



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 61158-6-2:2023. It is identical to IEC 61158-6-2:2023. It supersedes BS EN IEC 61158-6-2:2019, which is withdrawn.

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

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

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2024
Published by BSI Standards Limited 2024

ISBN 978 0 539 26666 5

ICS 25.040.40; 35.100.70; 35.110

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 31 March 2024.

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 61158-6-2:2023)

Réseaux de communication industriels - Spécifications des
bus de terrain - Partie 6-2: Spécification du protocole de la
couche liaison de données - Eléments de type 2
(IEC 61158-6-2:2023)

Industrielle Kommunikationsnetze - Feldbusse - Teil 6-2:
Protokollspezifikation des Application Layer
(Anwendungsschicht) - Typ 2-Elemente
(IEC 61158-6-2:2023)

This European Standard was approved by CENELEC on 2023-04-28. 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, Türkiye 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/1204/FDIS, future edition 5 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 61158-6-2:2023.

The following dates are fixed:

- latest date by which the document has to be implemented at national (dop) 2024-01-28 level by publication of an identical national standard or by endorsement
- latest date by which the national standards conflicting with the (dow) 2026-04-28 document have to be withdrawn

This document supersedes EN IEC 61158-6-2:2019 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.

Any feedback and questions on this document should be directed to the users' national committee. A complete listing of these bodies can be found on the CENELEC website.

Endorsement notice

The text of the International Standard IEC 61158-6-2:2023 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 standard indicated:

IEC 61131-3	NOTE Approved as EN 61131-3
IEC 61131-9	NOTE Approved as EN IEC 61131-9
IEC 61784-1 (series)	NOTE Approved as EN IEC 61784-1 (series)
IEC 61784-2 (series)	NOTE Approved as EN IEC 61784-2 (series)

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	2023	Industrial communication networks - Fieldbus specifications - Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series	-	-
IEC 61158-3-2	2023	Industrial communication networks - Fieldbus specifications - Part 3-2: Data-link layer service definition - Type 2 elements	-	-
IEC 61158-4-2	2023	Industrial communication networks - Fieldbus specifications - Part 4-2: Data-link layer protocol specification - Type 2 elements	EN IEC 61158-4-2	2023
IEC 61158-5-2	2023	Industrial communication networks - Fieldbus specifications - Part 5-2: Application layer service definition - Type 2 elements	-	-
IEC 61588	2021	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	EN IEC 61784-3-2	-
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/IEC 7498-1	-	Information technology - Open Systems Interconnection - Basic reference model: The basic model	-	-
ISO/IEC/IEEE 8802-3	-	Telecommunications and exchange between information technology systems - Requirements for local and metropolitan area networks - Part 3: Standard for Ethernet	-	-

EN IEC 61158-6-2:2023 (E)

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
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 11898-1	2015	Road vehicles - Controller area network (CAN) - Part 1: Data link layer and physical signalling	-	-
IEEE 802.1Q	2018	IEEE Standard for Local and Metropolitan Area Networks¿Bridges and Bridged Networks	-	-
IEEE 802.3	2018	IEEE Standard for Ethernet	-	-
IETF RFC 791	-	Internet Protocol	-	-
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	-	-

CONTENTS

FOREWORD.....	14
INTRODUCTION.....	17
1 Scope.....	18
1.1 General.....	18
1.2 Specifications	18
1.3 Conformance	19
2 Normative references	19
3 Terms, definitions, symbols, abbreviated terms and conventions	21
3.1 Terms and definitions from other ISO/IEC standards.....	21
3.1.1 Terms and definitions from ISO/IEC 7498-1	21
3.1.2 Terms and definitions from ISO/IEC 9545	21
3.1.3 Terms and definitions from ISO/IEC 8824-1	22
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	30
3.5 Conventions.....	30
3.5.1 General concept	30
3.5.2 Attribute specification	31
3.5.3 Common services	31
3.5.4 State machine conventions	34
4 Abstract syntax.....	36
4.1 FAL PDU abstract syntax	36
4.1.1 General	36
4.1.2 PDU structure.....	36
4.1.3 UCMM_PDUs	39
4.1.4 Transport_Headers.....	40
4.1.5 CM_PDUs.....	43
4.1.6 CM PDU components	56
4.1.7 MR headers	65
4.1.8 OM_Service_PDU.....	66
4.1.9 Message and connection paths.....	130
4.1.10 Class, attribute and service codes	146
4.1.11 Error codes.....	154
4.2 Data abstract syntax specification.....	169
4.2.1 Transport format specification.....	169
4.2.2 Abstract syntax notation	170
4.2.3 Control network data specification	170
4.2.4 Data type specification / dictionaries.....	172
4.3 Encapsulation abstract syntax.....	174
4.3.1 Encapsulation protocol	174
4.3.2 Encapsulation messages	175
4.3.3 Command descriptions	178
4.3.4 Common packet format.....	190
5 Transfer syntax.....	194
5.1 Compact encoding	194

5.1.1	Encoding rules.....	194
5.1.2	Encoding constraints	195
5.1.3	Examples.....	195
5.2	Data type reporting	202
5.2.1	Object data representation.....	202
5.2.2	Elementary data type reporting	202
5.2.3	Constructed data type reporting.....	203
6	Structure of FAL protocol state machines	209
7	AP-Context state machine	210
7.1	Overview.....	210
7.2	Connection object state machine	210
7.2.1	I/O Connection instance behavior	210
7.2.2	Bridged Connection instance behavior	215
7.2.3	Explicit Messaging Connection instance behavior	217
8	FAL service protocol machine (FSPM).....	219
8.1	General.....	219
8.2	Primitive definitions.....	219
8.3	Parameters of primitives	224
8.4	FSPM state machines	225
9	Application relationship protocol machines (ARPMs)	225
9.1	General.....	225
9.2	Connection-less ARPM (UCMM)	226
9.2.1	General	226
9.2.2	Primitive definitions	226
9.2.3	Parameters of primitives	227
9.2.4	UCMM state machines.....	228
9.2.5	Examples of UCMM sequences	233
9.2.6	Management UCMM	235
9.3	Connection-oriented ARPMs (transports)	236
9.3.1	Transport PDU buffer.....	236
9.3.2	Transport classes	237
9.3.3	Common primitive definitions.....	237
9.3.4	Parameters of common primitives	238
9.3.5	Transport state machines – class 0.....	238
9.3.6	Transport state machines – class 1.....	242
9.3.7	Transport state machines – class 2.....	247
9.3.8	Transport state machines – class 3.....	255
10	DLL mapping protocol machine 1 (DMPM 1).....	265
10.1	General.....	265
10.2	Link producer	265
10.3	Link consumer	266
10.4	Primitive definitions.....	266
10.4.1	Primitives exchanged between DMPM and ARPM.....	266
10.4.2	Parameters of ARPM/DMPM primitives	266
10.4.3	Primitives exchanged between data-link layer and DMPM.....	266
10.4.4	Parameters of DMPM/Data-link Layer primitives	267
10.4.5	Network connection ID.....	268
10.5	DMPM state machine	268

10.5.1	DMPM states	268
10.5.2	Functions used by DMPM	270
10.6	Data-link Layer service selection.....	270
11	DLL mapping protocol machine 2 (DMPM 2)	270
11.1	General.....	270
11.2	Mapping of UCMM PDUs	270
11.2.1	General	270
11.2.2	Common requirements for Connection Manager PDU's.....	272
11.2.3	Forward_Open PDU for class 2 and class 3 connections	274
11.2.4	Forward_Open for class 0 and class 1 connections.....	274
11.2.5	Forward_close	279
11.3	Mapping of transport class 0 and class 1 PDUs.....	279
11.3.1	Class 0 and class 1 PDUs.....	279
11.3.2	No dependency on TCP connections	279
11.3.3	Class 0 and class 1 packet ordering	280
11.3.4	Screening incoming connected data	280
11.4	Mapping of transport class 2 and class 3 PDU's.....	280
11.5	IGMP Usage	281
11.5.1	Background (informative).....	281
11.5.2	IGMP Membership Report messages	282
11.5.3	IGMP Leave Group messages	282
11.6	Quality of Service (QoS) for Type 2 Ethernet messages.....	282
11.6.1	Overview	282
11.6.2	DSCP format	283
11.6.3	IEEE Std 802.1Q-2018 format.....	284
11.6.4	Mapping Type 2 traffic to DSCP and IEEE Std 802.1Q-2018.....	284
11.6.5	Usage of DSCP for Type 2 Ethernet	285
11.6.6	Usage of IEEE Std 802.1Q-2018 for Type 2 Ethernet.....	285
11.6.7	User considerations with IEEE Std 802.1Q-2018	286
11.7	Encapsulation using TCP	286
11.7.1	General	286
11.7.2	Management of a TCP encapsulation session.....	286
11.7.3	TCP connection management	287
11.8	Encapsulation using UDP.....	288
12	DLL mapping protocol machine 3 (DMPM 3)	288
	Bibliography.....	289
	Figure 1 – Attribute table format and terms	31
	Figure 2 – Service request/response parameter	31
	Figure 3 – Example of an STD	34
	Figure 4 – Network connection parameters	56
	Figure 5 – Priority/Tick_time bit definition	59
	Figure 6 – Member ID/EX description (WORD).....	74
	Figure 7 – Transport Class Trigger attribute.....	118
	Figure 8 – DN_initial_comm_characteristics attribute format	122
	Figure 9 – Segment type.....	131
	Figure 10 – Port segment.....	132

Figure 11 – Logical segment encoding	134
Figure 12 – Extended network segment	141
Figure 13 – Symbolic segment encoding	142
Figure 14 – Encapsulation message	175
Figure 15 – FixedLengthBitString compact encoding bit placement rules	199
Figure 16 – Example compact encoding of a SWORD FixedLengthBitString	199
Figure 17 – Example compact encoding of a WORD FixedLengthBitString	199
Figure 18 – Example compact encoding of a DWORD FixedLengthBitString	200
Figure 19 – Example compact encoding of a LWORD FixedLengthBitString	200
Figure 20 – Example 1 of formal encoding of a structure type specification	205
Figure 21 – Example 2 of formal encoding of a structure type specification	205
Figure 22 – Example 3 of formal encoding of a handle structure type specification	206
Figure 23 – Example 4 of formal encoding of a handle structure type specification	206
Figure 24 – Example 5 of abbreviated encoding of a structure type specification	207
Figure 25 – Example 1 of formal encoding of an array type specification	207
Figure 26 – Example 2 of formal encoding of an array type specification	208
Figure 27 – Example 1 of abbreviated encoding of an array type specification	209
Figure 28 – Example 2 of abbreviated encoding of an array type specification	209
Figure 29 – I/O Connection object state transition diagram	210
Figure 30 – Bridged Connection object state transition diagram	215
Figure 31 – Explicit Messaging Connection object state transition diagram	217
Figure 32 – State transition diagram of UCMM client9	229
Figure 33 – State transition diagram of high-end UCMM server	231
Figure 34 – State transition diagram of low-end UCMM server	233
Figure 35 – Sequence diagram for a UCMM with one outstanding message	234
Figure 36 – Sequence diagram for a UCMM with multiple outstanding messages	235
Figure 37 – TPDU buffer	236
Figure 38 – Data flow diagram using a client transport class 0 and server transport class 0	239
Figure 39 – Sequence diagram of data transfer using transport class 0	239
Figure 40 – Class 0 client STD	240
Figure 41 – Class 0 server STD	241
Figure 42 – Data flow diagram using client transport class 1 and server transport class 1	242
Figure 43 – Sequence diagram of data transfer using client transport class 1 and server transport class 1	243
Figure 44 – Class 1 client STD	245
Figure 45 – Class 1 server STD	246
Figure 46 – Data flow diagram using client transport class 2 and server transport class 2	248
Figure 47 – Diagram of data transfer using client transport class 2 and server transport class 2 without returned data	249
Figure 48 – Sequence diagram of data transfer using client transport class 2 and server transport class 2 with returned data	250
Figure 49 – Class 2 client STD	

Figure 50 – Class 2 server STD	253
Figure 51 – Data flow diagram using client transport class 3 and server transport class 3	256
Figure 52 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 without returned data	257
Figure 53 – Sequence diagram of data transfer using client transport class 3 and server transport class 3 with returned data	258
Figure 54 – Class 3 client STD	260
Figure 55 – Class 3 server STD	263
Figure 56 – Data flow diagram for a link producer and consumer	265
Figure 57 – State transition diagram for a link producer	269
Figure 58 – State transition diagram for a link consumer	270
Figure 59 – DS field in the IP header	284
Figure 60 – IEEE Std 802.1Q-2018 tagged frame	284
Table 1 – Get_Attributes_All response service rules	32
Table 2 – Example class level object/service specific response data of Get_Attributes_All	32
Table 3 – Set_Attributes_All request service rules	33
Table 4 – Example request data of Set_Attributes_All	33
Table 5 – State event matrix format	35
Table 6 – Example state event matrix	35
Table 7 – UCMM_PDU header format	39
Table 8 – UCMM command codes	39
Table 9 – Transport class 0 header	40
Table 10 – Transport class 1 header	40
Table 11 – Transport class 2 header	40
Table 12 – Transport class 3 header	41
Table 13 – Real-time data header – exclusive owner	41
Table 14 – Real-time data header– redundant owner	41
Table 15 – Forward_Open request format	46
Table 16 – Forward_Open_Good response format	47
Table 17 – Forward_Open_Bad response format	47
Table 18 – Large_Forward_Open request format	48
Table 19 – Large_Forward_Open_Good response format	49
Table 20 – Large_Forward_Open_Bad response format	49
Table 21 – Forward_Close request format	50
Table 22 – Forward_Close_Good response format	50
Table 23 – Forward_Close_Bad response format	51
Table 24 – Unconnected_Send request format	52
Table 25 – Unconnected_Send_Good response format	52
Table 26 – Unconnected_Send_Bad response format	53
Table 27 – Get_Connection_Data request format	54
Table 28 – Get_Connection_Data response format	54

Table 29 – Search_Connection_Data request format	55
Table 30 – Get_Connection_Owner request format	55
Table 31 – Get_Connection_Owner response format	56
Table 32 – Time-out multiplier.....	59
Table 33 – Tick time units	60
Table 34 – Encoded application path ordering.....	64
Table 35 – Transport class, trigger and Is_Server format	65
Table 36 – MR_Request_Header format	65
Table 37 – MR_Response_Header format.....	66
Table 38 – Structure of Get_Attributes_All_ResponsePDU body	66
Table 39 – Structure of Set_Attributes_All_RequestPDU body	67
Table 40 – Structure of Get_Attribute_List_RequestPDU body	67
Table 41 – Structure of Get_Attribute_List_ResponsePDU body	67
Table 42 – Structure of Set_Attribute_List_RequestPDU body	67
Table 43 – Structure of Set_Attribute_List_ResponsePDU body.....	68
Table 44 – Structure of Reset_RequestPDU body	68
Table 45 – Structure of Reset_ResponsePDU body	68
Table 46 – Structure of Start_RequestPDU body	68
Table 47 – Structure of Start_ResponsePDU body.....	68
Table 48 – Structure of Stop_RequestPDU body.....	69
Table 49 – Structure of Stop_ResponsePDU body	69
Table 50 – Structure of Create_RequestPDU body	69
Table 51 – Structure of Create_ResponsePDU body.....	69
Table 52 – Structure of Delete_RequestPDU body.....	69
Table 53 – Structure of Delete_ResponsePDU body	70
Table 54 – Structure of Get_Attribute_Single_ResponsePDU body	70
Table 55 – Structure of Set_Attribute_Single_RequestPDU body	70
Table 56 – Structure of Set_Attribute_Single_ResponsePDU body	70
Table 57 – Structure of Find_Next_Object_Instance_RequestPDU body	71
Table 58 – Structure of Find_Next_Object_Instance_ResponsePDU body	71
Table 59 – Structure of Apply_Attributes_RequestPDU body	71
Table 60 – Structure of Apply_Attributes_ResponsePDU body	71
Table 61 – Structure of Save_RequestPDU body	71
Table 62 – Structure of Save_ResponsePDU body	72
Table 63 – Structure of Restore_RequestPDU body.....	72
Table 64 – Structure of Restore_ResponsePDU body	72
Table 65 – Structure of Get_Member_ResponsePDU body	72
Table 66 – Structure of Set_Member_RequestPDU body	72
Table 67 – Structure of Set_Member_ResponsePDU body.....	73
Table 68 – Structure of Insert_Member_RequestPDU body.....	73
Table 69 – Structure of Insert_Member_ResponsePDU body	73
Table 70 – Structure of Remove_Member_ResponsePDU body	73
Table 71 – Common structure of _Member_RequestPDU body (basic format).....	74

Table 72 – Common structure of _Member_ResponsePDU body (basic format)	75
Table 73 – Common structure of _Member_RequestPDU body (extended format).....	75
Table 74 – Common structure of _Member_ResponsePDU body (extended format)	75
Table 75 – Extended Protocol ID.....	76
Table 76 – Structure of _Member_RequestPDU body (Multiple Sequential Members)	76
Table 77 – Structure of _Member_ResponsePDU body (Multiple Sequential Members).....	76
Table 78 – Structure of _Member_RequestPDU body (International String Selection)	77
Table 79 – Structure of _Member_ResponsePDU body (International String Selection).....	77
Table 80 – Structure of Group_Sync_RequestPDU body.....	77
Table 81 – Structure of Group_Sync_ResponsePDU body	78
Table 82 – Structure of Multiple_Service_Packet_RequestPDU body	78
Table 83 – Structure of Multiple_Service_Packet_ResponsePDU body	78
Table 84 – Structure of Get_Connection_Point_Member_List_ResponsePDU body.....	79
Table 85 – Identity object class attributes	80
Table 86 – Identity object instance attributes	80
Table 87 – Identity object Vendor ID ranges	83
Table 88 – Identity object bit definitions for status instance attribute	83
Table 89 – Default values for extended device status field (bits 4 to 7) of status instance attribute	83
Table 90 – Identity object bit definitions for protection mode instance attribute	84
Table 91 – Identity object bit definitions for features supported attribute	84
Table 92 – Class level object/service specific response data of Get_Attributes_All	84
Table 93 – Instance level object/service specific response data of Get_Attributes_All.....	85
Table 94 – Object-specific request parameter for Reset	86
Table 95 – Reset service parameter values	86
Table 96 – Communication link attributes that shall be preserved	86
Table 97 – Structure of Flash_LEDs_RequestPDU body	87
Table 98 – Message Router object class attributes	87
Table 99 – Message Router object instance attributes	87
Table 100 – Class level object/service specific response data of Get_Attributes_All	88
Table 101 – Instance level object/service specific response data of Get_Attributes_All.....	88
Table 102 – Structure of Symbolic_Translation_RequestPDU body.....	88
Table 103 – Structure of Symbolic_Translation_ResponsePDU body	88
Table 104 – Object specific status for Symbolic_Translation service	89
Table 105 – Structure of Send_Receive_Fragment_RequestPDU body – Phase 1	89
Table 106 – Structure of Send_Receive_Fragment_RequestPDU body – Phase 2	89
Table 107 – Structure of Send_Receive_Fragment_ResponsePDU body – Phase 2.....	90
Table 108 – Request/Response Fragmentation Flags	90
Table 109 – Fragmentation Flags Usage.....	90
Table 110 – Object specific status for Send_Receive_Fragment service	91
Table 111 – Assembly object class attributes.....	92
Table 112 – Assembly object instance attributes.....	92
Table 113 – Assembly Instance ID ranges	93

Table 114 – Standard Network Diagnostic assembly content and ordering	94
Table 115 – Object-specific request parameter for Create.....	94
Table 116 – Object-specific response parameter for Create	95
Table 117 – Acknowledge Handler object class attributes	95
Table 118 – Acknowledge Handler object instance attributes	95
Table 119 – Structure of Add_AckData_Path_RequestPDU body	96
Table 120 – Structure of Remove_AckData_Path_RequestPDU body	96
Table 121 – Time Sync object class attributes	96
Table 122 – Time Sync object instance attributes	97
Table 123 – ClockIdentity encoding for different network implementations	101
Table 124 – ClockClass values	101
Table 125 – TimeAccuracy values.....	102
Table 126 – TimePropertyFlags bit values	102
Table 127 – TimeSource values.....	103
Table 128 – Types of Clock	103
Table 129 – Network protocol to PortPhysicalAddressInfo mapping	103
Table 130 – Time Sync connection point 1, Standard Network Diagnostics	104
Table 131 – Class level object/service specific response data of Get_Attributes_All	104
Table 132 – Parameter object class attributes.....	105
Table 133 – Parameter Class Descriptor bit values	105
Table 134 – Parameter object instance attributes.....	106
Table 135 – Semantics of Descriptor Instance attribute.....	107
Table 136 – Descriptor Scaling bits usage	107
Table 137 – Minimum and Maximum Value semantics.....	108
Table 138 – Scaling Formula attributes	109
Table 139 – Scaling links	109
Table 140 – Class level object/service specific response data of Get_Attributes_All	110
Table 141 – Instance level object/service specific response data of Get_Attributes_All (Parameter object stub)	110
Table 142 – Instance level object/service specific response data of Get_Attributes_All (full Parameter object)	111
Table 143 – Structure of Get_Enum_String_RequestPDU body.....	112
Table 144 – Structure of Get_Enum_String_ResponsePDU body	112
Table 145 – Enumerated strings Type versus Parameter data type	112
Table 146 – Connection Manager object class attributes.....	113
Table 147 – Connection Manager object instance attributes.....	113
Table 148 – Connection Manager connection point 1, Standard Network Diagnostics	114
Table 149 – Class level object/service specific response data of Get_Attributes_All	114
Table 150 – Instance level object/service specific response data of Get_Attributes_All.....	115
Table 151 – Instance level object/service specific request data of Set_Attributes_All.....	115
Table 152 – Connection object class attributes	116
Table 153 – Connection object instance attributes	116
Table 154 – Values assigned to the state attribute	117
Table 155 – Values assigned to the instance_type attribute	118

Table 156 – Possible values within Direction Bit	119
Table 157 – Possible values within Production Trigger Bits	119
Table 158 – Possible values within Transport Class Bits	120
Table 159 – TransportClass_Trigger attribute values summary	120
Table 160 – Transport Class 0 client behavior summary	121
Table 161 – Transport Class 1, 2 and 3 client behavior summary	121
Table 162 – Values defined for the DN_produced_connection_id attribute	121
Table 163 – Values defined for the DN_consumed_connection_id attribute	122
Table 164 – Values for the Initial Production Characteristics nibble	123
Table 165 – Values for the Initial Consumption Characteristics nibble	124
Table 166 – Values for the watchdog_timeout_action	127
Table 167 – Object-specific response parameters for Apply_Attributes	129
Table 168 – Object-specific response parameter for Set_Attribute_Single	129
Table 169 – Structure of Connection_Bind_RequestPDU body	129
Table 170 – Object specific status for Connection_Bind service	129
Table 171 – Structure of Producing_Application_Lookup_RequestPDU body	130
Table 172 – Structure of Producing_Application_Lookup_ResponsePDU body	130
Table 173 – Producing_Application_Lookup Service status codes	130
Table 174 – Possible port segment examples	133
Table 175 – TCP/IP link address examples	134
Table 176 – Extended Logical Type	135
Table 177 – Electronic key segment format	136
Table 178 – Key Format Table (key type 4)	137
Table 179 – Serial Number Key Format Table (key type 5)	137
Table 180 – Logical segments examples	138
Table 181 – Network segments	139
Table 182 – Extended network segment subtype definitions	141
Table 183 – Symbolic segment examples	143
Table 184 – Data segment	143
Table 185 – ANSI_Extended_Symbol segment	144
Table 186 – Addressing categories	146
Table 187 – Class code ID ranges	147
Table 188 – Class Attribute ID ranges	147
Table 189 – Instance Attribute ID ranges	147
Table 190 – Connection Point ranges	148
Table 191 – Service code ranges	148
Table 192 – Class codes	149
Table 193 – Reserved class attributes for all object class definitions	150
Table 194 – Common services list	151
Table 195 – Identity object specific services list	151
Table 196 – Message Router object specific services list	152
Table 197 – Acknowledge Handler object specific services list	152
Table 198 – Parameter object specific services list	152

Table 199 – Services specific to Connection Manager	152
Table 200 – Services specific to Connection object.....	153
Table 201 – Device type numbering	153
Table 202 – Implementation profile numbering.....	154
Table 203 – Connection Manager service request error codes	155
Table 204 – General status codes.....	165
Table 205 – Extended status code for a general status of "Key Failure in path.....	167
Table 206 – Identity object status codes	168
Table 207 – TCP port numbers	174
Table 208 – UDP port numbers.....	175
Table 209 – Encapsulation header	175
Table 210 – Encapsulation command codes	176
Table 211 – Encapsulation status codes	177
Table 212 – Nop request encapsulation header	178
Table 213 – RegisterSession request encapsulation header	179
Table 214 – RegisterSession request data portion	179
Table 215 – RegisterSession reply encapsulation header	180
Table 216 – RegisterSession reply data portion (successful)	180
Table 217 – UnRegisterSession request encapsulation header	181
Table 218 – ListServices request encapsulation header.....	182
Table 219 – ListServices reply encapsulation header.....	182
Table 220 – ListServices reply data portion (successful).....	182
Table 221 – Communications capability flags.....	183
Table 222 – ListIdentity request encapsulation header.....	184
Table 223 – ListIdentity reply encapsulation header.....	185
Table 224 – ListIdentity reply data portion (successful).....	185
Table 225 – Type 2 identity item	186
Table 226 – Type 2 Ethernet Capability item.....	186
Table 227 – ListInterfaces request encapsulation header.....	187
Table 228 – ListInterfaces reply encapsulation header.....	187
Table 229 – SendRRData request encapsulation header	188
Table 230 – SendRRData request data portion	188
Table 231 – SendRRData reply encapsulation header	189
Table 232 – SendUnitData request encapsulation header	189
Table 233 – SendUnitData request data portion.....	190
Table 234 – Common packet format.....	190
Table 235 – CPF item format	190
Table 236 – Item Type ID numbers	191
Table 237 – Null address item.....	191
Table 238 – Connected address item.....	192
Table 239 – Sequenced address item	192
Table 240 – Unconnected data item.....	192
Table 241 – Connected data item	193

Table 242 – Sockaddr info items	193
Table 243 – Usage of CPF items	194
Table 244 – BOOLEAN encoding	195
Table 245 – Example compact encoding of a BOOL value	195
Table 246 – Encoding of SignedInteger values	196
Table 247 – Example compact encoding of a SignedInteger value	196
Table 248 – UnsignedInteger values	196
Table 249 – Example compact encoding of an UnsignedInteger	196
Table 250 – FixedLengthReal values	196
Table 251 – Example compact encoding of a REAL value	197
Table 252 – Example compact encoding of a LREAL value	197
Table 253 – FixedLengthReal values	197
Table 254 – STRING value	198
Table 255 – STRING2 value	198
Table 256 – STRINGN value	198
Table 257 – SHORT_STRING value	198
Table 258 – Example compact encoding of a STRING value	198
Table 259 – Example compact encoding of STRING2 value	199
Table 260 – SHORT_STRING type	199
Table 261 – Example compact encoding of a single dimensional ARRAY	200
Table 262 – Example compact encoding of a multi-dimensional ARRAY	201
Table 263 – Example compact encoding of a STRUCTURE	201
Table 264 – Identification codes and descriptions of elementary data types	203
Table 265 – Identification codes and descriptions of constructed data types	204
Table 266 – Formal structure encoding definition	204
Table 267 – Formal structure with handles encoding definition	205
Table 268 – Abbreviated structure encoding definition	206
Table 269 – Formal array encoding definition	207
Table 270 – Abbreviated array encoding definition	208
Table 271 – I/O Connection state event matrix	211
Table 272 – Bridged Connection state event matrix	216
Table 273 – Explicit Messaging Connection state event matrix	217
Table 274 – Primitives issued by FAL user to FSPM	220
Table 275 – Primitives issued by FAL user to FSPM	221
Table 276 – Primitives issued by FSPM to FAL user	223
Table 277 – Parameters used with primitives exchanged between FAL user and FSPM	225
Table 278 – Primitives issued by FSPM to ARPM	227
Table 279 – Primitives issued by ARPM to FSPM	227
Table 280 – Parameters used with primitives exchanged between FSPM and ARPM	228
Table 281 – UCMM client states	228
Table 282 – State event matrix of UCMM client	229
Table 283 – High-end UCMM server states	230
Table 284 – State event matrix of high-end UCMM server	231

Table 285 – Low-end UCMM server states	232
Table 286 – State event matrix of low-end UCMM server	233
Table 287 – Notification	236
Table 288 – Transport classes	237
Table 289 – Primitives issued by FSPM to ARPM	237
Table 290 – Primitives issued by ARPM to FSPM	238
Table 291 – Parameters used with primitives exchanged between FSPM and ARPM	238
Table 292 – Class 0 transport client states	240
Table 293 – Class 0 client SEM	240
Table 294 – Class 0 transport server states	241
Table 295 – Class 0 server SEM	241
Table 296 – Class 1 transport client states	244
Table 297 – Class 1 client SEM	245
Table 298 – Class 1 transport server states	246
Table 299 – Class 1 server SEM	247
Table 300 – Class 2 transport client states	251
Table 301 – Class 2 client SEM	252
Table 302 – Class 2 transport server states	253
Table 303 – Class 2 server SEM	254
Table 304 – Class 3 transport client states	259
Table 305 – Class 3 client SEM	260
Table 306 – Class 3 transport server states	262
Table 307 – Class 3 server SEM	264
Table 308 – Primitives issued by ARPM to DMPM	266
Table 309 – Primitives issued by DMPM to ARPM	266
Table 310 – Parameters used with primitives exchanged between ARPM and DMPM	266
Table 311 – Primitives exchanged between data-link layer and DMPM	267
Table 312 – Parameters used with primitives exchanged between DMPM and Data-link	267
Table 313 – Selection of connection ID	268
Table 314 – Link producer states	268
Table 315 – State event matrix of link producer	269
Table 316 – Link consumer states	269
Table 317 – State event matrix of link consumer	270
Table 318 – UCMM request	271
Table 319 – UCMM reply	272
Table 320 – Network Connection ID selection	273
Table 321 – Sockaddr Info usage	275
Table 322 – Example multicast assignments	278
Table 323 – UDP data format for class 0 and class 1	279
Table 324 – Transport class 2 and class 3 connected data	281
Table 325 – Default DSCP and IEEE Std 802.1Q-2018 mapping	285

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 the IEC 61784-1 series and the IEC 61784-2 series.

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

This fifth edition cancels and replaces the fourth edition published in 2019. This edition constitutes a technical revision.

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

- a) update of normative and bibliographic references;
- b) review of Get/Set_Attributes_All parameter format in 3.5.3;
- c) new services in 4.1.2.1, 4.1.8.1 and 8.2, 8.3;
- d) clarifications of services in 4.1.5;
- e) definition of specific connection path in 4.1.6.12;
- f) clarifications and updates of Get/Set_attribute_list services in 4.1.8.1;
- g) clarifications, new attributes for the Identity object in 4.1.8.2;
- h) new attributes, service parameters and service for the Message Router object in 4.1.8.3;
- i) clarifications, new attribute and other extensions for the Assembly object in 4.1.8.4;
- j) clarifications, new attributes, service parameters, services and diagnostics connection points for the Time Sync object in 4.1.8.6;
- k) clarifications, new services and addition of diagnostics connection points for the Connection Manager object in 4.1.8.9;
- l) clarifications and extensions of Path Segments in 4.1.9;
- m) updates and extensions of class, attribute and service codes in 4.1.10;
- n) clarifications and additions of error codes in 4.1.11;
- o) update of STIME, UTIME and NTIME data types in 4.2.3 and 5.1.3.5;
- p) updates of encapsulation protocol in 4.3.1;
- q) addition of internal services in 7.1;
- r) removal of obsoleted transport options and related services in Clause 9 and Clause 11;
- s) updates of DMPM2 in Clause 11;
- t) removal of all references to CPF and CPs (material moved to profile documents);
- u) miscellaneous editorial corrections.

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

Draft	Report on voting
65C/1204/FDIS	65C/1245/RVD

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

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

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 document will remain unchanged until the stability date indicated on the IEC website under webstore.iec.ch in the data related to the specific document. At this date, the document will be

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

INTRODUCTION

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

The application protocol provides the application service by making use of the services available from the data-link or other immediately lower layer. The primary aim of this 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 can 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 document specifies interactions between remote applications and defines the externally visible behavior provided by the Type 2 fieldbus application layer in terms of

- the formal abstract syntax defining the application layer protocol data units conveyed between communicating application entities;
- the transfer syntax defining encoding rules that are applied to the application layer protocol data units;
- the application context state machine defining the application service behavior visible between communicating application entities;
- 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

- define the wire-representation of the service primitives defined in IEC 61158-5-2, and
- 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 the IEC 61784-1 series and the IEC 61784-2 series 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 61158-1:2023, *Industrial communication networks – Fieldbus specifications – Part 1: Overview and guidance for the IEC 61158 and IEC 61784 series*

IEC 61158-3-2:2023, *Industrial communication networks – Fieldbus specifications – Part 3-2: Data-link layer service definition – Type 2 elements*

IEC 61158-4-2:2023, *Industrial communication networks – Fieldbus specifications – Part 4-2: Data-link layer protocol specification – Type 2 elements*

IEC 61158-5-2:2023, *Industrial communication networks – Fieldbus specifications – Part 5-2: Application layer service definition – Type 2 elements*

IEC 61588:2021, *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, *Telecommunications and exchange between information technology systems – Requirements for local and metropolitan area networks – Part 3: Standard for Ethernet*

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

ISO/IEC 8825-1, *Information technology – ASN.1 encoding rules – Part 1: 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 11898-1:2015, *Road vehicles – Controller area network (CAN) – Part 1: Data link layer and physical signalling*

IEEE Std 802.1Q-2018, *IEEE standard for local and metropolitan area networks – Bridges and bridged networks*

IEEE Std 802.3-2018, *IEEE Standard for Ethernet*

IETF RFC 791, J. Postel, *Internet Protocol*, September 1981, available at <https://www.rfc-editor.org/info/rfc791> [viewed 2022-02-18]

IETF RFC 1035, P.V. Mockapetris, *Domain Names – Implementation and Specification*, November 1987, available at <https://www.rfc-editor.org/info/rfc1035> [viewed 2022-02-18]

IETF RFC 1112, S.E. Deering, *Host Extensions for IP Multicasting*, August 1989, available at <https://www.rfc-editor.org/info/rfc1112> [viewed 2022-02-18]

IETF RFC 1117, S. Romano, M.K. Stahl, M. Recker, *Internet Numbers*, August 1989, available at <https://www.rfc-editor.org/info/rfc1117> [viewed 2022-02-18]

IETF RFC 1122, R. Braden, *Requirements for Internet Hosts – Communication Layers*, October 1989, available at <https://www.rfc-editor.org/info/rfc1122> [viewed 2022-02-18]

IETF RFC 1759, R. Smith, F. Wright, T. Hastings, S. Zilles, J. Gyllenskog, *Printer MIB*, March 1995, available at <https://www.rfc-editor.org/info/rfc1759> [viewed 2022-02-18]

IETF RFC 2236, W. Fenner, *Internet Group Management Protocol, Version 2*, November 1997, available at <https://www.rfc-editor.org/info/rfc2236> [viewed 2022-02-18]

IETF RFC 2474, K. Nichols, S. Blake, F. Baker, D. Black, *Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers*, December 1998, available at <https://www.rfc-editor.org/info/rfc2474> [viewed 2022-02-18]

IETF RFC 2475, S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, *An Architecture for Differentiated Services*, December 1998, available at <https://www.rfc-editor.org/info/rfc2475> [viewed 2022-02-18]

IETF RFC 2597, J. Heinanen, F. Baker, W. Weiss, J. Wroclawski, *Assured Forwarding PHB Group*, June 1999, available at <https://www.rfc-editor.org/info/rfc2597> [viewed 2022-02-18]

IETF RFC 2873, X. Xiao, A. Hannan, V. Paxson, E. Crabbe, *TCP Processing of the IPv4 Precedence Field*, June 2000, available at <https://www.rfc-editor.org/info/rfc2873> [viewed 2022-02-18]

IETF RFC 3140, D. Black, S. Brim, B. Carpenter, F. Le Faucheur, *Per Hop Behavior Identification Codes*, June 2001, available at <https://www.rfc-editor.org/info/rfc3140> [viewed 2022-02-18]

IETF RFC 3246, B. Davie, A. Charny, J.C.R. Bennet, K. Benson, J.Y. Le Boudec, W. Courtney, S. Davari, V. Firoiu, D. Stiliadis, *An Expedited Forwarding PHB (Per-Hop Behavior)*, March 2002, available at <https://www.rfc-editor.org/info/rfc3246> [viewed 2022-02-18]

IETF RFC 3376, B. Cain, S. Deering, I. Kouvelas, B. Fenner, A. Thyagarajan, *Internet Group Management Protocol, Version 3*, October 2002, available at <https://www.rfc-editor.org/info/rfc3376> [viewed 2022-02-18]

IETF RFC 4594, J. Babiarez, K. Chan, F. Baker, *Configuration Guidelines for DiffServ Service Classes*, August 2006, available at <https://www.rfc-editor.org/info/rfc4594> [viewed 2022-02-18]

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 <https://www.electropedia.org/>
- ISO Online browsing platform: available at <https://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

- a) abstract syntax
- b) application entity
- c) application process
- d) application protocol data unit
- e) application service element
- f) application entity invocation
- g) application process invocation
- h) application transaction
- i) presentation context
- j) real open system
- k) 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 can 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 ¹ – 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

¹ This definition and several others are based on the legacy IEC 61588:2009 and not the latest IEC 61588:2021.

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 61588:2021, 3.1.12]

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

implementation profile

collection of information and functionality supported by a device, independently of its device type

Note 1 to entry: Unlike device profiles, implementation profiles group subsets of Type 2 technologies and characteristics, providing a high-level description of the capabilities of a device.

3.3.35

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.

3.3.36

instance attribute

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

3.3.37

instantiated

object that has been created in a device

3.3.38

interoperability

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

3.3.39

Keeper

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

3.3.40

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.

3.3.41

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.

3.3.42

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.

3.3.43**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]

3.3.44**member**

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

3.3.45**Message Router**

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

3.3.46**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.

3.3.47**network**

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

3.3.48**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

3.3.49**object specific service**

service unique to the object class which defines it

3.3.50**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]

3.3.51**originator**

client responsible for establishing a connection path to the target

3.3.52**parent clock**

master clock to which a clock is synchronized

[SOURCE: IEC 61588:2009, 3.1.23]

3.3.53

point-to-point connection

connection that exists between exactly two application objects

3.3.54

Precision Time Protocol

PTP

protocol defined by IEC 61588:2021

Note 1 to entry: As an adjective, it indicates that the modified noun is specified in or interpreted in the context of IEC 61588:2021.

[SOURCE: IEC 61588:2021, 3.1.48, modified – second sentence changed to a Note, and references to IEEE Std 1588 replaced by IEC 61588:2021]

3.3.55

produce

act of sending data to be received by a consumer

3.3.56

producer

node that is responsible for sending data

3.3.57

PTP message

one of the messages defined in IEC 61588:2021

[SOURCE: IEC 61588:2021, 3.1.58, modified – NOTE deleted, and reference to IEEE Std 1588 replaced by IEC 61588:2021]

3.3.58

PTP port

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

[SOURCE: IEC 61588:2021, 3.1.61, modified – “PTP instance” changed to “clock”]

3.3.59

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

3.3.60

resource

processing or information capability of a subsystem

3.3.61

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

3.3.62

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

3.3.63**service**

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

3.3.64**synchronized clocks**

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

[SOURCE: IEC 61588:2021, 3.1.75, modified – reworded according to IEC rules]

3.3.65**system time**

absolute time value as defined by Type 2 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 Type 2, System Time is a 64-bit integer value in units of nanoseconds with a value of 0 corresponding to the date 1970-01-01.

3.3.66**target**

end-node to which a connection is established

3.3.67**temporary node**

transient node

3.3.68**transaction id**

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

3.3.69**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]

3.3.70**unconnected message manager****UCMM**

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

3.3.71**unconnected service**

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

3.3.72**vendor ID**

identification of each product manufacturer/vendor by a unique number

Note 1 to entry: Vendor IDs are assigned by the ODVA, Inc. organization (see <www.odva.org>).

3.4 Abbreviated terms and symbols

ASCII	American Standard Code for Information Interchange
CID	connection ID
CAN	Controller Area Network (see ISO 11898-1)
CM_API	actual packet interval
CM_RPI	requested packet interval
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)
O2T or O⇒T	originator to target (connection parameters)
OSI	open systems interconnection (see ISO/IEC 7498-1)
PDU	protocol data unit
PHB	per-hop behavior
PTP	Precision Time Protocol (see IEC 61588)
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
T2O or T⇒O	target to originator (connection parameters)
TCP	Terminal Control Protocol (see IETF RFC 793)
ToS	type of service
TPDU	transport protocol data unit
TUI	table unique identifier
Tx	transmit
UDP	User Datagram Protocol (see IETF RFC 768)
Xmit	transmit

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	Data 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 structure of the data returned in the Service_ResponsePDU shall be specified.

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 how to specify 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

Attribute ID	Data type	Attribute Name	Default Value (if not implemented)
1	UINT	Revision	1
2	UINT	Max Instance	
3	UINT	Number of Instances	
4	STRUCT of	Optional attribute list	
	UINT	Number of attributes	0
	ARRAY of UINT	Optional attributes	(null)

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 structure of the data supplied in the Service_RequestPDU shall be specified.

The rules specified in Table 3 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 3 – 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 4 is an example of how to specify the service data portion of a Set_Attributes_All request for instance level attributes of an object which supports required settable instance attributes 1, 4, 10, 11, 15, 32.

Table 4 – Example request data of Set_Attributes_All

Attribute ID	Data type	Attribute name	Default Value (if not implemented)
1	UINT	Output Range	
4	USINT	Value Data Type	
10	UINT	Fault State	0
11	UINT	Idle State	0
15	UINT	Fault Value	0
32	UINT	Idle Value	0

3.5.3.3.2 Revisions

A server processing a Set_Attributes_All request can 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 can 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 can 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.

Behavior 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 behavior 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 can 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.

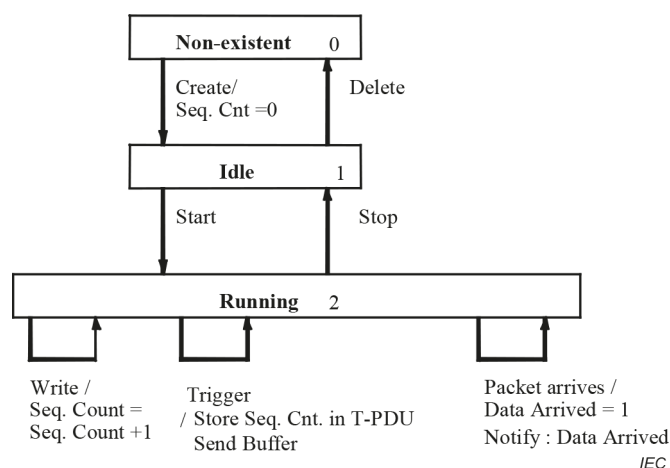


Figure 3 – Example of an STD

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 5.

Table 5 – State event matrix format

Event	State ^a		
	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)
^a 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 6 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 6 – 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 (routers). 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 routers. 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
- Send_Receive_Fragment_RequestPDU
- Connection_Bind_RequestPDU
- Producing_Application_Lookup_RequestPDU
- Flash_LEDs_RequestPDU
- Multiple_Service_Packet_RequestPDU
- Get_Connection_Point_Member_List_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
- Send_Receive_Fragment_RequestPDU
- Connection_Bind_ResponsePDU
- Producing_Application_Lookup_ResponsePDU
- Flash_LEDs_ResponsePDU
- Multiple_Service_Packet_ResponsePDU
- Get_Connection_Point_Member_List_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 TPDU's 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 7.

Table 7 – 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 8. UCMM packets received with a reserved `command_code` shall be discarded without acknowledgement.

Table 8 – 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 $(8 \mid \text{timeout} \& 7) << (\text{timeout} >> 3 \& 31)$.

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 9.

Table 9 – 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 10.

Table 10 – 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 11.

Table 11 – 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 12.

Table 12 – Transport class 3 header

Parameter name	Format
sequence_count	UINT

4.1.4.6 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).

4.1.4.7 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).

4.1.4.8 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 13.

Table 13 – 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.

4.1.4.9 Redundant owner O⇒T data format**4.1.4.9.1 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 14.

Table 14 – Real-time data header– redundant owner

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

4.1.4.9.2 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.

4.1.4.9.3 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 behavior if more than one other connection has its COO flag set.

4.1.4.9.4 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.

4.1.4.9.5 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.

4.1.4.9.6 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 behavior 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).

A client shall interpret any other error response returned by the target to mean that the target exists but does not support the requested function.

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. Routers shall always forward these requests. The associated behaviors 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).