOLEAN	1
ROO	USINT	2
Reserved	UDINT	28

4.1.4.10.2 Run_idle flag

The run_idle flag (bit 0) shall have the same meaning as it does in an exclusive owner connection and shall be one of RUN or IDLE. The reserved field (bits 4 to 31) shall be reserved and set to 0.

4.1.4.10.3 Claim output ownership (COO) flag

The COO flag shall be set (1) when an originator application wants its connection to be the owning connection of the target application. The COO flag shall be reset (0) when an originator application does not want its connection to be the owning connection of the target application. When the owning connection resets (0) its COO flag, its sibling connections shall be checked for a set (1) COO flag. The new owner shall be any of the connections that have their COO flag set.

NOTE This results in undefined behaviour if more than one other connection has its COO flag set.

4.1.4.10.4 Ready for ownership of outputs (ROO) priority value

The ROO priority value shall be non-zero when an originator application does not want to force its connection to be the owning connection of the target application, but is ready to be the owning connection should there be no originator applications claiming to be the owning connection. The ROO priority value shall be zero when the originator application does not want to be the owning connection of the target connection and is not to be the owning connection should there be no originator application claiming to be the owning connection. The ROO priority value shall be used only when the COO flag is reset.

The value of the ROO field can range from 0 to 3. The originator applications shall each determine a unique non-zero ROO value.

4.1.4.10.5 Determining the owning connection

The originator applications shall determine among themselves which originator application has the owning connection. The owning connection shall be determined by the originator application that sets its COO flag. In situations where multiple originator applications have their COO flag set or where no originator connections have their COO flag set, the following rules shall be applied by the target transport to determine the owning connection.

- There shall be no owning connection until an originator application sends a real-time packet with the COO flag set;
- If there is only one originator application which had the COO flag set in its last real-time packet, that originator application shall have the owning connection;
- If there are multiple originator applications which had the COO flag set in its last real-time packet, the last originator application that transitioned its COO flag from reset to set shall have the owning connection.
- If the originator application with the owning connection resets its COO flag, closes its connection, or if that connection times out, and no other originator applications have their COO flags set, the originator application with the highest non-zero ROO priority value shall have the owning connection.
- If all of the originator applications have their COO flags reset and ROO priority values set to zero, there shall be no owning connection.
- When the first real-time packet containing a set COO flag is received by the target transport, the originator application that sent the real-time packet shall have the owning connection.

4.1.4.10.6 Transporting events and data to a target application

The connection related events from each of the redundant owner connections shall be combined using the following rules such that the target application sees only a single exclusive owner connection:

- Until an owning connection is initially determined, the transport shall not indicate real-time data reception to the target application;
- If an owning connection is determined, the transport shall indicate the event consistent with the owning connections real-time data to the target application. If the Run/Idle flag is reset in the real-time data for the owning connection, the transport shall indicate the idle state to the target application. If the Run/Idle flag is set in the real-time data for the owning connection, the transport shall indicate the run state and the real-time data to the target application;
- If an owning connection had been previously determined, but no originator is currently claiming or ready for ownership, the transport shall indicate idle state to the target application;
- If all the redundant connections are closed or have been timed out, the target transport shall indicate the event consistent with the last connection to have been closed or timed out to the target application.

4.1.5 CM_PDUs

4.1.5.1 General

Connection establishment and maintenance services are provided by the Connection Manager. They are

CM_Forward_Open (corresponds to Forward_Open and Large_Forward_Open PDUs)

CM_Forward_Close (corresponds to Forward_Close PDUs)

CM_Unconnected_Send (corresponds to Unconnected_Send PDUs)

CM_Get_Connection_Data (corresponds to Get_Connection_Data PDUs)

CM_Search_Connection_Data (corresponds to Search_Connection_Data PDUs)

CM_Get_Connection_Owner (corresponds to Get_Connection_Owner PDUs)

4.1.5.2 Connection Manager

The Connection Manager object is used to manage the establishment and maintenance of communication connections.

The Connection Manager objects at different nodes shall communicate using the UCMM services described in IEC 61158-5-2. The TPDUs with which peer Connection Managers communicate shall be derived from the services of the Connection Manager.

4.1.5.3 Forward_Open

4.1.5.3.1 General

The Forward_Open service is used to establish a connection with a target device. This service results in local connection establishment on each link along the path.

NOTE The processing of a single Forward_Open service can result in the creation of multiple active Connection object instances.

A Forward Open can be either a non-null Forward_Open or a null Forward_Open. A non-null Forward_Open is a Forward_Open service request for which at least one of the connection types in the O2T_ or T2O_connection_parameters field is not 00 (NULL). A null Forward_Open is a Forward_Open service request for which the connection type in both the

O2T_ and T2O_connection_parameters fields is 00 (NULL) and results in no connection being established.

A Forward_Open (both null and non-null) can be either not matching or matching. A matching Forward_Open service request received by the target device is one where the Connection Triad matches an existing connection.

The usage of each of the combinations of non-null/null and not matching/matching Forward_Open is described in 4.1.5.3.2 and summarized in the list below:

- non-null / not matching – open a connection;
- non-null / matching – error;
- null / not matching – ping a device, or configure;
- null / matching – reconfigure.

The response to the Forward_Open_Request_PDU is a Forward_Open_ResponsePDU. The Forward_Open response shall contain the originator connection serial number and owner vendor and serial number and the connection IDs (CID) necessary to complete the connection if successful and error information if the connection failed.

4.1.5.3.2 Forward_Open variants

4.1.5.3.2.1 Non null Forward_Open, not matching

The Forward_Open request sets up network, transport, and application connections. An application connection consists of a single transport connection, and one or two network connections that are in turn comprised of multiple link connections. Each port segment in the connection path uses a link connection. The Forward_Open service between two devices builds one or two link connections as specified by the network connection parameter and the requested packet intervals (CM_RPI). Since up to two network connections can be required for a single transport connection, they are differentiated by the O2T and T2O designations; O2T means originator to target, and T2O means target to originator.

The path specified in the MR_Request_Header for the Forward_Open service shall not contain any Electronic Key segment. Electronic key segments may be included in the path specified in the connection_path parameter of the Forward_Open service.

Success shall be returned when the connection requested has been established from this point forward in the path. This response also shall indicate the connection serial number and the actual packet interval of the connection. Once the successful response has been received, the connection shall be open from this point forward in the path. Targets shall wait at least 10 seconds after sending the CM_Forward_Open_Good_Response for the first packet on a connection.

4.1.5.3.2.2 Non null Forward_Open, matching

A non-null Forward_Open for which the Connection Triad matches an existing connection shall return a general status = 0x01, extended status = 0x0100 (Connection in use or Duplicate Forward_Open).

The suggested originator behaviour in the non-null Forward_Open, matching case should be to either close and re-establish the connection or wait for the connection to time-out and then establish the connection again. It shall be recognized that in the latter case, it shall take the connection 60 s (first data time-out) to time-out before the connection can be re-established.

4.1.5.3.2.3 Null Forward_Open, not matching

A null Forward_Open for which the Connection Triad does not match an existing connection's parameters can be used for the following functions.

a) Ping a device, with the following characteristics:

- the single application path "20 01 24 01" (i.e. Identity object);
- an electronic key segment may be included;
- no data segment is included;
- no connection is established.

b) Configure a device's application, with the following characteristics:

- a configuration application path and data segment shall be included in the request (the data is sent to the application specified by the path and applied);
- if the entire configuration cannot be applied, then none of the configuration shall be applied and the appropriate error code returned;
- an electronic key segment may be included;
- no connection is established.

A target device shall respond to a null Forward_Open request with one of the following errors when the target does not support the requested function ("ping a device" or "configure a device's application"):

- Null Forward_Open function not supported (general status 0x01, extended status 0x0132) (recommended).
- Invalid network connection parameter (general status 0x01, extended status 0x0108) (deprecated).

4.1.5.3.2.4 Null Forward_Open, matching

A Null Forward_Open for which the Connection Triad matches an existing connection's parameters can be used to reconfigure a target device's application. Intermediate nodes shall always forward these requests. The associated behaviours are:

- a configuration application path and data segment shall be included in the request and they are sent to the application to change the application configuration;
- if the entire configuration cannot be applied then none of the configuration shall be applied and the appropriate error code returned;
- the connection shall not be interrupted due to this request.

A target device shall respond to a null Forward_Open request with one of the following errors when the target does not support the requested function ("reconfigure a target device's application"):

- Null Forward_Open function not supported (general status 0x01, extended status 0x0132) (recommended).
- Invalid network connection parameter (general status 0x01, extended status 0x0108) (deprecated).

If the interpretation of the consumed/produced data changes as a result of the reconfigure operation, care shall be taken in the producing and consuming applications. There is no specified mechanism to coordinate between the producing and consuming applications, so a change in the meaning of the real time data can result in unexpected operation. Devices have the option to reject a reconfiguration request, with an Object state conflict error (general status = 0x0c), to prevent this situation.